МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут
комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи

за темою Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python, та розробка програмного забезпечення для її реалізації

Виконав: студент 6 курсу, групи 6151м
спеціальності
121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

Птичкін В.В.
(підпис, прізвище та ініціали)

Керівник Устенко І.В.
(підпис, прізвище та ініціали)

Рецензент Приходько С.Б
(підпис, прізвище та ініціали)

Завідувач кафедри Приходько С.Б.
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 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»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Приходько С.Б.
" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Птичкіну Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python, та розробка програмного забезпечення для її реалізації
керівник роботи
Устенко Ірина Валеріївна к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи

4.Зміст магістерської роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• 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 за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту програмного забезпечення
• Організаційно-економічний розділ Розрахунок економічної ефективності від розробки і впровадження програмного забезпечення
• Розділи з охорони праці та охорони навколишнього середовища Охорона праці в офісі фірми Охорона навколишнього середовища
• Висновки
• Список використаних джерел
• Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік графічного матеріалу

_____
_____
_____
_____
_____
_____

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання___

КАЛЕНДАРНИЙ ПЛАН

Назва етапів кваліфікаційної (магістерської) роботи (МР)	Термін виконання	Примітка
1. Підготовка вступної частини МР	18.10.2020	
2. Підготовка розділу (ів) МР з огляду літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямів досліджень	19.10.2020	
3. Підготовка розділу (ів) МР з результатів власних досліджень	22.10.2020	
4. Підготовка розділу МР з проекту програмного забезпечення	16.11.2020	
5. Підготовка організаційно-економічного розділу	18.11.2020	
6. Підготовка розділу з охорони праці	20.11.2020	
7. Підготовка розділу з охорони навколишнього середовища	23.11.2020	
8. Підготовка розділу МР – Висновки	25.11.2020	
9. Оформлення списку використаних джерел	27.11.2020	
10. Оформлення додатків	30.11.2020	
11. Підготовка презентації МР та доповіді	01.12.2020	
12. Подання МР на попередній захист	01.12.2020	
13. Подання МР рецензенту	11.12.2020	
14. Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	14.12.2020	
15. Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді	18.12.2020	
16. Подання на кафедру ПЗАС письмового відгуку та рецензії на кваліфікаційну роботу	18.12.2020	

Студент _____ Птичкін В.В
(підпис) (прізвище та ініціали)

Керівник роботи _____ Устенко І.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Птичкін Владислав Віталійович

«Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python, та розробка програмного забезпечення для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2020 р.

Обсяг роботи: 140 стор., 19 табл., 29 рис., 19 вик. джер., 5 додатків.

Актуальність теми. Ефективна математична модель для оцінювання розміру програмних проектів є важливою проблемою для розробки програмного забезпечення, тому що під час розробки програмного забезпечення необхідно планувати вільний простір, тому у якості параметру математичної моделі було обрано кількість методів, на основі якого можна отримати розмір у кілобайтах. В подальшому планується перехід до функціональних точок, як міра розміру програмного забезпечення для прогнозування трудомісткості та вартості програмних проектів.

Мета і завдання дослідження. Метою є підвищення достовірності оцінювання розміру програмних проектів створених мовою Python та розробка програмного забезпечення для її реалізації.

Завдання дослідження. Удосконалити математичну модель для оцінювання розміру програмних проектів створених мовою Python.

Об’єкт дослідження. Об’єктом дослідження є процес оцінювання розміру програмних проектів.

Предмет дослідження. Предметом дослідження є математична модель для оцінювання розміру програмних проектів написаних мовою Python.

Методи дослідження. Методами дослідження є методи для знаходження важелів математичної моделі через метод найменших квадратів, пошук викидів через видалені залишки та зворотне десяткове логарифмічне перетворення.

Наукова новизна одержаних результатів. Удосконалено математичну модель для оцінювання розміру програмних проектів створених мовою Python за рахунок приведення лінійної регресії до нелінійної регресії через нормалізуюче перетворення у вигляді десяткового логарифму, що дозволить поліпшити середню величину відносної похибки та рівень прогнозування.

Практичне значення одержаних результатів. Розроблено програмне забезпечення для оцінювання розміру програмних проектів створених мовою Python.

Ключові слова: удосконалення математичної моделі, нелінійна регресійна модель, зворотнє перетворення, оцінювання розміру програмних проектів.

ABSTRACT

Ptychkin Vladislav Vitaliovich

«Improving the mathematical model for the size estimation of Python-based software projects and developing the software for its implementation»

The qualification work is presented on the 140 pages of typewritten text, contains 19 tables, 29 figures, 19 appendices and 5 references.

Relevance of the topic of the work. An effective mathematical model for estimating the size of software projects is an important problem for software development, because free software must be planned during software development, so the number of methods based on which the size in kilobytes can be obtained was chosen as a parameter of the mathematical model. In the future, it is planned to move to functional points as a measure of the size of software for forecasting the complexity and cost of software projects.

The goal and objectives of the study. The aim is to increase the reliability of estimating the size of software projects created in Python and to develop software for its implementation.

Objectives of the study. Improve the mathematical model for estimating the size of software projects created in Python.

The object of the study. The object of research is the process of estimating the size of software projects.

The subject of the study. The subject of the study is a mathematical model for estimating the size of software projects written in Python.

The methods of the study. The research methods are methods for finding the levers of the mathematical model through the least squares method, the search for emissions through the removed residues and the inverse logarithmic transformation.

The scientific novelty of obtained results. Improved mathematical model for estimating the size of software projects created in Python by reducing linear regression to nonlinear regression through normalization transformation in the form of a decimal logarithm, which will improve the average relative error and prediction level.

The practical significance of obtained results. Software for estimating the size of software projects created in Python has been developed.

Keywords: improvement of mathematical model, nonlinear regression model, inverse transformation, estimation of size of software projects.

ЗМІСТ

ВСТУП

Розробка програмного забезпечення є важким і трудомістким процесом, адже правильне проектування архітектури і основних складових програмних проектах впливають на складність процесу розробки, розширення, а це в свою чергу безпосередньо відображається на розмірі проекту.

Розмір програмних проектів є вагомою складовою програмного забезпечення на усіх етапах розробки, адже для економії витрат та зусиль розробників необхідне правильне оцінювання розміру розроблюваного проекту.

Для оцінювання розміру програмного забезпечення існують багато методів, що можна використати при проектуванні окремих проектів, але не існує таких готових поширених інструментів для оцінювання, які мають усі необхідні метрики та реальну математичну модель, що надає необхідний прогнозований розмір програмних проектів.

Актуальність теми. Ефективна математична модель для оцінювання розміру програмних проектів є важливою проблемою для розробки програмного забезпечення, тому що під час розробки програмного забезпечення, важливим є процес проектування і планування, для виявлення необхідного простору, виділеного на розроблене програмне забезпечення. Адже, у процесі проектування можна уникнути більшої кількості витрат, які виникають на етапах розширення та підтримки вже розробленого програмного забезпечення. Тому у якості параметру математичної моделі було обрано кількість методів, на основі якого можна отримати розмір у кілобайтах. Отримавши прогнозований розмір розроблюваного програмного проекту, можна спроектувати правильну архітектуру проекту і задовольнити виділене на цей проект вільне місце. В подальшому планується перехід до кількості строк коду для прогнозування трудомісткості та вартості програмних проектів. Для достовірного

прогнозування розміру проекту необхідно удосконалення математичної моделі та існуючих методів. Якісна математична модель має надати можливість більш ефективного проектування та розробки програмного забезпечення.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є підвищення достовірності оцінювання розміру програмних проектів створених мовою Python та розробка програмного забезпечення для її реалізації.

Завдання дослідження. Проаналізувати методи дослідження математичної моделі для оцінювання розміру програмних проектів створених мовою Python; удосконалити математичну модель для оцінювання розміру програмних проектів створених мовою Python.

Об'єкт дослідження. Об'єктом дослідження є процес оцінювання розміру програмних проектів.

Предмет дослідження. Предметом дослідження є математична модель для оцінювання розміру програмних проектів написаних мовою Python.

Методи дослідження. Методами дослідження є методи для знаходження важелів математичної моделі через метод найменших квадратів, пошук викидів через видалені залишки та зворотне десяткове логарифмічне перетворення.

Наукова новизна. Наукова новизна роботи полягає в удосконаленні математичної моделі для оцінювання розміру програмних проектів створених мовою Python, за рахунок приведення лінійної регресії до нелінійної регресії через зворотне десяткове логарифмічне перетворення та знаходження викидів, що дозволить поліпшити коефіцієнт детермінації, середню величину відносної похибки та рівень прогнозування.

Практична цінність. Практична цінність – це програма для оцінювання розміру програмних проектів створених мовою Python.

Особистий внесок здобувача. Дана кваліфікаційна робота є самостійно виконаною науковою роботою.

1 АНАЛІЗ МЕТОДІВ ДОСЛІДЖЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ТА ЇЇ ПРИЗНАЧЕННЯ

1.1 Опис предметної галузі

У якості мови програмування для проектів, які досліджуються та розмір яких оцінюється математичною моделлю, було обрано Python. Python – це дуже популярна мова програмування на сьогоднішній день, яка орієнтована на підвищення ефективності розробки, швидкості написання коду і на високу якість самого ПЗ, що корисно як у маленьких, так і у великих проектах. Дана мова програмування підтримує такі парадигми програмування, як структурне, об’єктно-орієнтоване, функціональне, а також аспектно-орієнтоване програмування. В основному в Python дуже легка і ефективна ООП складова, тому що в основному сам дизайн цієї мови програмування побудований навколо об’єктно-орієнтованої моделі програмування, але реалізація є специфічною у порівнянні з іншими мовами. Простий синтаксис, набір вбудованих методів і реалізації усіх базових аспектів мови програмування, надає можливість писати ПЗ набагато меншою кількістю коду, тому у використанні ООП інколи відмовляють, адже надмірне використання класів іноді непотрібне і тільки підвищує складність коду.

Тому при проектуванні ПЗ на Python інколи можна обійтись без ООП, оскільки їх можна замінити на модулі із набором тематичних функцій (статичних методів). А також деякі реалізації на Python дозволяють обійтись без змінюваних об’єктів.

Тому перед створенням програмних проект необхідно оцінити розмір майбутнього ПЗ, адже від різних факторів розробка буде вимагати різних

підходів. Програмне забезпечення, що плануються, як довгострокове при використанні ООП скоротить час розробки у майбутньому, але при розробці невеликих проектів, тільки відніме час на планування та ускладнить розробку, що напряму впливає на розмір ПЗ.

У якості параметрів для цієї мови програмування були обрані: кількість методів – оскільки цей показник росте однаково під час використання різних парадигм програмування, та розмір – це параметр, що зростає напряму при написанні програмного коду.

1.2 Опис концепції математичної моделі та її призначення.

Математична модель в першу чергу приймає участь у дослідженні реально об’єкта, або явища з ціллю отримати інформацію про властивості, або характеристики досліджуваного об’єкту та прогнозування з позиції отримання нових знань. Кожна модель нетотожна досліджуваному об’єкту-оригіналу, отже при її моделюванні враховуються лише важливі для неї фактори. Якщо результати моделювання задовольняють шуканий результат і можуть використовуватись для прогнозування поведінки, або властивостей досліджуваного, то модель вважається адекватною.

Спочатку для побудови математичної моделі створюється рівняння регресії, а після чого знаходяться параметри за допомогою вибірки вимірюваних даних. Рівняння регресії – це математична формула, яка містить у собі незалежні змінні для знаходження залежної змінної. Змінні позначаються як: Y – залежна змінна, яку ми прогнозуємо, вона завжди одна, X – незалежні змінні, що виступають характеристиками, яких може бути декілька.

Рівняння лінійної регресії виглядає як $y = a + b * x$, де a і b коефіцієнти, що наведено у джерелі [1]. Вони приймають учать у знаходженні сили та характеру впливу незалежних змінних на залежну, та характеризують ступінь значимості окремих змінних для підвищення точності моделі. Значення a представляє собою пересічення лінії оцінювання, а b – кутовий коефіцієнт, або градієнт оцінюваної лінії [2]. В результаті аналізу коефіцієнти розраховуються шляхом використання методі найменших квадратів [3].

Поліноміальна регресія підходить для нелінійних моделей даних, визначається як залежність між незалежною змінною x та залежною змінною y, та модулюється як n-ої ступені многочлена в x. Фактично це нелінійна залежність між величиною x і умовним середнім від y [4].

Модель поліноміальної регресії можна визначити як очікуване значення y, представлене поліномом n-ої ступені. Як наведено у джерелі [4]:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \varepsilon.$$

Удосконалення лінійної моделі можна виконати шляхом переходу від лінійної до нелінійної. Зазвичай вхідні дані, що не мають нормального розподілу повинні набути такого розподілу шляхом нормалізування даних за допомогою перетворення. При використанні десяткового логарифмічного перетворення, дані набувають нормального розподілу, але для застосування такої моделі, вона повинна повернутися до початкового стану, шляхом зворотного перетворення. Тому після використання зворотного перетворення регресія набуває нелінійного характеру, така модель набуває рівняння типу, як наведено у джерелі [5]:

$$log\ (y) = \beta_0 + \beta_1\ log(x). \quad (1.1)$$

Тому для неперетворених значень, передбачене значення буде розраховуватися за допомогою зворотного десяткового логарифмічного перетворення [5]:

$$y' = 10^{b_0 + b_1 \log(x)}. \quad (1.2)$$

Удосконалення математичної моделі було обрано перехід до моделі нелінійної регресії.

1.3 Аналіз існуючих методів для дослідження лінійної регресії

Лінійна регресії – це метод апроксимації залежності між вхідними та вихідними змінними на основі лінійної моделі. Апроксимація являє собою математичний метод, в основі якого лежить заміна одних математичних об’єктів іншими, близькими до початкових у тому чи інакшому сенсі, але заміняючи більш простими моделями, залишаючи лише важливі дані.

Цей процес важливий для прогнозування поведінки, або стану досліджуваного об’єкту, адже так можна визначити параметри динаміку змін конкретного об’єкту, за допомогою аналізу його початкового стану. Для прогнозування і дослідження математичної моделі зазвичай використовують математичні методи, які мають чіткі і суворі обґрунтування.

Отже на основі залежності між однією вхідною і другою вхідною змінними, розглядається лінійна регресія, що визначається рівнянням регресії, що наведено у джерелі [1]:

$$y = a + bx, \quad (1.3)$$

на базі якої будується відповідна пряма, відома як лінія регресії.

Для визначення коефіцієнтів a і b, необхідно знайти суму квадратів відхилень точок від лінії, яка була б мінімальною. Ці коефіцієнти розраховуються за допомогою метода найменших квадратів.

Метод найменших квадратів – це математичний метод для оцінювання параметрів моделей на основі експериментальних даних. Якщо дані відомі з деякою похибкою, то замість невідомих значень параметра моделі використовується наближене. Тому параметри моделі повинні бути розраховані так, щоб мінімізувати різницю між експериментальними даними і теоретичними, які були обчислені за допомогою запропонованої моделі.

Мірою неузгодженості між фактичними значеннями і значеннями, оціненими моделлю в методі найменших квадратів, служить сума квадратів різниць між ними, як приведено у джерелі [7]:

$$SSE(a,b) = \sum_{i=1}^{N}(y - y')^2 = \sum_{i=1}^{N}(y_i - (a + bx_i))^2, \qquad (1.4)$$

де y' – оцінка, яка отримана за допомогою моделі, y – фактичне спостережуване значення. Кращою моделлю буде та модель, яка мінімізує дану суму. Параметри регресійної моделі обчислюються за допомогою даного метода таким чином, щоб сума квадратів відстаней від лінії регресії до фактичних значень даних була мінімальною [6].

Метод найменших квадратів є частиною методу повної суми квадратів (Total sum of squares – SST), що використовується для перевірки значимості лінійної регресії використовують оцінювання мінливості моделі регресії.

Метод повної суми квадратів дозволяє оцінити зміну значень y_i навколо середнього $\bar{y}$. Цей метод розділяється на суму квадратів регресії (Regression sum of squares – SSR), та суму квадратів помилок (Error sum of squares – SSE), що є одним і тим самим з методом найменших квадратів.

Сума квадратів регресії (SSR) – це сума квадратів різниці між $\hat{y}_i$ (передбаченим значенням змінної y) і $\bar{y}$ (середнім значенням змінної y).

Повна сума квадратів (SST) дорівнює сумі квадратів регресії плюс сума квадратів помилок. З джерела [7] були обрані наступні формули:

$$SST = SSR + SSE.$$

Це можна виявити, як сума різниць між спостережуваними значеннями змінної *y* і її середнім значенням, що наведено у джерелі [7]:

$$SST = \sum_{i=1}^{N}(y_i - \bar{y})^2.$$

Сума квадратів регресії дорівнює сумі квадратів різниць між передбачуваними значеннями змінної *y* і її середнім значенням, наведено у джерелі [7]:

$$SSR = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2.$$

Дисперсія і стандартне відхилення дозволяють оцінити ступінь коливань даних навколо середнього значення. Вибіркова дисперсія являється наближенням середнього арифметичного, обчисленого на основі квадратів різниць між кожним елементом вибірки і вибірковим середнім. Для вибірки *X1, X2, ... Xn* вибіркова дисперсія (позначається символом S2) задається наступною формулою, що зазначена у джерелі [8]:

$$S^2 = \frac{(X_1-\bar{X})^2+(X_2-\bar{X})^2+\cdots+(X_n-\bar{X})^2}{n-1}.$$

В загальному випадку вибіркова дисперсія – це сума квадратів різниць між елементами вибірки і вибірковим середнім, поділена на величину, що дорівнює обсягу вибірки мінус один [8]:

$$S^2 = \frac{\sum_{i=1}^{N}(X_i-\bar{X})^2}{n-1},$$

де $\bar{X}$ – арифметичне середнє, *n* – об’єм вибірки, X_i – *i*-й елемент вибірки *X*.

Найбільш практичною і широко поширеним оцінюванням розкиду даних є стандартне вибіркове відхилення. Цей показник позначається символом S і дорівнює квадратному кореню з вибіркової дисперсії, що наведено у джерелі [9]:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}.$$

Дисперсія і стандартне відхилення дозволяють оцінити розкид даних навколо середнього значення, інакше кажучи, визначити, скільки елементів вибірки менше середнього, а скільки – більше. Дисперсія має деякі цінні математичні властивості. Однак її величина є квадратом одиниці виміру. Стандартне відхилення дозволяє оцінити величину коливань елементів вибірки навколо середнього значення. Практично у всіх ситуаціях основна кількість спостережуваних величин лежить в інтервалі плюс-мінус одне стандартне відхилення від середнього значення. Отже, знаючи середнє арифметичне елементів вибірки і стандартне вибіркове відхилення, можна визначити інтервал, якому належить основна маса даних [10].

Але для застосування методів дослідження математичної моделі, необхідно нормалізувати вхідні дані і привести їх до деякої загальної шкали. Застосування нормалізації даних обумовлена тим, що в різних параметрах вхідного набору даних можуть бути представлені різними розмірами та у різних масштабах і змінюватися в різних діапазонах.

Тому може виникати порушення впливу між вхідних змінними, представленими в різних розмірах один до одного, на вихідну змінну. Тобто такий вплив обумовлено не залежністю даних, а в результаті зміни масштабу. Тому на математичній моделі можуть відобразитися некоректні залежності. Оскільки такі випадки необхідно виявити і використати нормалізацію даних, за одним з основних методів нормалізації.

Одним із методів нормалізації даних є метод десяткового масштабування. Цей метод нормалізації здійснюється за рахунок переміщення десяткового дробу на максимальне число розрядів досліджуваних чисел (тобто, якщо найбільше

число 1000, то число розрядів буде рівнятися 4), відповідне порядку числа, що наведено у джерелі [11]:

$$x_i' = x_i/10^n,$$

де n – число розрядів найбільшого значення.

Наступним методом є мінімаксна нормалізація. Недоліком минулого методу десяткового масштабування було отримані значення не займають весь діапазон від 0 до 1, а займають лише його частину, від найбільшого і від найменшого спостережуваних значень. Наприклад, якщо діапазон спостережуваних даних замалий (від 300 до 600), виходить, що при використанні десяткового масштабування нормалізовані дані будуть у діапазоні лише від 0.3 до 0.6, тобто мінливість таких даних буде дуже низькою, це означає що така математична модель не може якісною. Але використання мінімаксного методу нормалізації може вирішити дану проблему. Його реалізацію можна описати такою формулою, що наведена у джерелі [11]:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

Така нормалізація дозволяє отримати дані приведені до діапазону [0, 1].

Також існує метод нормалізації середнім (Z-нормалізація). Одним з недоліків мінімаксної нормалізації є присутність викидів, що можуть розтягнути діапазон даних, що можуть призвести до концентрації адекватних нормалізованих значень у вузькому діапазону навколо нуля. Для того, щоб уникнути таких випадків необхідно визначати діапазон не за допомогою максимальних і мінімальних значень, а за допомогою середнього та дисперсії, що наведено у джерелі [11]:

$$x_i' = (x_i - \bar{X})/\sigma_x.$$

Значення отримані по даній формулі, називаються Z-оцінками. Абсолютне значення при розраховані таких значень є оцінюванням відстані значення x та

його середнього значення у загальній сукупності. Якщо Z оцінка менше нуля, то *x* нижче середнього, а якщо більше нуля, то *x* вище середнього [11].

Для знаходження важелів математичної моделі найпоширенішим методом являється метод найменших квадратів, при якому функція двох важелів a і b, що наведено у джерелі [7]:

$$F(a,b) = \sum_{i=1}^{n}(y_i - (ax_i + b))^2,$$

що приймає найменше значення. Сума квадратів відхилення досліджуваних даних буде мінімальним від знайденої прямої.

В даній кваліфікаційній роботі для знаходження коефіцієнтів ліній регресії обрано метод найменших квадратів.

1.4. Аналіз аномальних значень та методи знаходження викидів

Зазвичай в результаті процесу аналізу досліджуваних даних, математична модель містить у собі дані, які не вписуються в цю модель, оскільки містять похибки, некоректні дані, або помилки зв’язані зі значною мінливістю остаточних даних. Такі вхідні дані часто називають аномальними значеннями.

Для виконання аналітичної обробки даних аномальні значення необхідно видалити з вибірки досліджуваних даних, оскільки вони можуть визвати некоректну роботу методів, які використовуються для перетворень та оцінювання отриманої моделі і привести до спотворення кінцевих результатів.

Одним з таких методів знаходження аномальних значень, а саме викидів, буде видалений залишок (Deleted residual). Розрахунок видалених залишків полягає в тому, що отримане рівняння регресії, повинно бути без впливу потенційного викиду знайденого викиду на нього. Для цього i-тий спостережений викид видаляється, а за рештою отриманих (n-i) спостереженнями будується нове

рівняння регресії та розраховується нове очікуване значення для *i*-того спостереження $\hat{y}_{i(i)}$, за формулою розрахунку очікуваного значення $\hat{y}_i = a + bx_i$. Різниця між спостережуваними значеннями та очікуваними за відсутності даного спостереженого значення і є видаленим залишком. За абсолютним значенням видалені залишки завжди більше звичайних залишків, що наведено у джерелі [12]:

$$r_{(i)} = y_i - \hat{y}_{i(i)},$$

де $\hat{y}_{i(i)}$ – нове очікуване значення для вилученого *i*-того спостереження.

Наступний метод з'ясування наявності викидів у досліджуваних даних, це стьюдентизований видалений залишок (Studentized deleted residual – SDR). Даний метод є відношенням залишку до його стандартної похибки, обчисленої по регресійної залежності з видаленим *i*-тим спостереженням. Виділення *i*-того спостереженого значення дозволяє викинути його вплив, як потенціального викиду на досліджувану регресійну залежність.

При розрахунку стьюдентизованого видаленого залишку залучається зовнішнє оцінювання дисперсії похибки регресії, вільна від впливу потенційного викиду. Зовнішній або віддалений стьюдентизований залишок має t-розподіл Стьюдента з числом ступенів свободи $v = n\text{-}k$. Для не надто малих вибірок і невеликого числа параметрів регресії k критичні значення t-розподілу для 5% - ного рівня значимості близькі до 2. На підставі цього спостереження з віддаленими стьюдентизованими залишками рівними 2 і більше як правило, стосуються викидів, хоча для більш певного висновку бажано розрахувати безпосередньо досягнутий рівень значущості, що наведено у джерелі [12]:

$$rt_{(i)} = \frac{y_i - \hat{y}_i}{\sqrt{S_e^2}\sqrt{1 - \frac{1}{n} + \frac{(x_i - \bar{x})^2}{(n-1)S_x^2}}}, \qquad (1.5)$$

де S_x^2 – дисперсія предикатів x_i, S_e^2 – дисперсія похибки регресії.

Дисперсія предикатів x_i розраховується за формулою, що зазначена у джерелі [12]:

$$S_x^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}.$$

Знаменник $\sqrt{S_e^2}$ – це формула стандартного відхилення похибки регресії, що в даному випадку розраховується для кожного видаленого *i*-того спостереження. Кожне *i*-те спостереження видаляється із набору даних, на момент вирішення даної формули. Далі для нових даних без поточного видаленого спостереження, ми обчислюємо дисперсію похибки регресії, що наведена у джерелі [12]:

$$S_e^2 = \frac{\sum_{i=1}^{n} r_{(i)}{}^2}{n-k} = \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_{i(i)}\right)^2}{n-k},$$

де $r_{(i)}$ – це видалений залишок (замість звичайного залишку), k – кількість параметрів у моделі.

Після отриманих даних, кожен *i*-тий предикат, який матиме значення більше 2, може вважатися викидом.

Нахил регресійної залежності може бути представлений в якості суми нахилів *n* окремих регресій, які проходять через кожне спостережуване значення і центр системи середніх значень $(\bar{x}; \bar{y})$, які мають деякий важіль h_i. Ці важелі називаються показниками впливу, оскільки ця величина визначає нахил регресії. Найвіддаленіші від центра системи середніх значень по осі *x* мають найбільше значення для загального рівняння регресії, а значення важеля h_i більше. Важіль h_i визначаються в діапазоні від 0 до 1. Великими значеннями вважаються такі значення, яке наведено у джерелі [12]:

$$h_i > \frac{2k}{n},$$

де k – кількість параметрів, n – кількість даних.

Наступним методом для знаходження викидів є відстань Махалонобіса, що є мірою відстані між векторами випадкових величин (може визначатися як

нормалізована Євклідова відстань, якщо використані матриці коваріації є одиничною матрицею).

Відстань Махалонобіса – це відстань між багатомірними векторами (матрицями) X, до множини середніх значень μ і матрицею коваріації S, що розраховується за даною формулою, що наведена у джерелі [13]:

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}.$$

Відстань Махалонобіса можна представити як відстань від порівнюваної точки до центру (середньої точки) досліджуваного простору, що визначається кореляційними незалежними змінними. Центр називають центроїдом, що є центром тяжкості серед досліджуваних точок [13].

Ще одним методом є відстань Кука. Відстань Кука – це оцінка впливу спостереження, під час розрахунку методу найменших квадратів у регресійному аналізі. Ця відстань вимірюється в результаті розрахунку різниці між очікуваними значеннями вихідної регресії та очікуваними значеннями поточної регресії, що була побудована без спостережуваного видаленого *i*-го значення, що наведено у джерелі [14].

Повна формула наведена у джерелі [12] і визначається як:

$$CD_i = \frac{\sum_{n=1}^{i}\left(\hat{y}-\hat{y}_{(i)}\right)^2}{kS_e^2}.$$

Одним з наступних критеріїв для знаходження викидів є критерій Фішера. Критерій Фішера – це статистичний критерій для оцінювання значимості відмінності (ділення) дисперсій наборів даних, що представляють із себе дисперсію усіх незалежних змінних *x* та дисперсію залежних змінних *y*.

Розраховане значення називається F-статистика, яка наведена у джерелі [15]:

$$F = \frac{S_x^2}{S_y^2}.$$

Для того, щоб дані що спостерігаються мали розподіл Фішера, необхідно щоб дисперсії обох вибірок були незалежні один від одного та суми квадратів мали розподіл Хі-квадрату. Для цього дані мають бути нормально розподілені. Якщо знайдене статистичне значення більше критичного рівня значимості (p-value) і більше ступенів вільності для чисельника та знаменника, то дисперсії значно розрізняються [15].

Отже, залишки є складовою багатьох методів для знаходження викидів, але знайдені результати при розрахунку цієї величини мають незначну залежність до досліджуваної моделі. Адже при використанні залишків, можна дізнатися лише різницю фактичного значення до значення моделі. Таку величину необхідно використовувати у стандартизованих залишках та у відстані Кука. Стьюдентизовані залишки використовують видалені залишки.

При розрахунку стандартизованих залишків вплив можна побачити лише у рамках одного спостереженого значення, а при розрахунку стьюдентизованих залишків можна знайти міру впливу спостереження на модель, завдяки порівнянню передбачення з спостереженням і без нього. Цей метод дозволяє побачити наскільки потенційний викид впливає на лінію регресії, що підвищує ефективність пошуку аномальних значень.

Тому методом для знаходження викидів було обрано формулу (1.5) – метод стьюдентизованих видалених залишків.

1.5 Методи оцінювання доцільності математичної моделі

Оцінювання значимості об’язковий процес перевірки правильності обраної регресії, що надає змогу зрозуміти доцільність обраної математичної моделі та отримати оцінку якості прогнозованого *y*.

Перевірка значимості рівняння регресії розраховується на основі дисперсійного аналізу (які були розглянуті у пункті 1.2).

Наступні формули були наведені у джерелі [7]:

Повна сума квадратів (Total sum of squares – SST):

$$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2.$$

Сума квадратів регресії (Regression sum of squares – SSR):

$$SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2.$$

Сума квадратів помилок (Error sum of squares — SSE):

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Коефіцієнт детермінації – це доля дисперсії, яка залежить від змінної *y*, що являється мірою згоди, за допомогою якої можна з'ясувати, наскільки рівняння регресії відповідає реальним даним. Даний коефіцієнт змінюється у діапазоні від 0 до 1. Якщо він дорівнює 0, це значить, що зв'язок між змінними регресійної моделі відсутній і замість неї для оцінювання значення вихідної змінної можна використовувати просте середнє її спостережуваних значень. Якщо коефіцієнт детермінації дорівнює 1, це відповідає ідеальної моделі, коли всі точки спостережень лежать точно на лінії регресії, тобто сума квадратів їх відхилень дорівнює 0.

Коефіцієнт детермінації розраховується за формулою, яка наведена у джерелі [16]:

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \quad (1.6)$$

де $\hat{y}_i$ – передбачуване значення змінної *y*.

На практиці, якщо коефіцієнт детермінації близький до 1, це вказує на те, що модель працює дуже добре (має високу значимість), а якщо до 0, то це означає низьку значимість моделі, коли вхідна змінна погано впливає на поведінку

вихідної, тобто лінійна залежність між ними відсутня. Така модель буде мати низьку ефективність.

Також існує коефіцієнт кореляції (Пірсона) – показник, який характеризує силу зв’язку між двома, або більше змінними. Якщо коефіцієнт кореляції описує зв'язок між двома випадковими величинами, то він називається простим, якщо між однією випадковою величиною і їх групою, то множинним.

Простий коефіцієнт кореляції Пірсона розраховується за формулою, що наведена у джерелі [17]:

$$r = \frac{\sum_{i=1}^{n}(x_n - \bar{x})(y_n - \bar{y})}{n\sigma_x\sigma_y},$$

де n – число спостережень, x та y – випадкові змінні.

Значення коефіцієнта кореляції завжди розташовані в діапазоні від -1 до 1. Якщо коефіцієнт кореляції близький до 1, то між змінними спостерігається позитивна кореляція. Іншими словами, відзначається високий ступінь зв'язку між змінними. В даному випадку, якщо значення змінної x зростатимуть, то і вихідна змінна також буде збільшуватися. Якщо коефіцієнт кореляції близький до -1, це означає, що між змінними має місце сильна негативна кореляція. Іншими словами, поведінка вихідної змінної буде протилежним поведінки вхідний. Якщо значення x буде зростати, то y буде зменшуватися, і навпаки. Проміжні значення, близькі до 0, будуть вказувати на слабку кореляцію між змінними і, відповідно, низьку залежність. Іншими словами, поведінка змінної x не буде зовсім (або майже зовсім) впливати на поведінку y (і навпаки).

Коефіцієнт кореляції дорівнює квадратному кореню коефіцієнта детермінації, тому може застосовуватися для оцінювання значущості регресійних моделей.

Також даний коефіцієнт дозволяє визначити вплив кореляції, в якій відсутня мультиколініарність, в якій незалежні змінні пов’язані між собою і зміна однієї

безпосередньо впливає на іншу, що в свою чергу впливає на ефективність лінійної регресії.

Метриками для перевірки ефективності отриманої математичної моделі було обрано коефіцієнт детермінації, середня величина відносної похибки та рівень прогнозування.

Коефіцієнт детермінації дозволяє оцінити наскільки великий зв’язок між змінними моделі регресії і наскільки вони відповідають лінії регресії.

Середня величина відносної похибки являється середнім відхиленням значень, що розраховуються від фактичних.

Рівень прогнозування являється відношенням кількісті значень середнього відхилення, що входять до вказаного ліміту, до кількості значень.

На таких отриманих метриках можна зробити загальний висновок щодо достовірності математичної моделі.

1.6 Висновки до розділу 1

У даному пункті було розглянуті існуючі методи для дослідження лінійної і нелінійних регресій. Були досліджені джерела, які були використані для аналізу предметної галузі та був виконаний пошук необхідних методів та критеріїв оцінювання, що дозолять удосконалити математичну модель для оцінювання розміру програмних проектів створених мовою Python. У якості метода для пошуку важелів був обраний метод найменших квадратів. Для пошуку викидів був обраний метод зовнішніх стьюдентизованих залишків. Для того, щоб перейти від лінійної регресії до нелінійної, було використано зворотне десяткове логарифмічне перетворення, в результаті якого можна отримати модель нелінійної регресії. У якості оцінювання доцільності були обрані методи

коефіцієнту детермінації, середньої величини відносної похибки та рівень прогнозування.

2 УДОСКОНАЛЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ПРОЕКТІВ, ЩО СТВОРЕНІ МОВОЮ PYTHON

2.1 Аналіз даних для побудови математичної моделі для оцінювання розміру програмних проектів створених мовою Python

Для побудови лінійної регресії необхідно визначити лінійне рівняння з незалежною та залежною змінними (критеріями) математичної моделі. Для досліджуваних програмних проектів написаних мовою Python були обрані критерії «Кількість методів» і «Розмір програми».

Після обраних критеріїв був відібраний список проектів на електронному ресурсі Github. До вхідних даних було обрано 96 проектів, з яких була отримана вибірка, яка містить у собі дані з кількістю методів та розміром файлів проекту, які стосуються лише програмного коду.

Критерієм X виступає «Кількість методів», що була набута шляхом парсингу обраних проектів, були перевірені усі файли розширення «.py» на наявність входження синтаксису, який відповідає за позначення методу. Критерієм Y виступає розмір програми, що був набутий шляхом вимірювання кожного з оцінюваних проектів, де враховувались лише файли з розширенням «.py», тому що більша кількість проектів має великий розмір за рахунок великої кількості файлів іншого розширення, зображень та інших неважливих для оцінювання файлів. Тому було обрано враховувати лише файли, які потенційно містять дані, що підпадають під критерій X. Розмір було вирішено розраховувати у кілобайтах.

Знайдені дані зображено у таблиці 2.1.

Таблиця 2.1 – Вхідна вибірка даних

	x	y		x	y		x	y
1	2	3	4	5	6	7	8	9
1	1339	487,6982	33	8962	8902,232	65	87	198,3311
2	860	1076,716	34	606	330,1846	66	9566	13854,06
3	71	113,8125	35	3628	1643,628	67	65	70,37695
4	370	254,2891	36	37311	27321,48	68	3134	3154,691
5	126	370,5205	37	529	551,1885	69	368	4575,316
6	905	610,0596	38	7050	8700,047	70	358	245,2959
7	154	254,6543	39	56820	30019,1	71	1270	598,6582
8	10768	13679,61	40	2558	2206,845	72	910	750,9688
9	194	189,4229	41	1706	3451,194	73	227	144,8203
10	12716	7005,311	42	31181	21436,23	74	42	66,12402
11	51	33,75879	43	179	252,6113	75	180	121,7148
12	607	659,0723	44	177	185,752	76	388	279,1299
13	666	374,5762	45	901	873,7441	77	5862	2274,923
14	4100	3879,222	46	4296	2821,701	78	10909	8041,81
15	8833	7622,136	47	1971	1384,298	79	34	52,99902
16	9532	8090,798	48	338	113,8193	80	1173	580,8672
17	56	18,35547	49	179	209,3115	81	156	131,1182
18	30	40,42578	50	1146	960,5742	82	214	176,6123
19	65	126,3975	51	995	600,5703	83	3416	2377,452
20	178	224,5723	52	847	477,458	84	2158	1290,646
21	785	2027,106	53	276	167,6797	85	1618	651,4082
22	67	61,03223	54	327	355,6006	86	965	450,9512
23	11398	8988,888	55	487	439,3154	87	2678	4466,801
24	74	80,7998	56	328	119,8291	88	155	54,11328
25	28	57,89844	57	2036	1048,096	89	97	81,82031
26	215	140,5322	58	879	766,8164	90	4820	2114,489
27	5168	3796,419	59	208	119,1914	91	503	183,7002
28	96	81,37598	60	3661	2094,167	92	9685	7228,156
29	1477	2139,466	61	379	217,0938	93	360	829,8779
30	538	221,1426	62	1934	1555,52	94	160	161,9121
31	1374	1683,655	63	2754	1395,973	95	1600	2387,804
32	391	265,084	64	144	79,75098	96	20691	16056,15

Розподіл за вхідними даними зображено на рисунках 2.1 та 2.2.

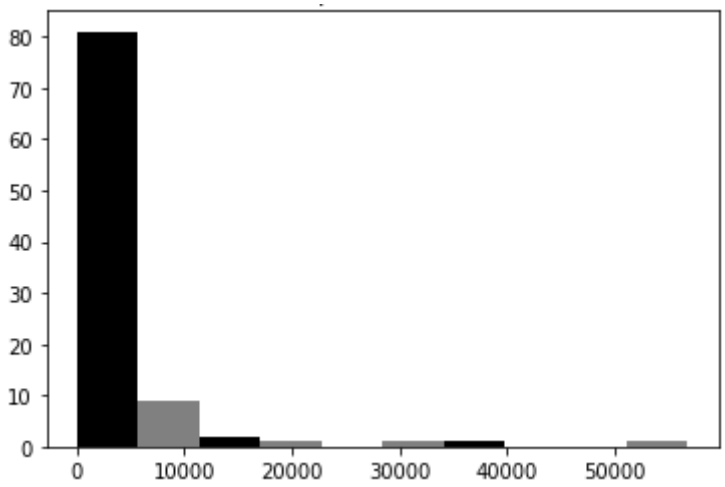

Рисунок 2.1 – Розподіл для параметру «Кількість методів»

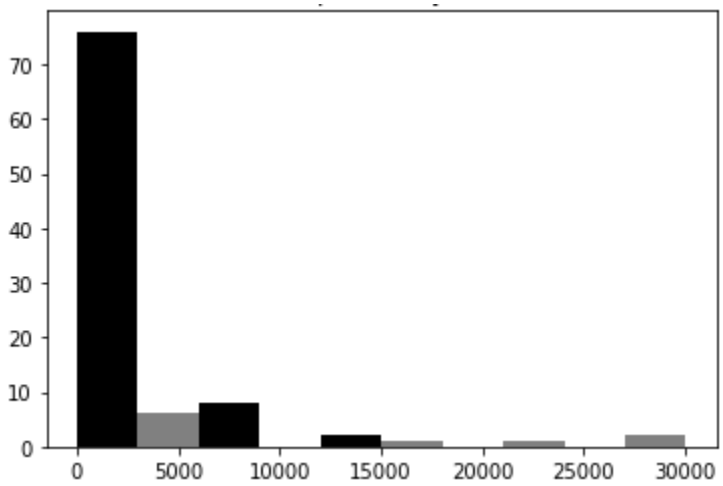

Рисунок 2.2 – Розподіл для параметру «Розмір програми»

Для використання та аналізування вхідних даних, їх необхідно перевірити на нормальність розподілу, оскільки для знаходження викидів та використання необхідних статистичних критеріїв, розподіл досліджуваних даних повинен бути нормальним.

Знаходження нормального розподілу може бути виконаний по різному, за використанням різних методів для нормалізування даних. У даному випадку було

вирішено використовувати логарифмічний розподіл, з використанням перетворення логарифму десяткового.

Знаходження нормального розподілу зображено на рисунках 2.3 та 2.4.

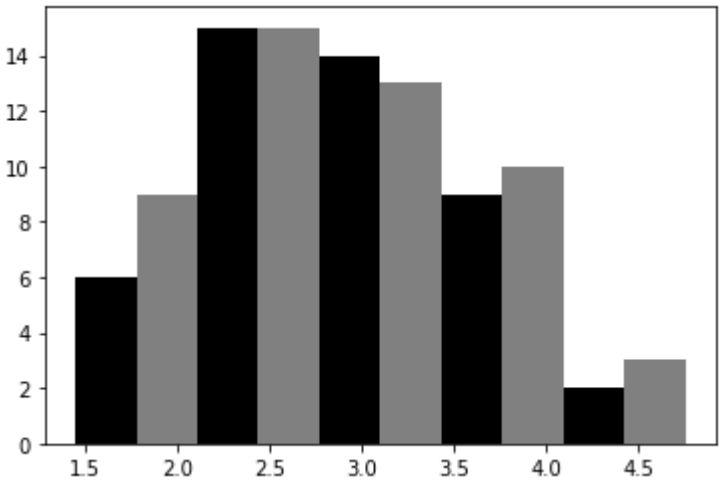

Рисунок 2.3 – Розподіл з використанням десяткового логарифмічного перетворення для параметру «Кількість методів»

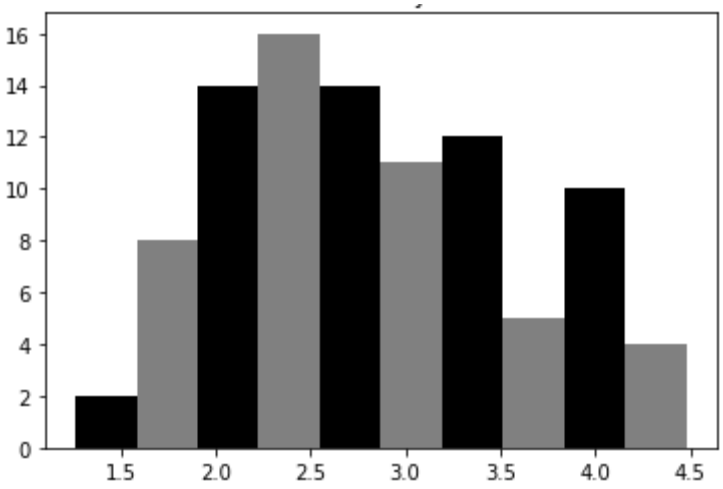

Рисунок 2.4 – Розподіл з використанням десяткового логарифмічного перетворення для параметру «Розмір програми»

Досліджувані дані перетворені до нормального розподілу приведено у таблиці 2.2.

Таблиця 2.2 – Нормальний розподіл даних

	x	y		x	y		x	y
1	2	3	4	5	6	7	8	9
1	3,126781	2,688151	33	3,952405	3,949499	65	1,939519	2,297391
2	2,934498	3,032101	34	2,782473	2,518757	66	3,98073	4,141577
3	1,851258	2,05619	35	3,559667	3,215804	67	1,812913	1,84743
4	2,568202	2,405328	36	4,571837	4,436504	68	3,496099	3,498957
5	2,100371	2,568812	37	2,723456	2,7413	69	2,565848	3,660421
6	2,956649	2,785372	38	3,848189	3,939522	70	2,553883	2,38969
7	2,187521	2,405951	39	4,754501	4,477398	71	3,103804	2,777179
8	4,032135	4,136074	40	3,407901	3,343772	72	2,959041	2,875622
9	2,287802	2,277432	41	3,231979	3,537969	73	2,356026	2,160829
10	4,104351	3,845427	42	4,49389	4,331148	74	1,623249	1,820359
11	1,70757	1,528387	43	2,252853	2,402453	75	2,255273	2,085344
12	2,783189	2,818933	44	2,247973	2,268933	76	2,588832	2,445806
13	2,823474	2,57354	45	2,954725	2,941384	77	3,768046	3,356967
14	3,612784	3,588745	46	3,633064	3,450511	78	4,037785	3,905354
15	3,946108	3,882077	47	3,294687	3,14123	79	1,531479	1,724268
16	3,979184	3,907991	48	2,528917	2,056216	80	3,069298	2,764077
17	1,748188	1,263765	49	2,252853	2,320793	81	2,193125	2,117663
18	1,477121	1,606658	50	3,059185	2,982531	82	2,330414	2,247021
19	1,812913	2,101738	51	2,997823	2,778564	83	3,533518	3,376112
20	2,25042	2,351356	52	2,927883	2,678935	84	3,334051	3,110807
21	2,89487	3,306877	53	2,440909	2,22448	85	3,208979	2,813853
22	1,826075	1,785559	54	2,514548	2,550962	86	2,984527	2,65413
23	4,056829	3,953706	55	2,687529	2,642776	87	3,427811	3,649997
24	1,869232	1,90741	56	2,515874	2,078562	88	2,190332	1,733304
25	1,447158	1,762667	57	3,308778	3,020401	89	1,986772	1,912861
26	2,332438	2,147776	58	2,943989	2,884691	90	3,683047	3,325205
27	3,713323	3,579374	59	2,318063	2,076245	91	2,701568	2,26411
28	1,982271	1,910496	60	3,5636	3,321011	92	3,9861	3,859028
29	3,16938	3,330305	61	2,578639	2,336647	93	2,556303	2,919014
30	2,730782	2,344672	62	3,286456	3,191875	94	2,20412	2,209279
31	3,137987	3,226253	63	3,439964	3,144877	95	3,20412	3,377999
32	2,592177	2,423383	64	2,158362	1,901736	96	4,315781	4,205641

Таким чином було отримано нормальний розподіл даних.

2.2 Побудова лінійної та нелінійної моделі регресії та знаходження довірчого інтервалу та інтервалу передбачення

На рисунках 2.3 та 2.4 зображені гістограми з розподілом даних, що отримано за допомогою десяткового логарифмічного перетворення. Аналізуючи отримане перетворення можна побачити, що дані прийняли нормальний розподіл.

Після отриманих нормалізованих даних, можна виконати знаходження викидів. Для таких даних можна розрахувати рівняння лінійної регресії. Знаходження коефіцієнтів було виконано за наведеної формули (1.4). Розрахування регресії можна виконати за наведеної формули (1.3) і виглядає як:

$$y = 0.63901x + 489.98019$$

Побудові графік лінійної моделі зображений на рисунку 2.5.

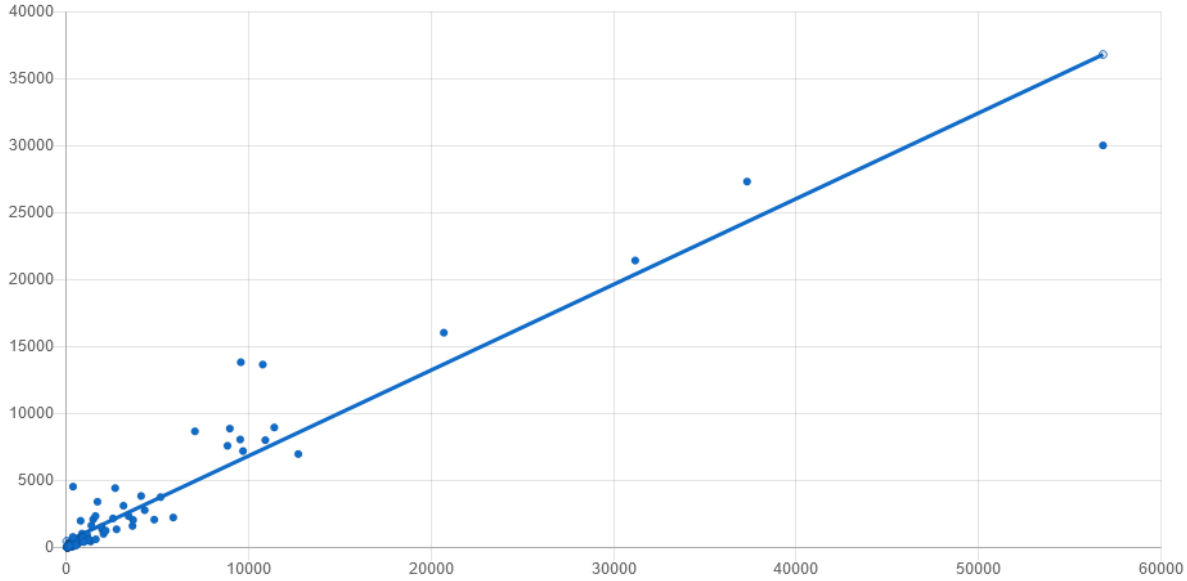

Рисунок 2.5 – Графік зображення лінійної регресії

Для дослідження математичної моделі необхідно розрахувати метрики отриманої регресії, а саме коефіцієнт детермінації (R^2), середня величина відносної похибки (*MMRE*), рівень прогнозування (*PRED*).

При розрахуванні коефіцієнту детермінації необхідно застосовувати формулу (1.6).

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

При розрахуванні середньої величини відносної похибки необхідно використовувати формулу:

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

При розрахуванні рівня прогнозування необхідно застосовувати формулу:

$$PRED(l) = \frac{k}{n},$$

де *k* – це кількість досліджуваних значень $\left| \frac{y_i - \hat{y}_i}{y_i} \right| \leq l \ (i = \overline{1,n})$.

При аналізі лінійної регресії були розраховані метрики:

$$R^2 = 0.914,$$

$$MMRE = 2.352,$$

$$PRED = 0.343.$$

Після нормалізації даних вхідні дані можуть бути перевірені на присутність аномальних даних. Такі дані називаються викидами і можуть бути знайдені різними способами. Один зі способів є методом стьюдентизованого видаленого залишку, що розраховується за такими формулами, що наведена у джерелі [12]:

$$rt_{(i)} = \frac{y_i - \hat{y}_i}{\sqrt{S_e^2}\sqrt{1 - \frac{1}{n} + \frac{(x_i - \bar{x})^2}{(n-1)S_x^2}}}.$$

Після знайдених значень для кожної точки, кожне значення порівнюється з критичним значенням t-розподілу для 5%-ного рівня значимості, що дорівнює приблизно 2, якщо отриманий результат дорівнює 2, або більше – можна вважати таку точку викидом.

Знайдені значення зовнішніх стьюдентизованих залишків за формулою (1.5) більших за 2 наведені у таблиці 2.3.

Таблиця 2.3 – Знайдені стьюдентизовані залишки

	x	y	rt
1	2	3	4
1	4,836282	5,914909	2,065039
2	4,025352	2,909928	2,13389
3	6,665684	7,614365	2,059865
4	5,908083	8,428431	5,466095
5	7,441907	8,146476	2,223162
6	5,823046	4,734612	2,164029
7	4,465908	5,289938	2,032206
8	5,043425	3,99108	2,216465
9	5,886104	6,721279	2,279948
10	5,793014	4,786067	2,205704
11	7,892826	8,404428	2,108201
12	6,22059	5,213305	2,121257
13	7,199678	6,189697	2,105955
14	9,16597	9,536334	2,160162
15	6,287859	5,398808	2,060002

З використанням метода стьюдентизованого видаленого залишку була виконана перевірка даних на викиди, що зображено на рисунку 2.6.

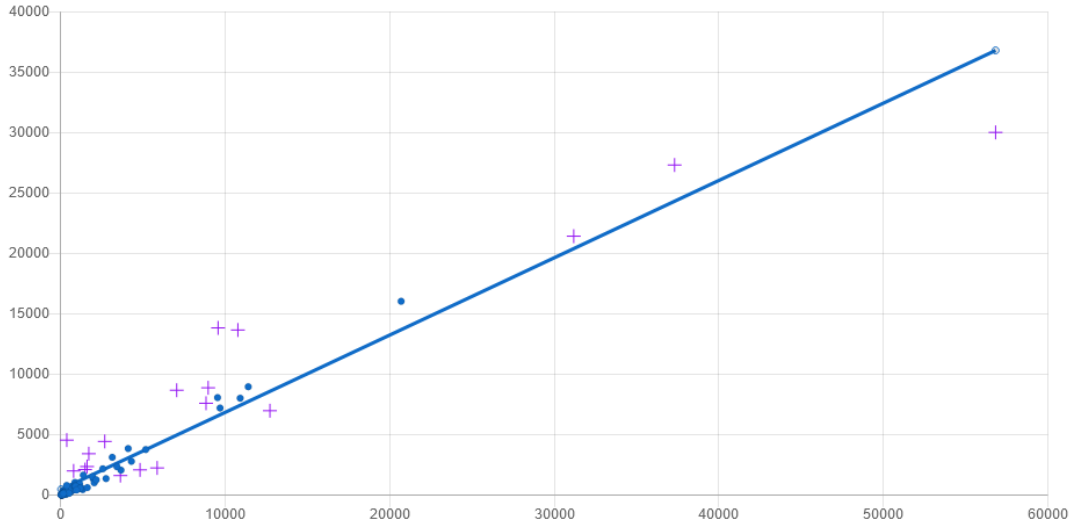

Рисунок 2.6 – Знайдені викиди на графіку лінійної регресії

На отриманих даних, які мають нормальний розподіл і які були перевірені на наявність викидів можна знайти довірчий інтервал та інтервал передбачення.

Довірчий інтервал розраховується за такою формулою, наведеною у джерелі [18]:

$$\hat{y}_h \pm t_{\left(\frac{a}{2}, n-2\right)} \times \sqrt{MSE \times \left(\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right)},$$

де *MSE* – середньоквадратична помилка.

Інтервал передбачення розраховується за такою формулою, наведеною у джерелі [18]:

$$\hat{y}_h \pm t_{\left(\frac{a}{2}, n-2\right)} \times \sqrt{MSE \times \left(1 + \frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right)}.$$

Середньоквадратична помилка за формулою, що вказана у джерелі [19]:

$$MSE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}.$$

Довірчий інтервал – це інтервал, у якому завдяки обраному значенню кількості методів (X), можна знайти вибірку проектів у яких середній розмір програми (Y) входить в знайдений довірчий інтервал.

Інтервал передбачення – інтервал, у який повинен входити розмір конкретної програми (Y) по обраному значенню кількості методів (X).

Довірчий інтервал приведено у таблиці 2.4.

Таблиця 2.4 – Значення довірчого інтервалу

	r-	r+		r-	r+		r-	r+
1	2	3	4	5	6	7	8	9
1	914,0747	1028,581	27	20808,79	27576,12	53	103,2292	123,782
2	596,865	670,9712	28	372,0872	421,8084	54	46,85774	58,85437
3	262,1206	300,5929	29	4392,993	5261,135	55	2050,681	2361,793
4	627,0402	704,6681	30	30783,72	41933,28	56	253,7637	291,3738
5	138,6686	163,6824	31	1692,587	1934,501	57	868,8878	977,1838
6	7626,417	9447,556	32	17604,86	23061,49	58	630,3785	708,4
7	36,8122	46,93623	33	128,0535	151,7779	59	161,963	189,6987
8	425,5326	480,803	34	126,6382	150,1881	60	128,7639	152,5759
9	465,758	525,2906	35	624,3532	701,6647	61	274,6449	314,4043
10	2641,363	3077,74	36	2759,977	3222,957	62	3695,337	4382,339
11	5425,22	6579,345	37	1321,848	1498,645	63	6608,93	8113,138
12	5825,758	7095,901	38	128,0535	151,7779	64	805,0789	904,8603
13	127,3469	150,9843	39	787,2575	884,7146	65	111,7517	133,4215
14	48,29182	60,54334	40	687,1305	771,9706	66	152,7961	179,4765
15	6885,275	8474,2	41	588,1265	661,225	67	2224,305	2570,902
16	53,30578	66,42716	42	196,4281	227,9903	68	1440,71	1637,584
17	153,5013	180,2635	43	232,1358	267,4964	69	1095,394	1236,259
18	3283,242	3868,629	44	343,1731	389,9301	70	667,1414	749,5526
19	69,03809	84,70932	45	232,8354	268,2693	71	69,75261	85,53415
20	1004,034	1131,335	46	1363,263	1546,964	72	3075,205	3610,987
21	378,269	428,6267	47	609,6194	685,2061	73	354,2054	402,0914
22	936,9425	1054,647	48	148,5618	174,7482	74	5913,076	7208,859
23	276,7272	316,7	49	2374,301	2752,48	75	114,5879	136,6222
24	5499,312	6674,698	50	268,3832	307,4997	76	1083,745	1222,849
25	424,8495	480,0482	51	1298,254	1471,161	77	12012,23	15331,2
26	2354,138	2728,024	52	1815,032	2079,964			

Інтервал передбачення приведено у таблиці 2.5.

Таблиця 2.5 – Значення інтервалу передбачення

	r-	r+		r-	r+		r-	r+
1	2	3	4	5	6	7	8	9
1	579,856	1621,436	27	14102,72	40689	53	67,2882	189,8983
2	378,4641	1058,17	28	236,81	662,7657	54	31,11788	88,62373
3	167,6645	469,9362	29	2862,04	8075,405	55	1314,151	3685,486
4	397,5416	1111,469	30	21071,4	61261,36	56	162,407	455,2764
5	89,799	252,7601	31	1081,078	3028,748	57	551,0539	1540,799
6	5037,143	14303,94	32	11880,34	34173,64	58	399,6531	1117,369
7	24,5894	70,26712	33	83,06844	233,972	59	104,5408	293,8964
8	270,4309	756,5604	34	82,17036	231,4647	60	83,5192	235,2305
9	295,7558	827,231	35	395,8421	1106,721	61	175,5421	491,9021
10	1701,158	4778,761	36	1779,198	4999,606	62	2397,711	6754,034
11	3552,763	10046,94	37	841,3121	2354,631	63	4349,31	12328,2
12	3821,81	10816,6	38	83,06844	233,972	64	510,4456	1427,153
13	82,62009	232,7203	39	499,1174	1395,46	65	72,71296	205,054
14	32,04699	91,23317	40	435,5818	1217,784	66	98,74374	277,7221
15	4535,859	12863,54	41	372,9424	1042,745	67	1427,605	4005,642
16	35,29043	100,3375	42	126,2985	354,5861	68	917,9918	2570,048
17	99,18987	278,9669	43	148,7953	417,3215	69	695,7859	1946,276
18	2124,579	5978,43	44	218,6293	612,0567	70	422,9203	1182,392
19	45,42464	128,7444	45	149,2358	418,5497	71	45,88359	130,0297
20	637,3035	1782,353	46	868,0079	2429,609	72	1987,059	5588,423
21	240,6976	673,6097	47	386,5257	1080,691	73	225,5658	631,4031
22	594,4458	1662,294	48	96,06427	270,2453	74	3880,532	10984,71
23	176,8517	495,5537	49	1525,829	4283,059	75	74,51641	210,0913
24	3602,492	10189,13	50	171,6038	480,9204	76	688,3211	1925,346
25	270,0011	755,361	51	826,1147	2311,957	77	8026,89	22943,13
26	1512,615	4245,724	52	1160,624	3252,734			

Довірчий інтервал та інтервал передбачення зображено на рисунку 2.7.

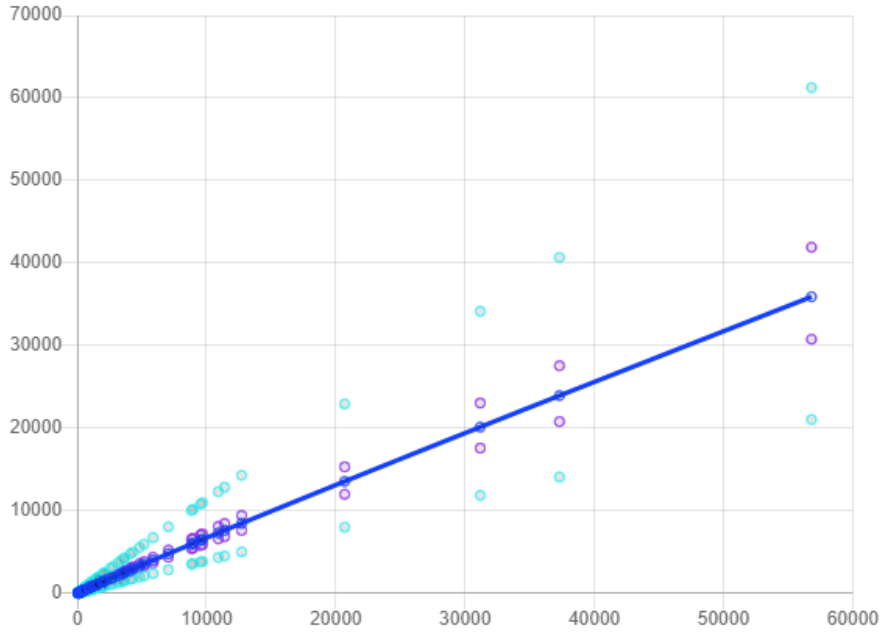

Рисунок 2.7 – Довірчий інтервал та інтервал передбачення

Після знайдених викидів необхідно перерахувати рівняння регресії для знаходження нової лінійної моделі, а також перерахувати метрики для нової лінійної регресії.

Нове рівняння лінійної регресії за формулою (1.3) після позбавлення даних від викидів:

$$y = 0.77237x + -20.92503.$$

Були перераховані метрики:

$$R^2 = 0.987,$$

$$MMRE = 0.383,$$

$$PRED = 0.653.$$

Для поліпшення математичної моделі, треба застосувати іншу, нелінійну регресійну модель. Це можна досягти завдяки зворотного перетворення, що вказано у за формулою (1.2) до нелінійної регресії, що вказана у формулі (1.1), що виявляється у новому рівнянні регресії:

$$y = 10^{0.91511\log(x)+ 0.16007}.$$

Така модель надає можливість перейти від перетворення вхідних даних для нормалізації, одразу до використання даної моделі на не нормалізованих даних.

Отримана нелінійна модель, що зображена на рисунку 2.8.

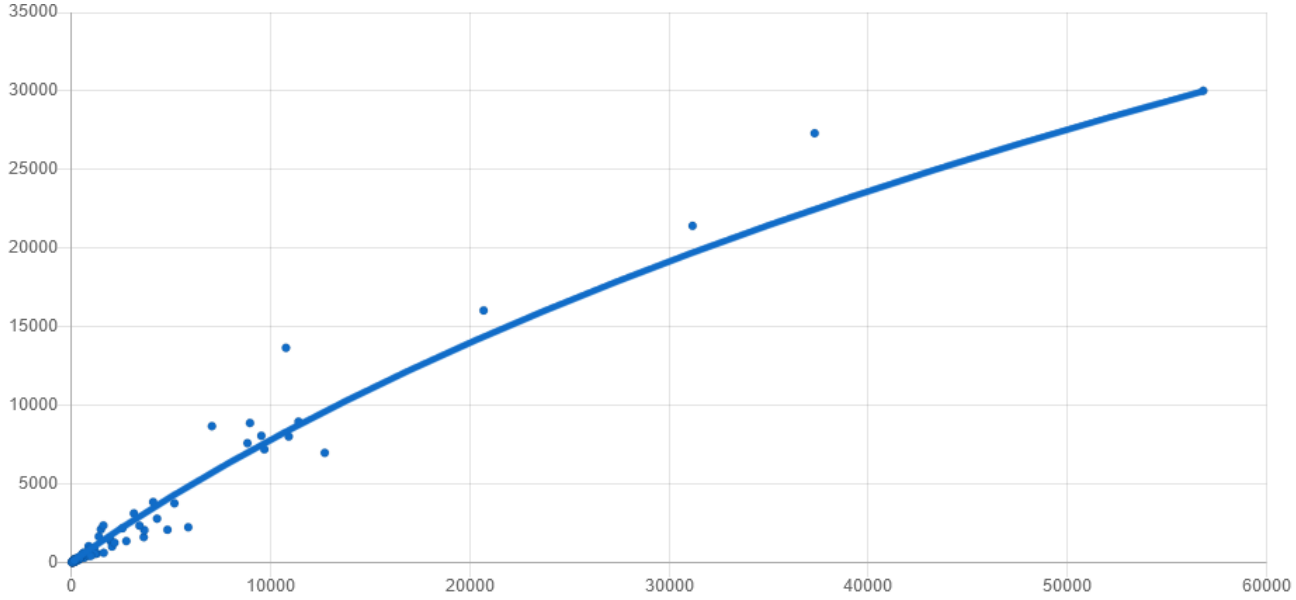

Рисунок 2.8 – Зображення нелінійної регресії на графіку

Для отриманої нелінійної моделі, були розраховані метрики:

$$R^2 = 0.959,$$

$$MMRE = 0.300,$$

$$PRED = 0.691.$$

Таким чином була побудована лінійна та нелінійна модель регресії.

2.3 Результати удосконалення математичної моделі

Виходячи з отриманих результатів, у яких були розраховані метрики для оцінювання якості моделі отримана математична модель була удосконалена, оскільки:

При застосуванні моделі нелінійної регресії, розраховані метрики мають два параметра, які краще за отримані результати при використанні лінійної моделі.

При порівнянні метрики MMRE, нелінійна модель краще на 0.083.

При порівнянні метрики PRED, нелінійна модель краще на 0.038.

За отриманою більшістю покращених показників можна сказати, що отримана математична модель була удосконалена.

2.4 Постановка задачі на розробку програмного забезпечення

Виходячи з аналізу предметної галузі та огляду літератури сформулюємо наступну постановку задачі: удосконалити математичну модель і на її основі розробити програмне забезпечення для покращення оцінювання програмних проектів створених мовою Python. Розробка надає можливість ефективного проектування програмних проектів, для економії ресурсів та часу, при проектуванні та розробці програмного забезпечення.

2.4.1 Вимоги до функціональних характеристик:

Розроблювана система повинна виконувати ряд функцій для користувача, що повинні надавати можливість завантажувати дані, розраховувати рівняння регресії, перетворювати дані та виводити результати у виді тексту та графіку:

До вимог функцій розроблюваного програмного забезпечення можна віднести:

- занесення вхідних даних у програму;
- обчислення розподілу вхідних даних;
- перетворення вхідних даних;
- обчислення лінійної регресії;
- обчислення нелінійної регресії;

- розрахунок метрик рівнянь регресії;
- вивід обчислень регресії на екран;
- вивід даних регресії на графік;
- розрахунок викидів;
- розрахунок інтервалів для введеної кількості методів.

2.4.2 Вимоги до організації вхідних та вихідних даних

Наведемо структури даних, які використовуються в програмі:

Вхідним файлом до програмного забезпечення є файл з даними, необхідними для дослідження, які мають JSON об’єкт з значеннями у вигляді ключ-значення, що мають інформацію по необхідним критеріям.

Вхідні дані мають сувору структуру JSON об’єкту, що має бути оформлений у правильному вигляді без помилок, та зберігається у файлах з типом даних «json».

Вихідних даних не передбачено.

2.4.3 Вимоги до складу та параметрів технічних засобів

Дане програмне забезпечення вимагає технічний засіб – персональний комп’ютер – для будь-якого користувача, що має змогу використати математичну модель для оцінювання розміру програмного забезпечення. Вимоги до системи на якій буде відбуватися адміністрування програмного забезпечення надані у таблиці 2.6.

Таблиця 2.6 – Системні вимоги для персонального комп’ютера

Параметр	Мінімальні	Рекомендовані
1	2	3
Операційна система	Windows 7	Windows 7, Windows 10
Процесор	32-разрядний (x86), 1ГГц, 1 ядро	64-разрядний (x64), 2ГГц+, 2 ядра +
Оперативна пам’ять	2 ГБ	4 ГБ
Вільне місце для ОС	16 ГБ	20 гб
Браузер	Firefox 54.0, Chrome 72.0	Firefox 65.0, Chrome 75.0
Підключення до інтернету	Не обов'язково	Не обов'язково

Таким чином було наведені вимоги до складу та параметрів технічних засобів.

2.4.4 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення використовується для оцінювання розміру програмного забезпечення та має інтерфейс веб-додатку. Для використання розроблюваного ПЗ, персональний комп’ютер користувача має відповідати системним вимогам наведеним у таблиці 2.6.

2.5 Висновки до розділу 2

У даному розділі було виконано удосконалення математичної моделі. Вхідні дані були нормалізовані, шляхом десяткового логарифмічного перетворення. Після нормалізації було виконана перевірка на викиди через зовнішні

стьюдентизовані видалені залишки і знайдені довірчий інтервал та інтервал передбачення. Після знайдених інтервалів було знайдено рівняння моделі нелінійної регресії, через зворотне перетворення. Після знайденої нелінійної регресії, нова математична модель була порівняна з початковою лінійною моделлю. Результатами порівняння отриманих моделей, було поліпшення двох критеріїв оцінювання MMRE на 0.083 та PRED на 0.038. З цього можна сказати, що модель для оцінювання програмних проектів створених мовою Python була удосконалена.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ПРОЕКТІВ, ЩО СТВОРЕНІ МОВОЮ PYTHON

Відповідно етапам розробки програмного забезпечення, даний розділ кваліфікаційної роботи складається з таких підрозділів: ескізного, технічного і робочого проектів.

3.1 Ескізний проект програмного забезпечення

В даному підпункті було розроблено ескізний проект для програмного забезпечення «Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python», мовою моделювання UML. Також була розроблена діаграма варіантів використання, для уточнення функціональності програмного забезпечення та аналізу вимог. Для розробленої діаграми варіантів використання була розроблена специфікація, для кожного з варіантів, а також сценарії для основних прецедентів у вигляді діаграм діяльності.

На основі приведених діаграми використання і діаграм діяльності, була розроблена інформаційна модель програмного забезпечення у вигляді діаграми класів.

3.1.1 Побудова моделі варіантів використання

Для визначення усіх потрібних функцій для системи було побудовано діаграму варіантів використання. На цій діаграмі зображено відношення акторів до їх прецедентів у системі, що відображає функціональність системи, системні взаємовідносини між діючими особами (акторами) та зв’язки між ними. Це дозволяє отримати повну картину про систему, яка розробляється, про її функціональну поведінку та дозволяє оцінити взаємодію системи із зовнішнім світом.

Проаналізувавши функціональні вимоги до програмного продукту, що були описані у постановці задачі на розробку ПЗ, були виділені такі варіанти використання:

- занесення вхідних даних у програму;
- обчислення розподілу вхідних даних;
- перетворення вхідних даних;
- обчислення лінійної регресії;
- обчислення нелінійної регресії;
- розрахунок метрик рівнянь регресії;
- вивід обчислень регресії на екран;
- вивід даних регресії на графік;
- розрахунок викидів;
- розрахунок інтервалів для введеної кількості методів.

На рисунку 3.1 наведено діаграму варіантів використання.

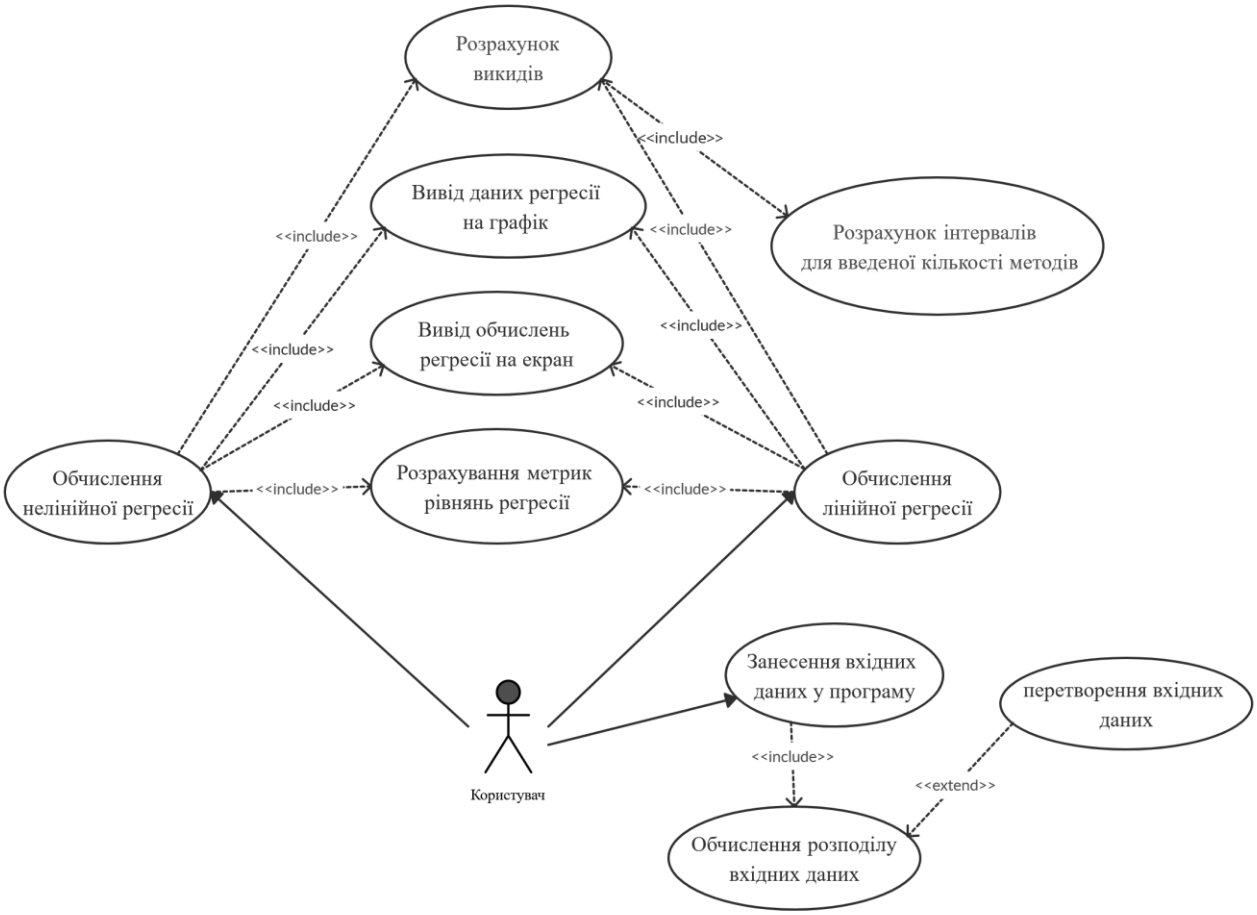

Рисунок 3.1 – Діаграма варіантів використання

3.1.2 Специфікації варіантів використання

Проаналізувавши діаграму варіантів використання з’ясуємо специфікації для кожного з них.

Специфікацію варіантів використання було наведено у лістингу таблиць від 3.1 до 3.10.

Таблиця 3.1 – Специфікація варіанту використання функції «Занесення вхідних даних у програму»

Занесення вхідних даних у програму	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Даний варіант використання заносить дані у програму
Передумови	Прецедент ініціює користувач. Користувач обирає функцію «Занести дані» на інтерфейсі користувача. Користувач обирає заздалегідь правильно відформатовані дані типу JSON
Основний потік подій	Користувач обирає файл з каталогу системи
Післяумови	Система надає можливість зробити перетворення даних, у випадку ненормального розподілу даних. Також відкриває можливість обчислити розподіл вхідних даних.

Таблиця 3.2 – Специфікація варіанту використання функції «Обчислення розподілу вхідних даних»

Обчислення розподілу вхідних даних	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Даний варіант використання надає змогу обчислити розподіл вхідних даних
Передумови	Користувач завантажує дані з файлу.
Основний потік подій	1) Користувач натискає кнопку розрахунку розподілу; 2) Користувач отримує дані на екран.

Таблиця 3.3 – Специфікація варіанту використання системи «Перетворення вхідних даних»

Перетворення вхідних даних	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Даний варіант використання дозволяє перетворити дані у випадку ненормального розподілу.
Передумови	Прецедент ініціює користувач, але після виконання прецеденту «Обчислення розподілу вхідних даних».
Основний потік подій	Користувач перетворює дані для нормального розподілу.

Таблиця 3.4 – Специфікація варіанту використання системи «Обчислення лінійної регресії»

Обчислення лінійної регресії	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Даний варіант використання надає можливість обчислити лінійну регресію.
Передумови	Користувач завантажує дані та обчислює розподіл регресії
Основний потік подій	Користувач обирає функцію обчислення лінійної регресії;
Післяумови	Система надає змогу виконати варіанти використання «Вивід обчислень регресії на екран», «Вивід даних регресії на графік» та «Розрахунок викидів».

Таблиця 3.5 – Специфікація варіанту використання системи «Обчислення нелінійної регресії»

Обчислення нелінійної регресії	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Даний варіант використання надає можливість обчислити нелінійну регресію.
Передумови	Користувач завантажує дані та обчислює розподіл регресії
Основний потік подій	Користувач обирає функцію обчислення нелінійної регресії;
Післяумови	Система надає змогу виконати варіанти використання «Вивід обчислень регресії на екран», «Вивід даних регресії на графік» та «Розрахунок викидів».

Таблиця 3.6 – Специфікація варіанту використання системи «Розрахунок метрик рівнянь регресії»

Розрахунок метрик рівнянь регресії	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Користувач розраховує і виводить обчислення на екран
Передумови	Прецедент ініціює користувач. Для виводу обчислень регресії на екран, користувачу необхідно виконати один з варіантів «Обчислення лінійної регресії», або «Обчислення нелінійної регресії»
Основний потік подій	Даний варіант використання розраховує метрики рівняння регресії і виводить їх на екран

Таблиця 3.7 – Специфікація варіанту використання системи «Вивід обчислень регресії на екран»

Вивід обчислень регресії на екран	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Користувач виводить обчислення на екран
Передумови	Прецедент ініціює користувач. Для виводу обчислень регресії на екран, користувачу необхідно виконати один з варіантів «Обчислення лінійної регресії», або «Обчислення нелінійної регресії»
Основний потік подій	Даний варіант використання виводить обчислення регресії на екран

Таблиця 3.8 – Специфікація варіанту використання системи «Вивід даних регресії на графік»

Вивід даних регресії на графік	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Користувач виводить обчислення на графік
Передумови	Прецедент ініціює користувач. Для виводу обчислень регресії на екран, користувачу необхідно виконати один з варіантів «Обчислення лінійної регресії» або «Обчислення нелінійної регресії»
Основний потік подій	Даний варіант використання виводить обчислення регресії на графік регресії

Таблиця 3.9 – Специфікація варіанту використання системи «Розрахунок викидів»

Розрахунок викидів	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Користувач знаходить викиди
Передумови	Прецедент ініціює користувач. Для виводу обчислень регресії на екран, користувачу необхідно виконати один з варіантів «Обчислення лінійної регресії», або «Обчислення нелінійної регресії».
Основний потік подій	Даний варіант використання обчислює викиди та відображає їх на графік.

Таблиця 3.10 – Специфікація варіанту використання системи «Розрахунок інтервалів та вибіркове середнє для введеної кількості методів».

Розрахунок інтервалів для введеної кількості методів	
Складова	Опис
1	2
Актори	Користувач
Інші учасники прецеденту	Система
Короткий опис	Користувач розраховує інтервали для введеної кількості методів
Передумови	Прецедент ініціює користувач. Для виводу інтервалів на екран, користувачу необхідно виконати один з варіантів «Обчислення нелінійної регресії»
Основний потік подій	Даний варіант використання розраховує інтервали для введеної кількості методів і виводить користувачеві на екран

3.1.3 Сценарії варіантів використання

На цьому етапі для аналізу процесу виконання варіантів використання створюється діаграма діяльності. Діаграми діяльності були згруповані до основних етапів виконання програми.

Діаграма діяльності для варіанту використання «Занесення вхідних даних у програму» зображена на рисунку 3.2:

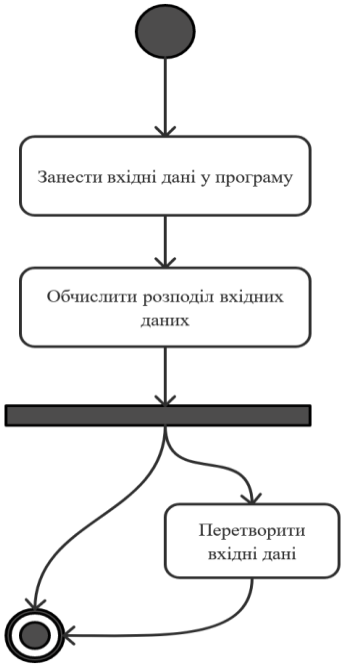

Рисунок 3.2 – Сценарій варіанту використання «Занесення вхідних даних у програму»

Діаграма діяльності для варіанту використання «Обчислення лінійної регресії» зображена на рисунку 3.3:

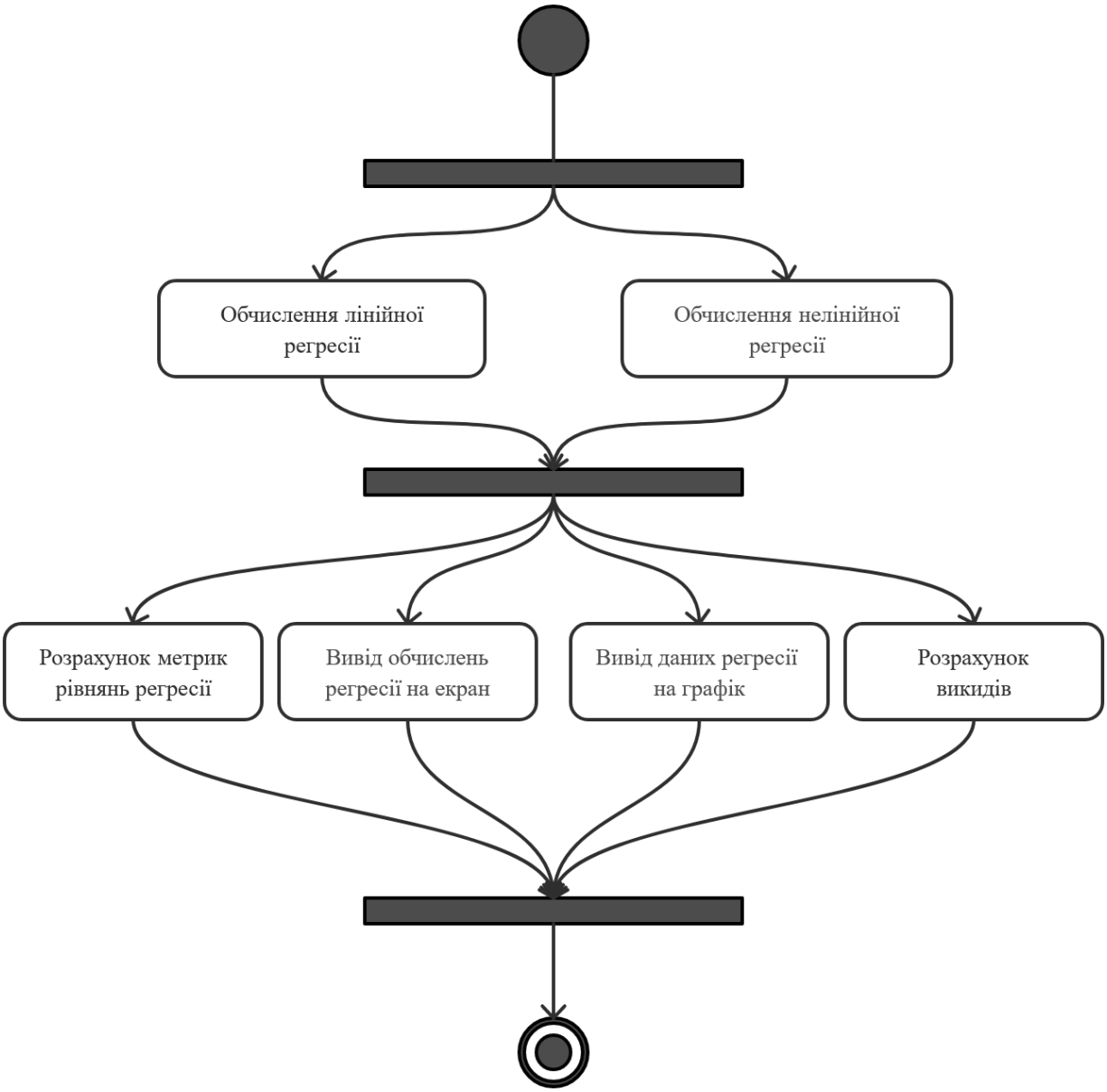

Рисунок 3.3 – Сценарій варіантів використання «Обчислення лінійної регресії» та «Обчислення нелінійної регресії»

Таким чином було розроблено специфікації варіантів використання.

3.1.4 Інформаційна модель

На основі розроблених діаграми варіантів використання та діаграми діяльності можна розробити інформаційну модель, у вигляді діаграми класів. На рисунку 3.4 наведено структуру діаграму:

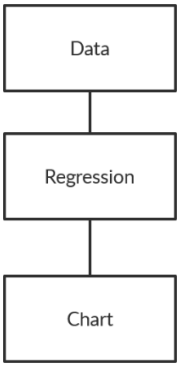

Рисунок 3.4 – Інформаційна модель

3.1.5 Розробка інтерфейсу програмного забезпечення

Інтерфейс користувача – це сукупність засобів, що забезпечує взаємодію програми з користувачем. На основі попередніх підпунктів були розроблені схема інтерфейсу, з представлених усіх функцій програмного забезпечення «Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python».

При завантажені інтерфейсу програми, користувач бачить панель керування програмою. Для використання основних функцій програми, користувачу необхідно завантажити дані за допомогою кнопки «Завантажити». Після чого дані будуть завантажені, а панель з кнопками для обчислень – відкриті для

використання. Також до розрахування розподілу вхідних даних, кнопка перетворення даних недоступна.

Дану візуальну схему з основними функціями інтерфейсу було зображено на рисунку 3.5:

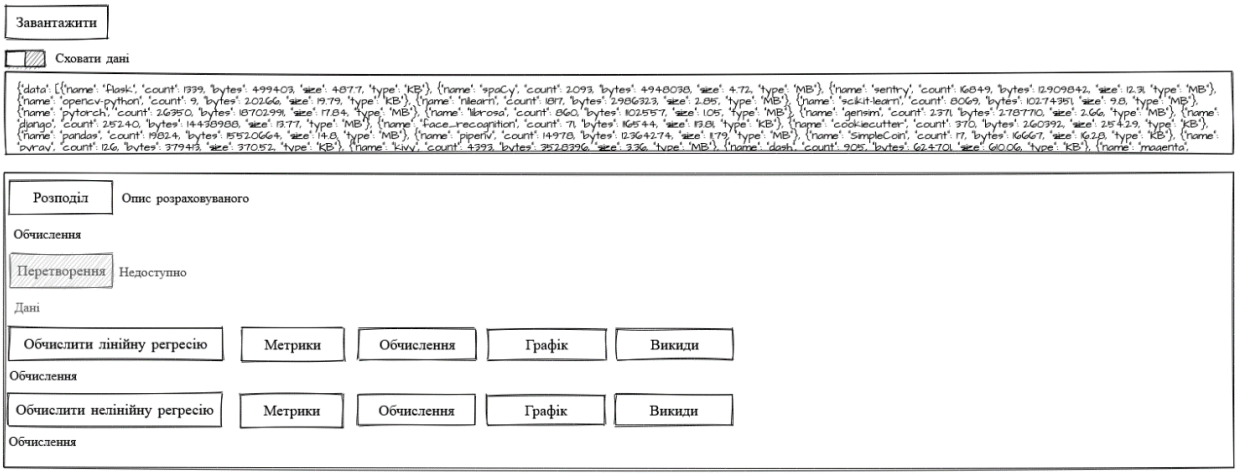

Рисунок 3.5 – Інтерфейс програмного забезпечення «Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python»

Після завантаження даних їх можна сховати. А після розрахунку розподілу, перетворення даних, можна обчислити регресію, обчислити її метрики, вивести дані обчислень, вивести розраховані дані на графік та знайти викиди, що зображено на рисунку 3.6:

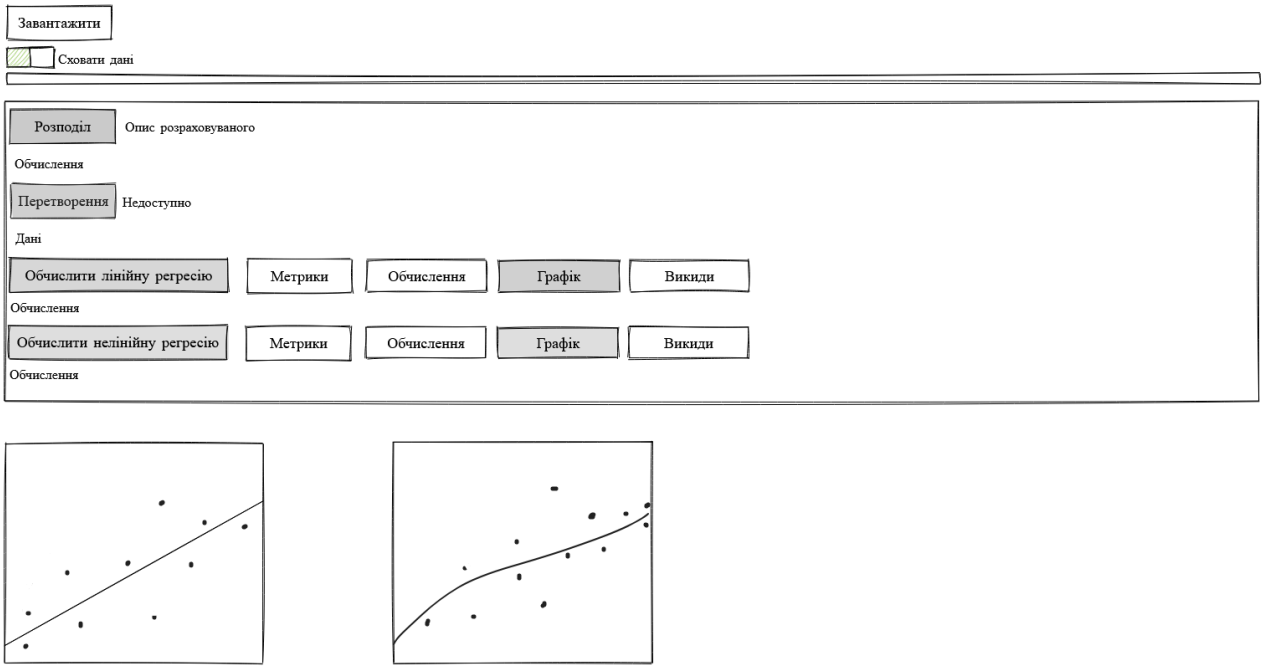

Рисунок 3.6 – Інтерфейс програмного забезпечення. Графік регресії

3.2 Технічний проект програмного забезпечення забезпечення для оцінювання розміру програмних проектів, що створені мовою Python

На основі ескізного проекту був розроблений технічний проект програмного забезпечення. В результаті розробленого технічного проекту були розроблені статична і динамічна модель. Наступна статична модель представлена діаграмою класів.

3.2.1 Структура класів

На основі інформаційної моделі приведеної в ескізному проекті, інформаційної моделі та моделі програмного інтерфейсу була розроблена статична модель, представлена діаграмою класів, яка наведена на рисунку 3.7:

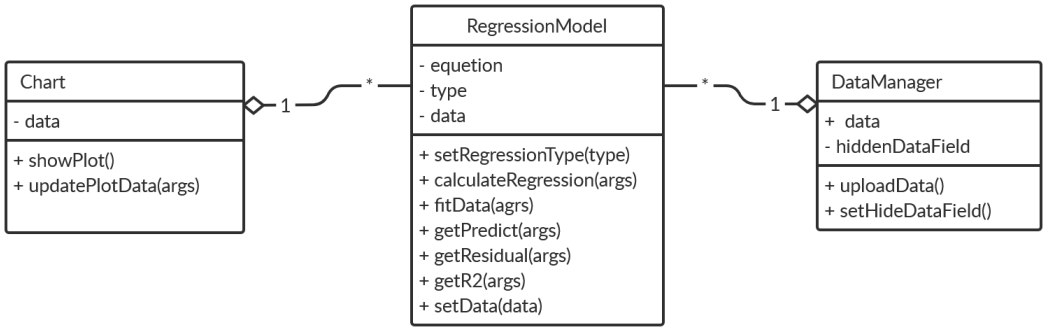

Рисунок 3.7 – Статична модель у вигляді діаграми класів

3.2.2 Специфікації класів

Клас: «DataManager»

Відповідальність: Даний клас виконує функцію менеджера даними.

Атрибути: data, hiddenDataField

Нижче представлені специфікації класів програми:

uploadData() – завантажити дані із файлу;

setHideDataField() – сховати поле з даними.

Клас: «RegressionModel»

Відповідальність: Даний клас відповідає за розрахунок даних регресійної моделі та передачу даних до класу Графіку.

Атрибути: equetion, type, data

Нижче представлені специфікації класів програми:

setRegressionType() – змінити тип регресії;

calculateRegression() – розрахувати регресії;

fitData() – корегувати дані;

getPredict() – отримати прогнозовані дані;

getResidual() – отримати залишки регресії;

getR2() – отримати коефіцієнт детермінації;

setData() – змінити дані.

Клас: «Chart»

Відповідальність: Даний клас відповідає за виведення даних на графік.

Атрибути: data

Нижче представлені специфікації класів програми:

showPlot() – відобразити графік лінійної регресії;

updatePlotData() – оновити точки на графіку.

3.2.3 Динамічна модель програмного забезпечення

Динамічна модель надає змогу зобразити на діаграмі послідовності зміні стану сутностей та можливостей використання цих сутностей актором. На основі діаграми варіантів використання та побудованої статичної моделі розробимо динамічну модель для усього процесу використання користувачем розроблюваного програмного забезпечення.

На рисунку 3.8 наведена діаграма послідовності, що містить у собі відносини сутностей системи та актора «Користувач»:

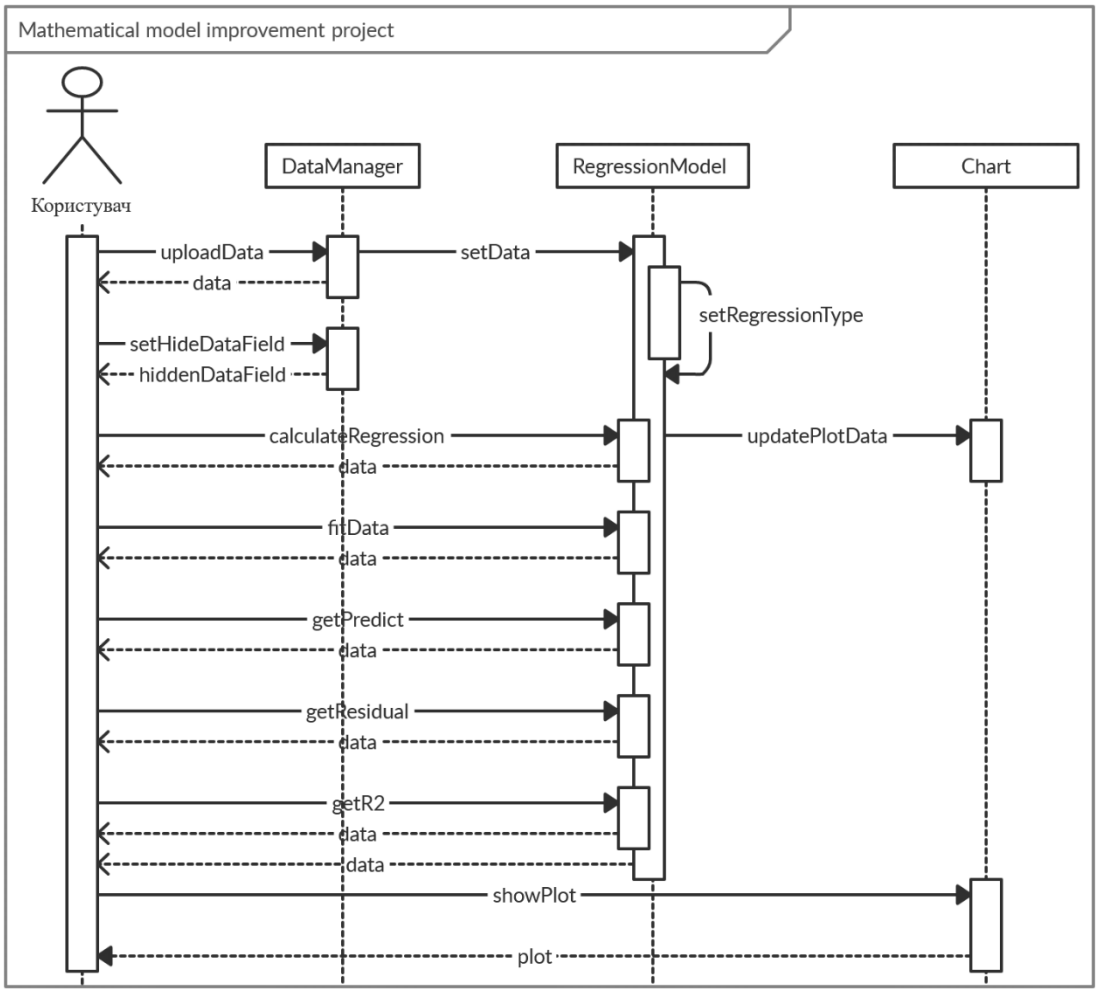

Рисунок 3.8 – Діаграма послідовності

На діаграмі послідовності актор «Користувач» має змогу звертатись через інтерфейс до трьох сутностей програмного забезпечення, а саме: «DataManager», «RegressionModel», «Chart».

DataManager дозволяє користувачу завантажити дані у програму, та передати її до RegressionModel. Сутність RegressionModel дозволяє користувачу робити маніпуляції з даними, обчислювати необхідні дані для розрахунку регресії, а також оновляти дані у сутності Chart. Сутність Chart в свою чергу надає змогу користувачеві зображати регресії та точки на графіках.

3.3 Робочий проект програмного забезпечення для оцінювання розміру програмних проектів, що створені мовою Python

3.3.1 Обґрунтування вибору мови та середовища для програмування

Для розробки програмного забезпечення «Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python» було вирішено обрати мову програмування JavaScript, тому що вона надає змогу швидко і легко розробляти інтерфейс програмного забезпечення. Дана мова не має потреби у спеціальній підготовці програми або компіляції для запуску. Інтерфейс розроблений за допомогою мови JavaScript має велику цінність завдяки тому, що його можна відобразити у браузері на комп'ютері, або на телефоні. Це надає змогу легкого поширення, або представлення програмного проекту, а також інтерфейс на веб-сторінці є зручним для розробки, адже перевірка зміни у програмі майже зовсім не займає часу. Мова JavaSript була зроблена для того, щоб «оживити» сторінки сайтів у браузері, це означає що інтерфейс повинен бути зручним, лаконічним, ефективним, але водночас приємним користувачу. Але ця тенденція набула значних змін на сьогоднішній день. Сьогодні значна кількість веб-сторінок працює по технології SPA, що в свою чергу є веб-сайтом з однією сторінкою, на якій усі функції виконуються динамічно, без оновлення сторінки. Ця технологія поширила подібний підхід і на розробку великих проектів, які використовують веб інтерфейси. Адже оновлення сторінки займає деякий час і негативно позначається на відносинах між окремими компонентами, їх станами та однозначно негативно впливає на зручність у використанні інтерфейсу користувачем. Для реалізації такої технології було обрано бібліотеку React.js. Ця бібліотека являється фреймворком для швидкої

розробки користувацького інтерфейсу, він надає велику швидкість, простоту та масштабованість. Для комфортної розробки було вирішено обрати PHPStorm у якості IDE, на підставі особистих переваг та великому досвіду у використані даної IDE. Це в першу чергу комерційна кросс-платформерна інтегрована середа для розробки в основному PHP проектів компанії JetBrains. Але вона підтримує усі ті функції і особливості розробки веб проектів, що і продукт WebStorm цієї ж компанії JetBrains, окрім цього вона є більш зручною у використанні і має більший спектр можливостей.

Ця середа для розробки має велику кількість необхідних можливостей для розробника веб-додатків. Зручний менеджер бібліотек, плагінів та зручна інтеграція терміналу, функцій дебагу і системи контролю версій.

До системи контролю версій тут входить зручна система VCS Git, яка надає зручний інтерфейс для усіх потрібних функцій управління онлайн репозитріїв.

3.3.2 Фізична модель даних

В даному пункті фізична модель представлена за допомогою діаграми компонентів та діаграми розгортання.

Діаграма компонентів описує проект програмного забезпечення у вигляді фізичного представлення системи та зв’язок кожного програмного компоненту архітектури розроблюваної системи. Даний проект реалізує програмне забезпечення, що в першу чергу реалізує функції, які не потребують втручання баз даних, або запитів до зовнішніх веб-серверів. Оскільки для системи важливий лише кліент-орієнтована частина, з якою працює користувач і весь функціонал який користувач використовує розраховується на стороні користувача у браузері. Для представлення системи у вигляді діаграми компонентів, можна розглянути

залежності між компонентами розроблюваного програмного забезпечення та використаними бібліотеками, що надають можливість реалізувати інтерфейс користувача. Компоненти відповідають окремим сутностям у рамках розроблюваного проекту, що зображено на рисунку 3.9:

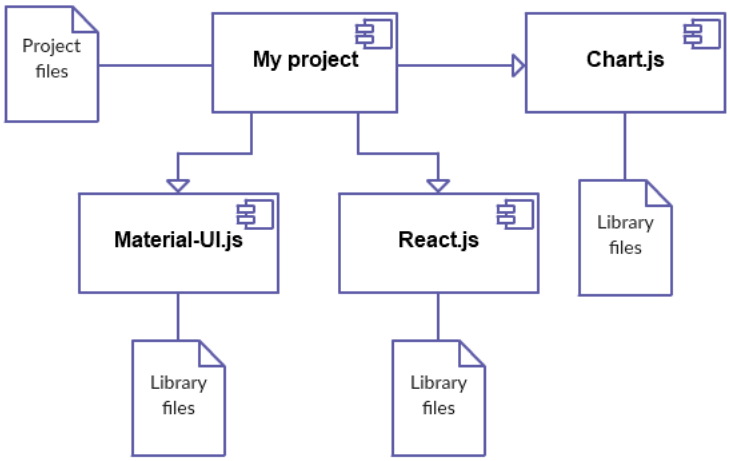

Рисунок 3.9 – Діаграма компонентів

Для представлення повного фізичного представлення системи програмного забезпечення необхідно вирішити, як і на яких платформах проект був розгорнутий. Для даної системи необхідними є проект, який використовується клієнтом у браузері, та веб-сервер, яким виступає веб-хостинг.

На рисунку 3.10 наведена діаграма розгортання:

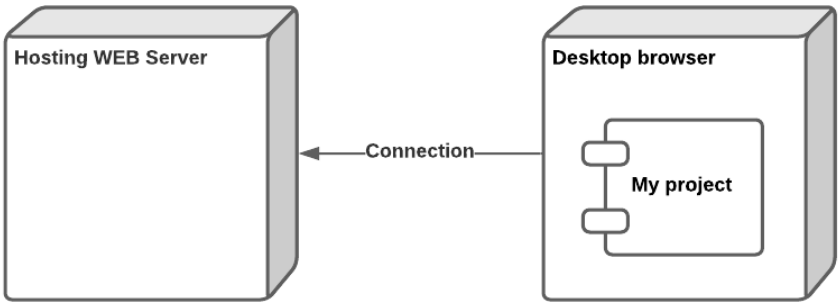

Рисунок 3.10 – Діаграма розгортання

3.3.3 Тестування варіантів використання

Для тестування варіантів використання зазвичай застосовують метод еквівалентної розбивки, в якому розбивають варіант використання на вхідну інформацію і кількість класів еквівалентності, задля перевірки усіх тестів еквівалентності розглядуваного класу, тому що помилка на одному тесті повинна бути виявлена іншими тестами цього класу еквівалентності.

У даному програмному забезпеченні є лише один варіант використання «Занесення вхідних даних у програму», який оброблює вхідні дані користувача, що може привести до можливих помилок, для чого необхідно провести тестування.

Тестування варіанта використання «Занесення вхідних даних у програму»

Вхідні дані: Текстовий файл

Текстовий файл має розширення типу .json містить дані необхідні для математичної моделі.

Вхідні дані у текстовому файлі повинні бути JSON структури.

Якщо дані мають помилку у побудові JSON об’єкту, або файл буде іншого розширення:

- у полі з даними буде виведена помилка некоректного json файлу.

- кнопки розподілу і перетворення будуть недоступні.

Тестові вимоги:

Перевірити, що у випадку неправильних даних, або файлу програмне забезпечення виводить помилку і блокує можливість використовувати кнопки.

Розробка тестових випадків для тестових вимог:

- перевірити, чи відбувається перевірка того, що обраний файл є неправильного розширення;

- перевірити, чи відбувається перевірка того, що вхідні дані мають

правильну структуру.

Тестові дані: Файл має розширення «.ру» і має пошкоджену структуру JSON об’єкту.

Результат: Було виведено помилку у поле для відображення вхідних даних, кнопки для обчислення розподілу і корегування даних були заблоковані.

3.3.4 Випробування програми

На початку роботи з програмним забезпеченням користувачу надається інтерфейс для виконання регресійного аналізу та виводу результатів на екран та графік.

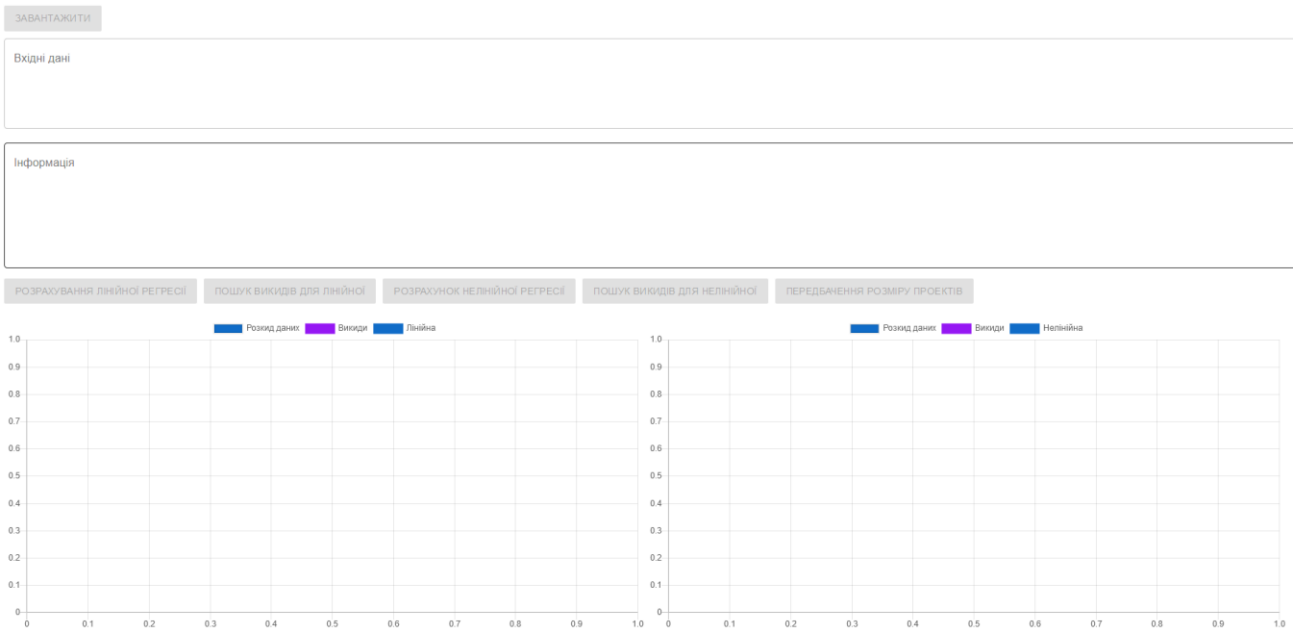

Рисунок 3.11 – Інтерфейс програми

На зображенні можна побачити усі необхідні для аналізу та оцінювання розміру програмних проектів функції. А саме: завантаження даних, розрахунок розподілу даних, перетворення вхідних даних, обчисленні лінійної регресії,

метрики лінійної регресії, графік лінійної регресії, викиди лінійної регресії, обчисленні нелінійної регресії, метрики нелінійної регресії, графік нелінійної регресії, викиди нелінійної регресії.

Перше що користувач повинен здійснити – завантажити дані, для цього необхідно натиснути кнопку «Завантажити», після цього на графіку рівняння регресії будуть відображені усі точки завантажених даних.

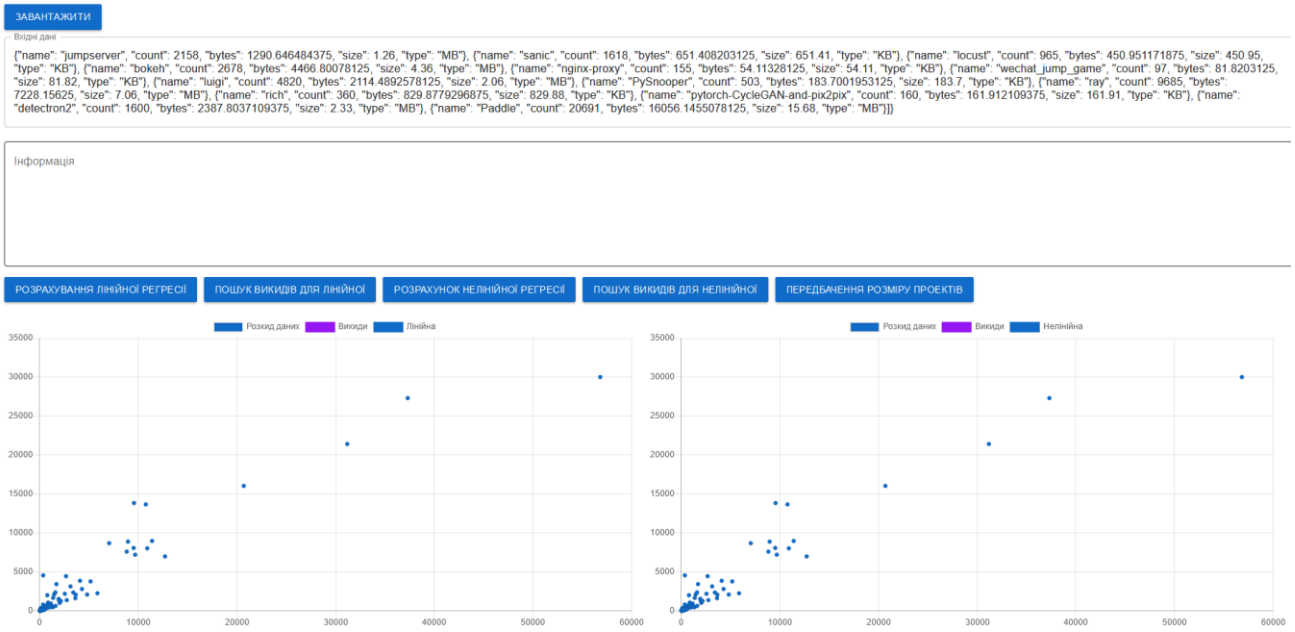

Рисунок 3.12 – Вхідні дані були завантажені

Але під час завантаження даних можна отримати ряд помилок. Якщо завантажувати файли, які мають неправильний тип файлу (не JSON), то користувач отримає помилку. А також, якщо файл має неправильну структуру JSON об’єкту, користувач отримує помилку про неправильні дані, що зображено на рисунку 3.13.

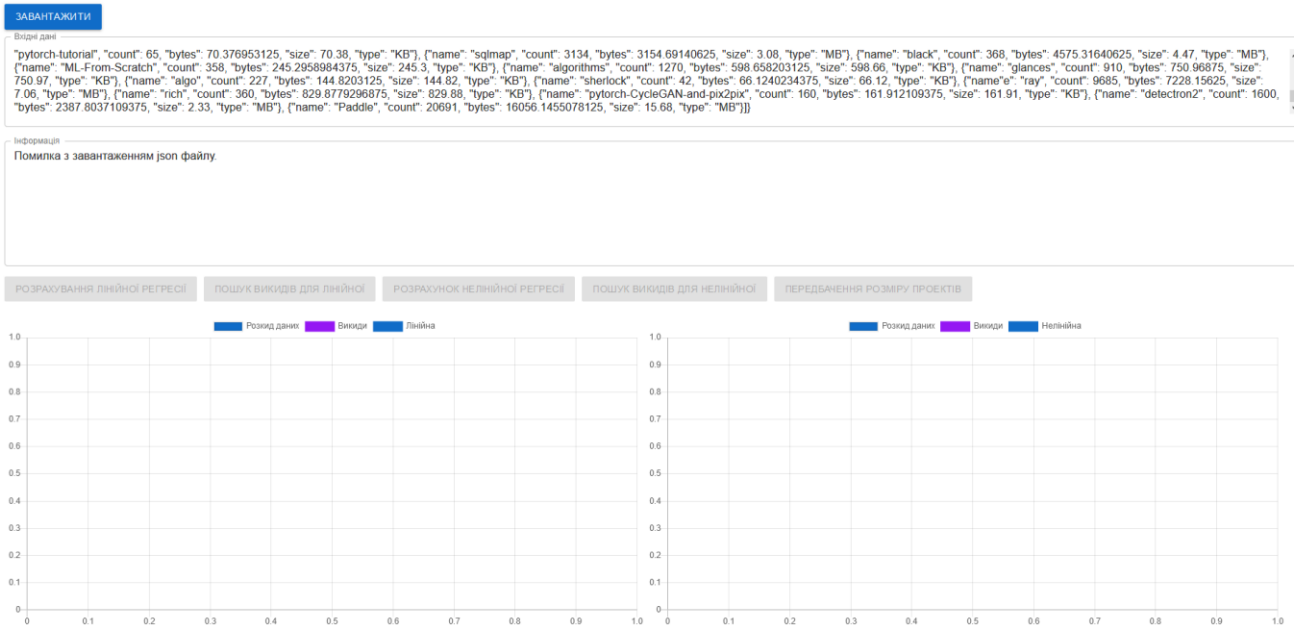

Рисунок 3.13 – Завантажуваний файл має помилки

Після завантаження файлу з помилками, користувач отримує повідомлення про помилку, а подальші функції програми недоступні для використання. Користувачу необхідно завантажити правильні дані для роботи з іншими функціями ПЗ.

Зазвичай вхідні дані мають ненормальний розподіл, та багато аномалій. Для перетворення даних до нормального розподілу, необхідно провести нормалізацію, для цього необхідно застосувати функції розрахунку розподілу та перетворення даних. Для цього необхідно натиснути кнопки «Розрахувати розподіл даних» та «Перетворення вхідних даних». Після перетворення даних, графік приймає іншу форму розподілу, що зображено на рисунку 3.14.

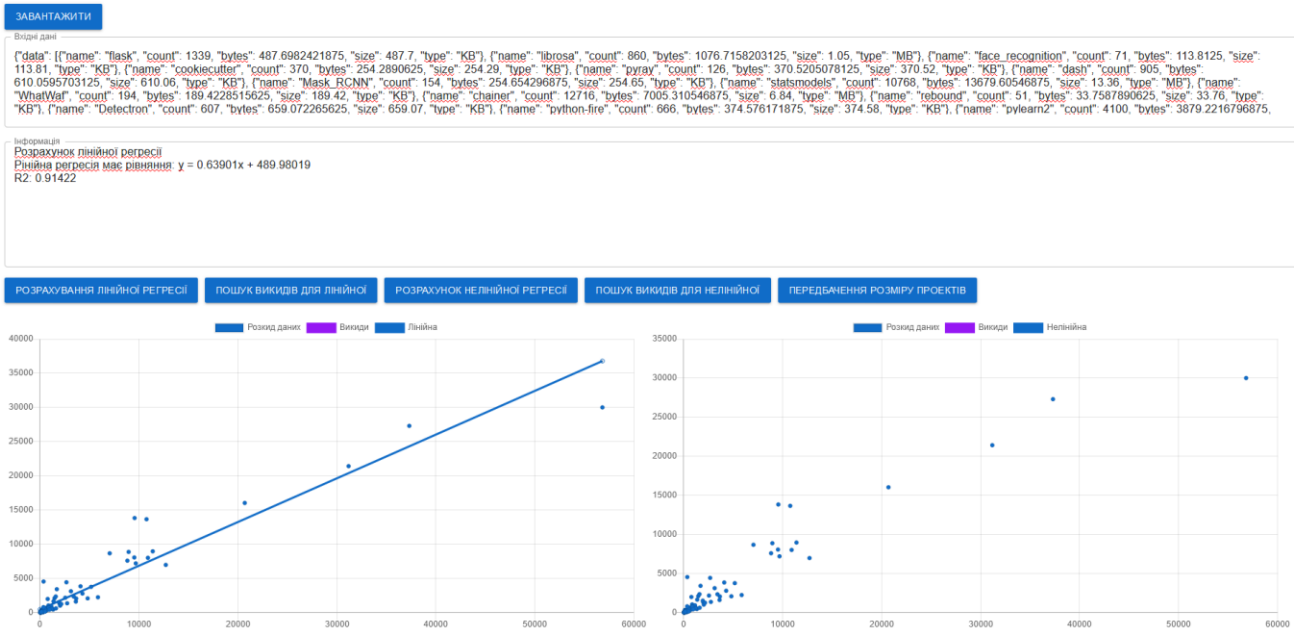

Рисунок 3.14 – Графік лінійної регресії

Після розрахованого рівняння регресії і метрик лінійної регресії, можна побачити поганий показник коефіцієнту кореляції, що значить, що дані мають аномалії, або значить що регресійна модель може бути вдосконаленою. Тому перше, що необхідно зробити – знайти викиди. Для цього необхідно натиснути на кнопку «Викиди лінійної регресії». Використання функції для викиду аномальних даних, покаже викиди на графіку іншим кольором, що зображено на рисунку 3.15.

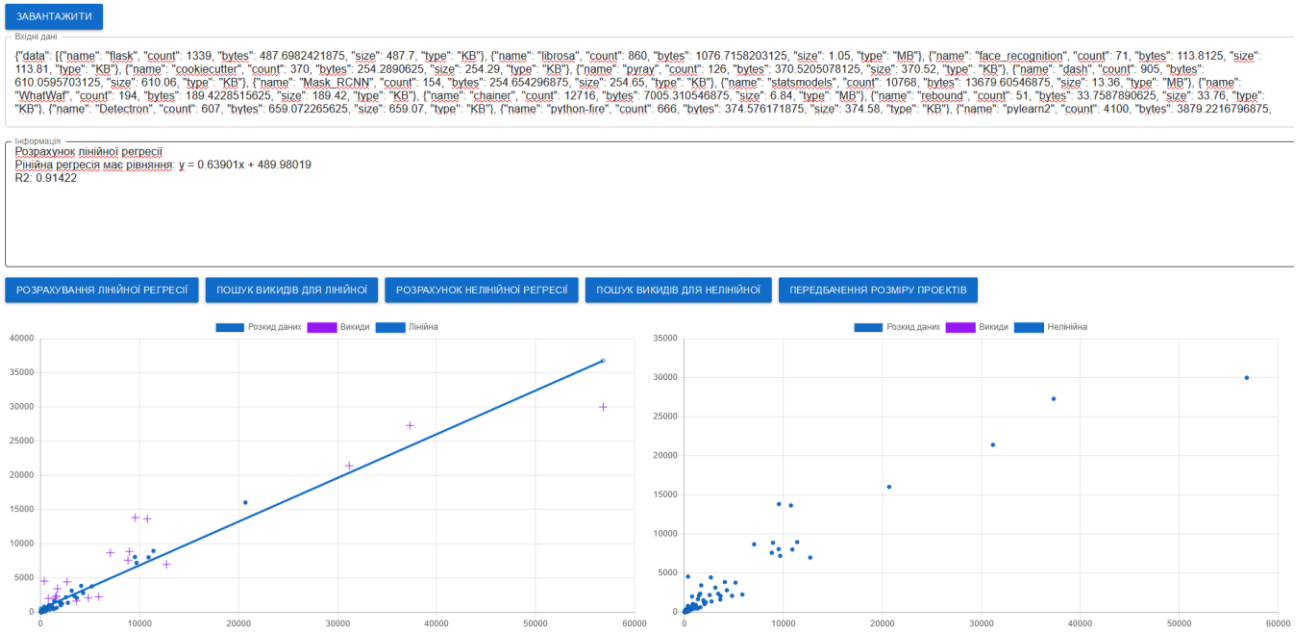

Рисунок 3.15 – Викиди лінійної регресії

Після визначення викидів графік лінійної регресії є неправильним, оскільки обчислення лінійної регресії має бути перераховано на нових даних без викидів. Для цього необхідно знову натиснути на кнопку «Обчислення лінійної регресії», після чого графік буде оновлено, а графік лінійної регресії буде перераховано на основі нових даних. Що зображено на рисунку 3.16.

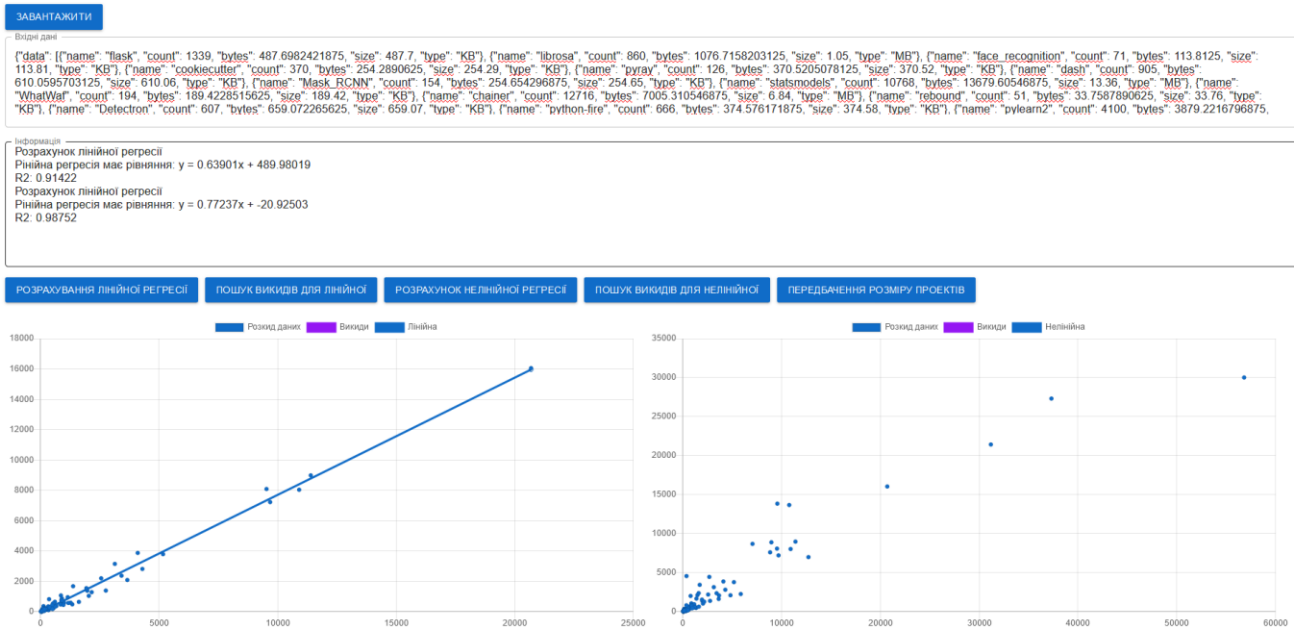

Рисунок 3.16 – Обчислення лінійної регресії без викидів

Після знайдених викидів можна перейти до моделі нелінійної регресії і розрахувати нелінійну регресію і вивести її на графік, що зображено на рисунку 3.17.

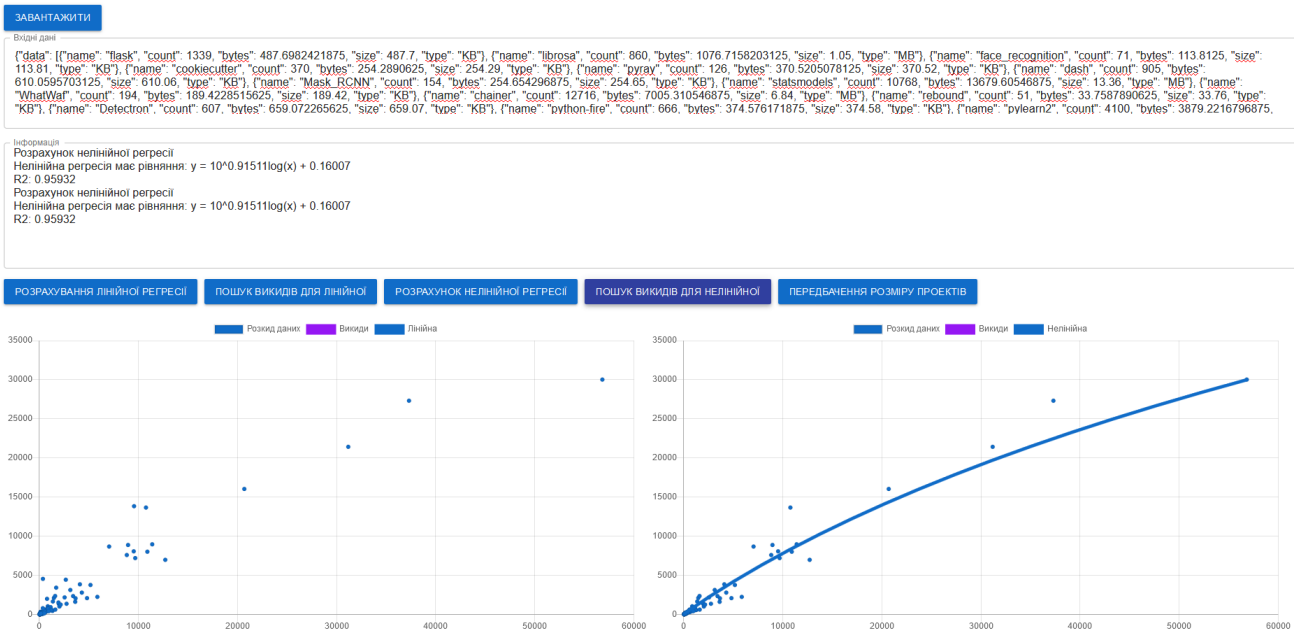

Рисунок 3.17 – Вивід нелінійної регресії

Для розрахунку розміру програмного проекту необхідно натиснути кнопку «Розрахунок розміру проектів» і ввести кількість методів:

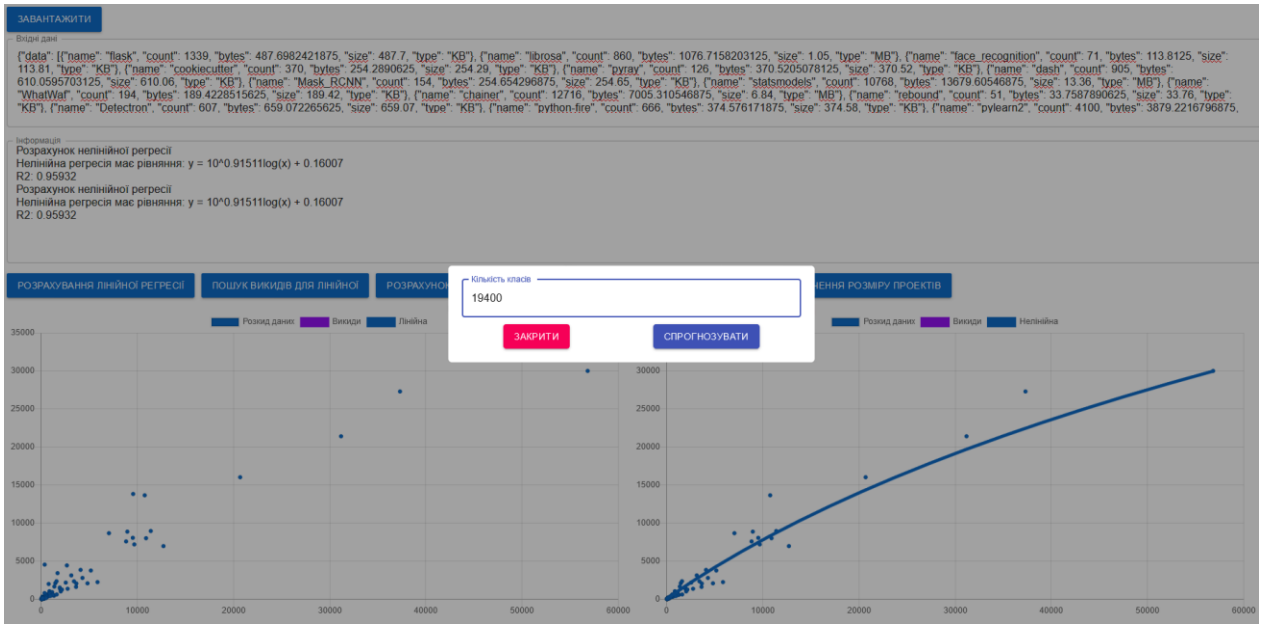

Рисунок 3.18 – Ввід кількості методів

Після розрахунку нелінійної регресії можна виконати розрахунок розміру програмного проекту, для цього не обхідно ввести кількість методів, після натискання кнопки «Розрахунок розміру проектів», що зображено на рисунку 3.19:

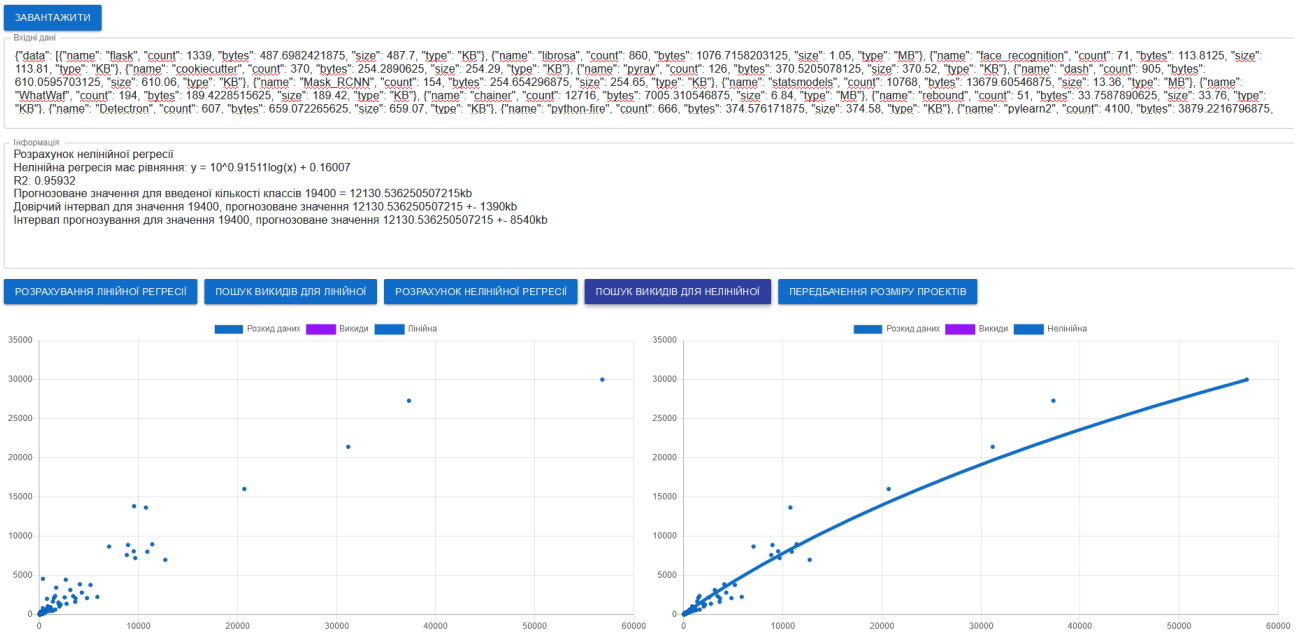

Рисунок 3.19 – Розрахований розмір для кількості методів програмного проекту

Таким чином було виконано випробування програми.

3.4 Висновки до розділу 3

У даному розділі були розроблені ескізний, технічний та робочій проекти. Під час розробки ескізного проекту була побудована модель варіантів використання з наведених у постановці задачі вимог до функцій програми, були розроблені специфікації до варіантів використання, сценарії варіантів використання, інформаційна модель та наведений інтерфейс програми у вигляді візуальної схеми. Під час розробки технічного проекту була з’ясована структура класів, специфікація класів та динамічна модель програмного забезпечення у

вигляді діаграми послідовності. Під час розробки робочого проекту програмного забезпечення був обґрунтований вибір мови та середовища для програмування, була наведена фізична модель даних та проведені тестування варіантів використання і випробування програмного забезпечення.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В процесі виконання кваліфікаційної роботи було розроблено програмне забезпечення для оцінювання розміру програмного забезпечення створених мовою Python.

Розроблене програмне забезпечення має ряд функцій:

1) занесення вхідних даних у програму;
2) обчислення розподілу вхідних даних;
3) перетворення вхідних даних;
4) обчислення лінійної регресії;
5) обчислення нелінійної регресії;
6) розрахунок метрик рівнянь регресії;
7) вивід обчислень регресії на екран;
8) вивід даних регресії на графік;
9) розрахунок викидів;
10) розрахунок інтервалів для введеної кількості методів;
11) вибіркове середнє для введеної кількості методів.

Програмне забезпечення було розроблене мовою програмування Javascript з використанням фреймворку React.js, що націлений на швидку розробку інтерфейсу користувача. Перевагами даної технології є:

1) відкритий код;
2) швидкодія розробленого інтерфейсу;
3) легка і швидка розробка;
4) реактивне оновлення веб-сторінок, без потреби у оновленні сторінки користувачем;
5) велика кількість бібліотек та додатків, що дозволяють побудувати необхідний функціонал інтерфейсу.

Для розробки були використані такі засоби:

1) середа розробки PHPStorm;
2) мова програмування: Javascript;
3) технології: HTML5, CSS3;
4) додатки: Chart.js;
5) фреймворк: React.js.

Текст програми представлено в додатку Б.

Основними факторами використання обраних технологій та засобів є:

1) швидкість обробки інформації;
2) полегшення створення інтерфейсів;
3) чуйний інтерфейс користувача.

Розроблене програмне забезпечення було успішно протестоване і усі задачі та поставлені вимоги були успішно виконані. Методика випробувань наведена у додатку Г.

Опис програми було представлено у додатку В.

Перевагою розробленого програмного забезпечення є те, що інтерфейс виконаний у вигляді веб-додатку, що дозволяє використовувати його без завантаження користувачем програми на комп’ютер. Тому таке ПЗ може бути використано не тільки на комп’ютері, а і на мобільних пристроях. Для цього необхідно мати сучасний браузер та вихід у мережу Інтернет.

Інструкція користувача наведена у додатку Д.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «УДОСКОНАЛЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ПРОЕКТІВ, ЩО СТВОРЕНІ МОВОЮ PYTHON»

Дана кваліфікаційна робота присвячена розробці програмного забезпечення для удосконалення математичної моделі для оцінювання розміру програмних проектів, що допоможе при розробці програмного забезпечення з правильною архітектурою, яка вплине на швидку розробку та якісну підтримку вже розробленої системи.

Ефективність економічна – результативність економічної діяльності, економічних програм і заходів, що характеризується відношенням отриманого економічного ефекту, результату до витрат чинників, ресурсів, що зумовило отримання цього результату, досягнення найбільшого обсягу виробництва із застосуванням ресурсів певної вартості. Ринок продуктів в IT – це система економічних, правових, організаційних відносин у сфері торгівлі продуктами інтелектуальної праці на комерційних засадах.

Для обґрунтування доцільності розробки з економічної точки зору необхідно розрахувати собівартість програмного забезпечення та прибуток, який буде отриманий в результаті продажу. Економічна доцільність витрат визначається річним економічним ефектом, для якого характерним показником створення і функціонування програмного продукту, що розраховується на стадії створення і впровадження – є ефективність витрат, яку характеризує строк окупності програмного забезпечення.

У даному підрозділі розраховується собівартість розробки програмного забезпечення, а також розрахунки економічної ефективності та строку окупності програмного забезпечення.

5.1 Розрахунок витрат на створення та впровадження програмного забезпечення

Витрати на розробку програмного забезпечення складаються з витрат на заробітну плату розробника, на амортизацію та експлуатацію ЕОМ, на якій виконується розробка, на засоби розробки та витрати на матеріали і комплектуючі.

Розробка програмного забезпечення виконується спеціалістом, місячний оклад якого складає 19500 грн. Додаткова заробітна плата складає 10% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 21450 грн./міс, а вартість сучасної ПЗВМ складає 10000->15000 грн. (середня вартість машини на базі AMD Ryzen). Вартість кіловат-години електроенергії дорівнює 0,90 грн. Витрати на допоміжні матеріали наведені в табл. 5.1

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума, грн.
Папір	90
Заправлення картриджа до принтера	130
Непередбачені витрати	200
Разом матеріали і комплектуючі вироби.	220

Вартість розробки системи розраховується за формулою:

$$С_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м} \quad (5.1)$$

де Т – тривалість розробки, міс.

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

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

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

$З_{е}$ – витрати на електроенергію;

$З_{м}$ – витрати на основні і допоміжні матеріали.

Розрахуємо $З_{е}$ при споживанні потужності 0,3 Вт, тривалості роботи на місяць рівної $21 * 8 = 168$ годин і вартості кіловат-години електроенергії 0,90 грн. до 100кВт включно та 1,68 після 100кВт отримаємо:

$З_{е} = (100 * 0{,}90 + 68 * 1{,}68) * 0{,}3 = 61{,}27$ грн.

Представимо всі поточні витрати на розробку програмного забезпечення в таблиці 4.2

Таблиця 5.2 – Витрати на розробку програмного забезпечення

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	міс.	3
Основна і додаткова заробітні плати	грн.	21450
Відрахування на соціальні заходи	грн.	6500
Загальногосподарські витрати	грн.	2050
Витрати на допоміжні матеріали	грн.	220
Витрати на електроенергію	грн.	61,27

Відповідно до формули (1) вартість розробки ПЗ складає:

$С_{пр} = (21450 + 6500 + 2050 + 61{,}27) * 3 + 220 = 90403{,}81$ грн.

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

$А_{об}$ = 15000 * 0,6 = 9000 грн.

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

$В_м$ = 15000 * 0,05 = 750 грн.

Річний обсяг робіт ПЕОМ у годинах визначається в такий спосіб:

$Ф_м$ = 253,3 * $Т_з$

де $Т_з$ – це середнє місячне завантаження устаткування (близько 7 годин), 253,3 – середня кількість робочих днів у році:

$Ф_м$ = 253,3 * 7 = 1773,1 год.

Витрати на електроенергію $З_е$ при 1773,1 годинах роботи устаткування в рік складуть:

$З_е$ = 1773,1 * 0,3 * 0,90 = 478,74 грн.

Експлуатаційні витрати для ПЕОМ за рік складуть:

$З_{зр}$ = 9000 + 750 + 478,74 = 10228.74 грн.

У перший рік витрати на створення й експлуатацію програмного забезпечення складуть:

$З_{се}$ = 90403,81 + 10228.74 = 100632.55 грн.

5.2 Розрахунок економічної ефективності розробки

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

Зарплата одного працівника в рік складає:

$19500 * 12 * 1 = 234000$ грн.

Річний економічний ефект розраховується по формулі:

$$Э_{рік} = \Delta C_n - E_n \cdot k, \quad (5.2)$$

де ΔC_n – вивільнені кошти після впровадження системи (234000 грн.) мінус експлуатаційні витрати (10228.74 грн.);

E_n – коефіцієнт ефективності(дорівнює коефіцієнту амортизації (0,6));

k – одноразові витрати на впровадження продукту (100632.55).

$Э_{рік} = 234000 - 10228.74 - 100632.55 * 0{,}6 = 163391.73$ грн.

Строк окупності системи розраховується по формулі:

$T = \frac{k}{\Delta C} = \frac{90403{,}81}{234000-10228.74} \approx 0.40$ року

Отже строк окупності програмного забезпечення складає 4.8 місяці.

5.3 Висновки

У цьому розділі були здійснені розрахунки на створення й експлуатацію програмного забезпечення, а також розрахунки економічної ефективності та строку окупності програмного забезпечення. Виходячи з розрахунків, впровадження такого програмного забезпечення окупить себе протягом першого року експлуатації за майже 5 місяців. Отже, можна зробити висновок, що дане програмне забезпечення економічно вигідне та обґрунтовано.

6 ОХОРОНА ПРАЦІ В ОФІСІ ФІРМИ

6.1 Вступна частина

Охорона праці – ряд систем розрахованих на догляд за життям та здоров'ям робітників у приміщені де працюють люди. До охорони праці стосуються процеси трудової діяльності, які включають багато заходів щодо захисту людської праці, а саме правові, лікувально-профілактичні, санітарно-гігієнічні, соціально-економічні та інші.

До функцій охорони праці входить саме проведення цих заходів, задля зниження впливу шкідливих факторів на організм робітників, шляхом застосування технік безпеки під час виконання роботи працівниками.

До техніки безпеки можна віднести ряд організаційних та технічних методів гарантування безпеки, що при виконанні необхідних для техніки безпеки заходів запобігають відсутністю небезпечних факторів, або зниження можливих випадків.

Робоче місце – місце постійного або тимчасового перебування працівника під час його трудової діяльності, в якому виконують виробничі завдання. Робоче місце є частиною виробничо-технологічної структури підприємства (організації), воно призначене для виконання частини технологічного (виробничого) процесу і визначається на основі трудових та інших діючих норм і нормативів.

Робоча зона – визначений простір, у якому розташовані робочі місця постійного або тимчасового перебування працівника під час його трудової діяльності, обмежений по висоті над рівнем підлоги або майданчика. До постійних відносяться робочі місця, на яких працює знаходиться більше 50% робочого часу за зміну або більше двох годин безперервно. Якщо робота

здійснюється в різних пунктах робочої зони, то постійним робочим місцем вважається вся робоча зона.

Умови праці – сукупність факторів виробничого середовища, що впливає на здоров'я і працездатність людини в процесі праці. Дослідження умов праці показали, що чинниками виробничого середовища в процесі праці є:

- санітарно-гігієнічна обстановка, яка визначає зовнішню середу в робочій зоні - мікроклімат, механічні коливання, випромінювання, температуру, освітлення та інші;
- психофізіологічні елементи: робоча поза, фізичне навантаження, нервово-психологічну напругу та інші, які обумовлені самим процесом праці;
- естетичні елементи: оформлення виробничих приміщень, обладнання, робочого місця, робочого інструменту та інші;
- соціально-психологічні елементи, складові характеристику так званого психологічного клімату.

Згідно Конституції України (ч. 4 ст. 43) кожна працездатна людина має право на належні, безпечні і здорові умови праці. А тому, у відповідності до вимог ст. 153 Кодексу законів про працю України, ст. 6 та ч. 1 ст. 13 Закону України «Про охорону праці», роботодавець зобов’язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці.

Отже, треба провести аналіз та виявити шкідливі та небезпечні фактори, які можуть впливати на працездатність, загрожувати здоров’ю чи життю працівників на робочих місцях у офісі, провести аналіз їх впливу на здоров’я, встановити граничнодопустимі рівні цих факторів на робочих місцях, тим самим визначити оптимальні умови необхідні для праці та навчання.

6.2 Аналіз шкідливих та небезпечних факторів в офісі фірми «Lexico Telecom»

Для аналізу шкідливих та небезпечних факторів необхідно розглянути характеристики умов праці робітників, а також можливі фактори, що можуть негативно вплинути на їхнє здоров’я.

Під умовами праці розуміють сукупність факторів та обставин трудового процесу і виробничого середовища, в якому здійснюється діяльність людини, що впливають на здоров’я та працездатність.

Під факторами трудового процесу розуміють основні його характеристики: важкість праці та напруженість праці. Важкість праці – це характеристика трудового процесу, яка відображає переважне навантаження на опорно-руховий апарат і функціональні системи організму (серцево-судинну, дихальну та інші, що забезпечують його діяльність). Напруженість праці – це характеристика трудового процесу, яка відображає навантаження переважно на центральну нервову систему, органи чуття, емоційну сферу працівника.

Під виробничим середовищем розуміють сукупність фізичних, хімічних, біологічних, психофізіологічних факторів на виробництві, що діють на людину. Усі ці фактори класифікують як небезпечні та шкідливі.

Небезпечні виробничі фактори – це ті, вплив яких на працівника призводить до травм, раптового погіршення здоров’я чи до смерті.

Шкідливі виробничі фактори – це ті, вплив яких на працівника може призвести до захворювання та зниження працездатності.

До фізичних небезпечних і шкідливих факторів належать:

– рухомі машини і механізми, рухомі частини виробничого обладнання, вироби, що пересуваються (матеріали, заготовки);

– підвищена запиленість і загазованість повітря робочої зони; –підвищена чи знижена температура поверхонь обладнання, матеріалів, повітря робочої зони;

– підвищені рівні шуму, вібрації, ультразвуку, інфразвукових коливань;

– підвищений чи знижений барометричний тиск і його різкі зміни;

– підвищена чи знижена вологість, рухомість, іонізація повітря;

– підвищений рівень іонізуючих випромінювань, напруги в електромережі, статичної електрики, електромагнітних випромінювань, напруженості електричного і магнітного полів;

– відсутність чи брак природного світла, знижена контрастність, пряма і відбита блискотливість, підвищена пульсація світлового потоку;

– підвищені рівні ультрафіолетової та інфрачервоної радіації;

– гострі краї, шершавість, заусиниці на поверхні заготовок, інструментів і обладнання;

– розташування робочого місця на значній висоті відносно землі (підлоги).

До хімічних небезпечних і шкідливих виробничих факторів належать речовини, які за характером впливу на організм людини поділяються на токсичні, подразнюючі, сенсабілізуючі, концерогенні і мутагенні. Ці хімічні речовини впливають на репродуктивну функцію людини. За шляхами проникнення в організм людини вони поділяються на ті, що проникають через органи дихання, шлунково-кишковий тракт, шкіряний покрив і слизові оболонки.

До біологічних небезпечних і шкідливих виробничих факторів належать патогенні мікроорганізми (бактерії, віруси, рикетсії, спірохети, грибки) та продукти їх життєдіяльності, а також рослини та живі істоти.

До психофізіологічних небезпечних і шкідливих виробничих факторів належать фізичні (статичні і динамічні) та нервово-психічні перевантаження (розумове перенапруження, перенапруження аналізаторів, монотонність праці, емоційні перевантаження).

До небезпечних факторів стосується і гігієнічні умови праці робочих місць, що негативно впливають на здоров'я людей, які там працюють.

Гігієнічне оцінювання умов і характеру праці на робочих місцях виконується на основі гігієнічної класифікації праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу. Гігієнічна класифікація базується на принципі диференціації умов праці залежно від фактично визначених рівнів указаних факторів у порівнянні з санітарними нормами, правилами, гігієнічними нормами, а також з урахуванням можливого шкідливого впливу їх на стан здоров’я працівників.

Умови праці та принципи гігієнічної класифікації поділяються на чотири класи.

Оптимальні умови праці (1-й клас) – умови, за яких зберігається не лише здоров’я працівників, а й створюються передумови для підтримання високого рівня працездатності. Оптимальні гігієнічні нормативи виробничих факторів встановлені для мікроклімату і факторів трудового процесу. Для інших факторів за оптимальні умовно приймаються такі умови праці, за яких несприятливі фактори виробничого середовища не перевищують рівнів, прийнятих за безпечні для населення.

Допустимі умови праці (2-й клас) – характеризуються такими рівнями шкідливих виробничих факторів виробничого середовища і трудового процесу, які не перевищують встановлених гігієнічних нормативів, а можливі зміни функціонального стану організму відновлюються за час регламентованого відпочинку чи до початку наступної зміни та не чинять несприятливого впливу на стан здоров’я працівників та їх потомство в найближчому і віддаленому періодах.

Шкідливі умови праці (3-й клас) – характеризуються такими рівнями шкідливих виробничих факторів, які перевищують гігієнічні нормативи і здатні

несприятливо впливати на організм працівника та на його нащадків. Шкідливі умови праці за ступенем перевищення гігієнічних нормативів і вираженості можливих змін в організмі працівників поділяються на чотири ступені.

Перший ступінь – умови праці характеризуються таким рівнем шкідливих факторів виробничого середовища і трудового процесу, які, як правило, зумовлюють функціональні зміни, що виходять за межі фізіологічних коливань (останні відновлюються при тривалішій, ніж початок наступної зміни, перерві контакту зі шкідливими факторами) та збільшують ризик погіршення здоров'я.

Другий ступінь – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які здатні спричинювати стійкі функціональні порушення, призводять у більшості випадків до зростання виробничо-зумовленої захворюваності, появи окремих ознак або легких форм професійної патології (як правило, без втрати професійної працездатності), що виникають після тривалої експозиції (10 років і більше).

Третій ступінь – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які призводять, окрім зростання виробничо-зумовленої захворюваності, до розвитку професійних захворювань, як правило, легкого і середнього ступенів важкості (з утратою професійної працездатності в період трудової діяльності).

Четвертий ступінь – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які здатні призводити до значного зростання хронічної патології та рівнів захворюваності з тимчасовою втратою працездатності, а також важких форм професійних захворювань (з утратою загальної працездатності).

Небезпечні (екстремальні) умови праці (4-й клас) – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища

і трудового процесу, вплив яких протягом робочої зміни (чи її частини) створює загрозу для життя, високий ризик виникнення важких форм професійних уражень.

Поширена думка, що офісні робітники працюють в приємній, безпечної обстановці. Хоча працювати в офісі не так небезпечно, як у багатьох інших місцях, тут також існує ряд джерел ризику для життя і здоров'я. Деякі з них представляють для людей, які працюють в офісі серйозну небезпеку.

Для працівника несприятливими подіями внаслідок впливу умов праці є втома, стрес, захворювання (хвороба) або травма.

Втома – це фізіологічний стан організму, що виникає внаслідок надмірно інтенсивного або тривалої діяльності, що виявляється тимчасовим зниженням функціональних можливостей людського організму. Розрізняють фізичну, розумову і емоційну втому.

Фізична втома проявляється порушенням функції м'язів: зниженням сили, точності, узгодженості і ритмічності рухів; виникає при інтенсивній або тривалій фізичної діяльності.

Розумова втома проявляється зниженням продуктивності інтелектуальної праці, ослабленням уваги (труднощі зосередження), уповільненням мислення, зниженням показників розумової активності, інтересу до роботи; виникає при інтенсивній інтелектуальній діяльності.

Емоційна втома проявляється помітним зниженням емоційних реакцій під впливом надсильних або монотонних подразників (стресів).

Надмірне робоче навантаження протягом тривалого часу або недостатній час відпочинку може привести до хронічної втомиабо перевтоми. Розрізняють розумовий і психічний (душевний) перевтома. В умовах сучасного ритму праці і життя все частіше у працівників з'являється синдром хронічної втоми.

Стрес є серйозною проблемою соціально-психологічного характеру для багатьох організацій. Він може бути викликаний багатьма факторами,

включаючи шум від переповнення приміщення людьми або працюючого обладнання, поганих відносин з начальством та / або колегами, збільшенням робочого навантаження і недостатньою свободою дій при виконанні роботи.

Іншим масово поширеним несприятливим наслідком праці є те або інше захворювання: нездужання, погане самопочуття, симптоми, які можуть проходити бурхливо або відносно швидко минути («гострі» -з медичної термінології) або тривати (роками) або періодично загострюватися (хронічні).

Захворювання, спровоковані умовами праці, часто називають виробничо-зумовленими захворюваннями.

Специфічний вплив факторів, пов'язаний з конкретними виробничими чинниками, призводить до розвитку певних, викликаних цими факторами, захворювань. Оскільки вони викликані несприятливими умовами праці конкретних робочих місць конкретних професій, то їх називають професійними захворюваннями.

Захворювання кістково-м'язової системи та пошкодження м'яких тканин типу тендиніту виникають в результаті використання офісних меблів та обладнання, які не відповідають індивідуальним анатомічним особливостям. Тендинит може розвинутися в результаті повторюваних рухів окремих частин тіла. Так, наприклад, у службовців починають боліти пальці від тривалого писання або при роботі з дуже товстими папками, які доводиться постійно діставати із шаф і повертати на місце. Багато офісних робітників страждають від різноманітних RSI, типу захворювань зап'ястя і грудної клітини, через погану підгонки меблів і устаткування і відсутності перерв у друкуванні (на клавіатурі комп'ютера) або інших періодично повторюваних діях. Недосконала конструкція меблів і устаткування призводять також до порушення постави і здавлення нервів нижніх кінцівок, тому що багато офісних робітників велику частину часу

проводять сидячи. Всі ці фактори сприяють виникненню захворювань попереку і нижніх кінцівок в такій же мірі, як і тривале стояння на ногах.

Безперервне використання комп'ютерів і погане загальне освітлення викликає напругу зору, в результаті чого воно поступово погіршується, з'являються головні болі, печіння і відчуття втоми в очах. Регулювання рівня освітленості приміщення і контрастності зображення на екрані комп'ютера, а також часта перефокусування очей необхідні для попередження виникнення проблем із зором. Освітлення має відповідати характеру виконуваної роботи.

6.3 Розрахунок рівня освітлення в офісі фірми «Lexico Telecom»

Рівень освітленості впливає на працездатність і самопочуття людини. Державні норми це враховують і визначають, яким має бути освітлення приміщень, залежно від їх призначення. Відштовхуючись від цих норм, планують освітлення об'єктів, а отже, від них безпосередньо залежить і те, які світильники і скільки лампочок необхідно купити.

Стосовно ДБН (Державно будівельні норми) є конкретні показники якості освітлення приміщень різних призначень: житлових, офісних, складських, виробничих. Спираючись на ці дані, розраховують рівень освітленості, враховуючи площу приміщення, висоту стелі, відбиваючу здатність поверхонь, тип світильника та інші параметри. Стосовно необхідного нам показника в офісних приміщеннях, нормою освітлення буде 300 Лк.

Норми виникли після того як були враховані всі найбільш важливі особливості фізіологічних зорових процесів людини і фактори, які впливають на концентрацію, зорову напругу під час роботи, відпочинку (наприклад, контрастність, колір робочої поверхні).

Для розрахунку рівня освітленості приміщення зазвичай використовується спосіб, який передбачає використання даних про величину світлового потоку, який вимірюється люменами (Лм). Світловий потік, що випромінює лампа, який вказується на упаковці придбаної лампи.

Для визначення величини світлового потоку використовується формула:

Норма освітлення * Площа * Коефіцієнт висоти стелі

При висоті стелі від 2,5 м до 2,7 м коефіцієнт дорівнює 1, при висоті 2,7-3 метрів - 1,2, якщо висота становить від 3 до 3,5 метрів - 1,5, від 3,5 м до 4,5 м - 2. Порахуємо, яка величина світлового потоку необхідна вітальні площею 30 квадратних метрів з висотою стель 2,5 м:

150 * 30 * 1 = 4500 (Лм)

Оскільки різні лампочки випромінюють світло різної сабо, потрібна їх кількість залежить від того, що ми виберемо – світлодіодну, енергозберігаючу, галогенну або лампу розжарювання, а також від того, якої потужності вона буде.

Таблиця 6.1 – Приклад різних видів ламп і їх потужності

Лампочка	Потужність, Вт	Світловий потік, Лм	Потрібно лампочок, шт
Розжарювання	40	470	4500/470=9,5
Галогенна	32	470	4500/470=9,5
Енергозберігаюча	15	700	4500/700=6,4
Світлодіодна	10	750	4500/750=6

Отже, для освітлення вітальні площею 30 квадратних метрів необхідно 10 галогенних ламп потужністю 30 Вт, стільки ж більш потужних лампочок розжарювання, 7 енергозберігаючих по 15 Вт кожна або 6 світлодіодних 10-

ватних лампочок. Після підрахунків підсумок з десятковими знаками завжди округляємо в більшу сторону.

Більш складний спосіб розрахунку включає більше змінних:

N = (E * S) / (U * p * Fi * Кз), де N – кількість світильників, E – норма освітленості, S – площа об'єкта, Кз – коефіцієнт запасу (залежить від виду лампочки, ступеня забруднення приміщення), U – коефіцієнт використання світлового потоку (залежить від висоти стін, площі кімнати, типу світильника, кольору покриття стелі, стін, підлоги), p – кількість патронів світильника, Fi – значення світлового потоку однієї лампи. Наприклад, для вітальні з першого прикладу планується встановити точкові світильники. Колір покриття стелі - білий, стіни світлі, на підлозі - темно-коричневий ламінат. Для об'єкта показник E становить 150 (дивимося в першій таблиці), S – 30, Кз – 1 (припускаємо, що купують світлодіодні лампи потужністю 10 Вт), U – 0,46, p – 1, Fi – 750.

Підставляємо дані:

N = (150 * 30) / (0,46 * 1 * 750 * 1) = 4500 / 345 = 13.

Виходячи зі розрахованої формули було знайдено оптимальне значення кількості світильників, тобто слід розмістити 13 світильників.

6.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

З метою запобігання або зменшення впливу шкідливих і небезпечних виробничих чинників під час підготовки робочих місць, або робочої зони, необхідно слідувати загальним вимогам, що встановлені законодавством Україні про працю.

Відповідно до законів України «Про охорону праці» умови праці на робочому місці, безпека технологічних процесів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам законодавства.

Більшість нормативів щодо умов праці офісних працівників встановлено на рівні державних стандартів**.** Відповідно до ч. 1 ст. 13 Закону про охорону праці роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів.

Як відомо, тривала робота за комп'ютером та з документами при недостатньому рівні освітленості може призвести до значного перенапруження зору, тому вимоги до освітлення є досить важливими.

Додатково, окрім вже перелічених документів, вимоги до освітлення встановлено ДБН В.2.5-28-2006 «Природне і штучне освітлення», затвердженими наказом Мінрегіону від 15.05.2006 р. № 168.

Як вже зазначалося, відносно вікон робоче місце необхідно організовувати так, щоб природне світло було з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98). Робоче місце необхідно розміщувати таким чином, щоб уникнути попадання прямого світла в очі. Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідне застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат (п. 4.19 ДСанПіН 3.3.2.007-98).

Штучне освітлення приміщення має здійснюватись системою загального рівномірного освітлення (п. 3.2.2 ДСанПіН 3.3.2.007-98). У приміщеннях при переважній роботі з документами допускається використання системи

комбінованого освітлення, тобто встановлення світильників місцевого освітлення додатково до загального.

Як джерела штучного освітлення необхідно використовувати люмінесцентні лампи. Згідно з п. 3.2.5 ДСанПіН 3.3.2.007-98 система загального освітлення має бути у вигляді суцільних або переривчатих ліній світильників, що розташовані збоку від робочих місць (зазвичай ліворуч) паралельно лінії зору працівників.

Допускається застосування ламп розжарювання у світильниках місцевого освітлення та, у разі влаштування відбитого освітлення у виробничих чи адміністративно-громадських приміщеннях, металогалогенних ламп потужністю 250 Вт.

Коефіцієнт пульсації не повинен перевищувати 5 % (п. 3.2.14 ДСанПіН 3.3.2.007-98). Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300 – 500 лк. Світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Для забезпечення нормованих значень освітленості у приміщеннях відповідно до п. 3.2.15 ДСанПіН 3.3.2.007-98 необхідно мити вікна і світильники не рідше 2 разів на рік, а також своєчасно замінювати лампи, що перегоріли.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

7.1 Вступна частина

Усе необхідне для життєдіяльності людина отримує з природи: повітря, воду, сировину для промисловості. Людське суспільство як частина природи може бути тільки в постійній взаємодії з нею. Вплив людини на навколишнє середовище є перетворюючим, що змінює її, причому далеко не завжди в кращу сторону, тому збереження природного середовища і розумна охорона природи - одна з най гостріших проблем, що стоять перед людством, особливо в сучасних умовах.

Раціональне використання землі, лісу, атмосфери і водних ресурсів в Україні передбачено Конституцією. В даний час у сфері охорони навколишнього середовища діє цілий ряд нормативних актів: Закон України «Про охорону навколишньої природного середовища»; Постанова Уряду України «Про затвердження порядку визначення плати і її граничних розмірів за забруднення навколишньої природного середовища» і інші.

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. Складовою частиною навколишнього середовища є природне середовище. Перед сучасним суспільством стоїть завдання не тільки зберегти природу, а й запобігти негативним наслідкам господарської діяльності людини в майбутньому.

Охорона навколишнього середовища являє собою важку комплексну проблему, яка має відношення до всього суспільства в цілому і до кожного окремого громадянина.

Розмова йде про рішення життєво важливої проблеми – захисту і охорони здоров'я нинішнього і майбутнього покоління людей від шкідливих наслідків їх науково-технічної і промислової діяльності.

На початку своєї історії людина задовольнялась лише простими фізіологічними потребами (в їжі, одязі, житлі). З розвитком суспільства використання природних ресурсів для задоволення його матеріальних потреб весь час зростало. Нині людина дуже активно впливає на природу.

Одне з небажаних, але очевидних наслідків технічного процесу – забруднення оточуючого середовища вторинними продуктами виробничо-технічної діяльності.

В результаті промислової діяльності природа потерпає постійні зміни. Так в Україні суттєво скоротилася площа зелених насаджень; відбувається підкислення ґрунту і води; відходи промисловості, в тому числі різні високотоксичні речовини, забруднюють повітря, водойми, ґрунти; в результаті спалювання великої кількості мінерального палива в біосфері збільшується концентрація вуглекислоти що може призвести до зміни теплового режиму (клімату) поверхні всієї планети. Наслідки всього цього відбиваються на здоров'ї людей. Так, з року в рік збільшується кількість випадків серцево-судинних і ракових захворювань.

Тривалий час панувала помилкова думка, ніби багатства природи невичерпні, а тому, мовляв, можна і не турбуватись про їх відтворення і відновлення.

Чому сьогодні так гостро ставиться питання щодо охорони природи і раціонального використання її ресурсів? Це насамперед пов'язано з тим, що

природним ресурсам планети і в Україні зокрема, вже завдано величезної шкоди. За останнє століття близько двох мільярдів гектарів земель – 15 відсотків усієї земної суші – зруйновано водою і вітровою ерозією. За всю історію людського суспільства на земній кулі знищено дві третини лісів. Нині підприємства викидають у води і повітряне середовище стільки забруднюючих речовин, що завдають серйозної шкоди населенню і природному середовищу на великих відстанях.

Кількість відходів на протязі тривалого часу збільшувалась пропорційно росту виробництва і населення. Доки є якості сировини широко використовувались речовини рослинного і тваринного походження, відходи, які утворювались залучалися силами природи в кругообіг речовин, природа забезпечувала самоочищення. Але зараз все частіше використовуються речовини синтетичного і мінерального походження. Відходи синтетичних миючих засобів не засвоюються розкладаючими мікроорганізмами, вони накопичуються в водоймах куди вони потрапляють зі стічними водами і забруднюють їх. При спалюванні нафтового палива в атмосферу разом з димовими газами, окрім оксидів вуглецю (CO2, CO) викидаються оксиди сірки (SO2), які взаємодіють з вологою і киснем повітря і утворюють сірчану кислоту – утворюються так звані «кислотні дощі». Під впливом кислотних дощів відбувається швидке підкислення води у річках, озерах, ставках та інших водоймах. Під впливом кислотних дощів збільшилась кислотність ґрунтів. Таких прикладів можна навести дуже багато.

Іншим поширеним забрудненням природного середовища являються ядохімікати і мінеральні добрива, які застосовуються у сільському господарстві. За останні роки застосування мінеральних добрив в Україні збільшилось в 43 рази, а різноманітних ядохімікатів в 10 разів. В результаті інтенсивної хімізації вдається отримувати більший врожай. Але одночасно зростає ступінь забруднення ґрунту, водоймищ і продуктів харчування.

Безцінним нашим багатством є вода. Значення її для людини загальновідоме. В Україні за даними науково-дослідних установ запаси води на душу населення постійно зменшуються, тоді як потреби населення, промисловості і сільського господарства щороку зростають. Особливо велику кількість води витрачає енергетика, металургійна, хімічна, легка промисловість, сільське господарство. Вже нині в ряді міст відчувається нестача прісної води. Вода багатьох річок та озер непридатна для життя і навіть купання. До 30 % мінеральних добрив змивається з полів і потрапляє у водойми. У водоймах, збагачених поживними речовинами, швидко розмножуються водорості, як результат – «цвітіння води». Потім водорості відмирають і починається їх гноїння що супроводжується споживанням кисню. Утримання кисню у водоймі скорочується і починає гинути риба. Така вода стає непридатною для використання в побуті і навіть у технічних цілях.

Тому наше обов`язкове завдання є – приділяти велику увагу та зберігати природне середовище.

7.2 Забруднення навколишнього середовища підприємствами України

Однією із основних засад внутрішньої та зовнішньої політики України є збереження навколишнього середовища та його складових, що є життєво необхідним для існування людини, її нинішнього й прийдешніх поколінь.

З метою виконання цього Україна визнає забезпечення екологічної безпеки одним із основних напрямів державної політики національної безпеки України.

По своїй суті державна екологічна політика, серед іншого, спрямована на вирішення існуючих екологічних проблем, що призводять до негативних

екологічних, соціальних та економічних наслідків, а також на попередження їх виникнення й поширення тощо.

Важливу роль у формуванні та реалізації екологічної політики на державному та регіональному рівні, а також у збереженні навколишнього середовища, відіграє громадськість, зокрема організації громадянського суспільства.

Зміна клімату є однією з основних проблем світового розвитку з потенційно серйозними загрозами для глобальної економіки та міжнародної безпеки внаслідок підвищення прямих і непрямих ризиків, пов'язаних з енергетичною безпекою, забезпеченням продовольством і питною водою, стабільним існуванням екосистем, ризиків для здоров'я і життя людей5.Основними напрямками діяльності в Україні щодо запобігання зміні клімату, які зазначені у Концепції з реалізації державної політики у сфері зміни клімату на період до 2030 року є:

- скорочення антропогенних викидів і збільшення абсорбції парникових газів та забезпечення поступового переходу до низьковуглецевого розвитку держави;

- адаптація до зміни клімату, підвищення опірності та зниження ризиків, пов'язаних із зміною клімату.

Згідно з Рамковою конвенцією ООН про зміну клімату, в результаті людської діяльності відбулося істотне збільшення концентрації парникових газів в атмосфері, що посилює природний парниковий ефект, і може призвести до додаткового потепління поверхні і атмосфери Землі та несприятливо вплинути на природні екосистеми і людство.

У документах Робочої групи МГЕЗК говориться, що найбільший внесок у зміну клімату дають сполуки, які відносяться до «парникових газів»: насамперед вуглекислий газ (діоксид вуглецю) і метан.

Забруднення атмосферного повітря – змінення складу і властивостей атмосферного повітря в результаті надходження або утворення в ньому фізичних, біологічних факторів і (або) хімічних сполук, що можуть несприятливо впливати на здоров’я людини та стан навколишнього природного середовища.

Як зазначається в Законі України «Про Основні засади (стратегію) державної екологічної політики України на період до 2030 року», забруднення атмосферного повітря є однією з найгостріших екологічних проблем. На сьогодні рівень забруднення атмосферного повітря великих міст і промислових регіонів є високим, незважаючи на спад виробництва в Україні.

За даними Всесвітньої організації охорони здоров’я, забруднення повітря є одним з основних факторів ризику для здоров'я, пов'язаних з навколишнім середовищем. Чим нижче рівні забруднення повітря, тим менше серцево-судинних і респіраторних захворювань як в тривалій, так і в короткостроковій перспективі.

Роки безконтрольної експлуатації природних ресурсів призвели до того, що у багатьох районах забруднення повітря у десятки разів перевищує гранично допустимі норми.

Головним джерелом забруднення атмосферного повітря в Україні від викидів стаціонарних джерел є підприємства паливно-енергетичного комплексу - 36% від загального обсягу викидів, підприємства обробної - 35% та видобувної промисловості - 25%. Основними забруднюючими речовинами є оксиди вуглецю, азоту, диоксиди сірки, аміак, феноли, формальдегід, бензапірен. Хоч обсяги викидів забруднюючих речовин останнім часом, передусім через зупинку багатьох підприємств, зменшилися, проте в деяких промислових регіонах вони і нині значно перевищують гранично допустимі норми.

Економіка республіки не була орієнтована на такі "дрібниці", як турбота про екологічно чисте середовище, екологічно безпечні технології виробництва,

здоров'я людей. Навіть зараз, коли здавалося б, усім зрозуміло, що господарювати, як раніше, згубно, - дізнаємося, що значну частину з виділених на нове будівництво коштів буде використано на спорудження екологічно небезпечних об'єктів. На своїй рідній землі продовжуємо господарювати не як справжні володарі національних багатств, а ніби тимчасові окупанти.

Велику стурбованість викликає неблагополучна в екологічному відношенні столиця України. Так, Київ, який, по суті, не має металургійної і видобувної промисловості, за загазованістю повітря, в тому числі й автотранспортом, попереду таких промислових центрів, як Запоріжжя, Кривий Ріг, Харків, Макіївка, Комунарськ. Індекс забруднення в Києві у 6 разів вищий, ніж у Львові. Обсяги викидів продуктів промисловості й транспорту (насамперед, сірковуглецю, діоксиду азоту, фенолу й аміаку) постійно зростають і сягають вже 330 тис. т на рік. З понад 40 тис. промислових підприємств і об'єктів міста лише третина має очисні споруди. Серед злісних отруювачів повітря - 5 гігантських ТЕЦ і десятки районних котелень із застарілою системою очищення, які зараз перейшли на «альтернативне» паливо природному газу – вугіль, що значно вплинуло на збільшення рівня забруднення.

6.3 Розробка заходів щодо зменшення забруднення

Охорона навколишнього природного середовища, раціональне використання природних ресурсів, забезпечення екологічної безпеки життєдіяльності людини – невід'ємна умова сталого економічного та соціального розвитку України.

Відносини у галузі охорони навколишнього природного середовища в Україні регулюються законом України «Про охорону навколишнього природного середовища», а також розроблюваними відповідно до нього земельним, водним, лісовим законодавством, законодавством про надра, про охорону атмосферного

повітря, про охорону і використання рослинного і тваринного світу та іншим спеціальним законодавством.

Основними принципами охорони навколишнього природного середовища є:

1. пріоритетність вимог екологічної безпеки, обов'язковість додержання екологічних стандартів, нормативів та лімітів використання природних ресурсів при здійсненні господарської, управлінської та іншої діяльності;

2. гарантування екологічно безпечного середовища для життя і здоров'я людей;

3. запобіжний характер заходів щодо охорони навколишнього природного середовища;

4. екологізація матеріального виробництва на основі комплексності рішень у питаннях охорони навколишнього природного середовища, використання та відтворення відновлюваних природних ресурсів, широкого впровадження новітніх технологій;

5. обов'язковість екологічної експертизи;

6. гласність і демократизм при прийнятті рішень, реалізація яких впливає на стан навколишнього природного середовища, формування у населення екологічного світогляду;

7. науково обґрунтоване нормування впливу господарської та іншої діяльності на навколишнє природне середовище;

8. компенсація шкоди, заподіяної порушенням законодавства про охорону навколишнього природного середовища;

9. встановлення екологічного податку, збору за спеціальне використання води, збору за спеціальне використання лісових ресурсів, плати за користування надрами відповідно до Податкового кодексу України.

Підприємства, установи, організації та громадяни - суб'єкти підприємницької діяльності, що здійснюють викиди забруднюючих речовин в

атмосферне повітря та діяльність яких пов'язана з впливом фізичних та біологічних факторів на його стан, зобов'язані:

1. здійснювати організаційно-господарські, технічні та інші заходи щодо забезпечення виконання вимог, передбачених стандартами та нормативами екологічної безпеки у галузі охорони атмосферного повітря, дозволами на викиди забруднюючих речовин тощо;

2. вживати заходів щодо зменшення обсягів викидів забруднюючих речовин і зменшення впливу фізичних факторів;

3. забезпечувати безперебійну ефективну роботу і підтримання у справному стані споруд, устаткування та апаратури для очищення викидів і зменшення рівнів впливу фізичних та біологічних факторів;

4. здійснювати контроль за обсягом і складом забруднюючих речовин, що викидаються в атмосферне повітря, і рівнями фізичного впливу та вести їх постійний облік;

5. заздалегідь розробляти спеціальні заходи щодо охорони атмосферного повітря на випадок виникнення надзвичайних ситуацій техногенного та природного характеру і вживати заходів для ліквідації причин, наслідків забруднення атмосферного повітря;

6. забезпечувати здійснення інструментально-лабораторних вимірювань параметрів викидів забруднюючих речовин стаціонарних і пересувних джерел та ефективності роботи газоочисних установок;

7. забезпечувати розроблення методик виконання вимірювань, що враховують специфічні умови викиду забруднюючих речовин;

8. використовувати метрологічно атестовані методики виконання вимірювань і повірені засоби вимірювальної техніки для визначення параметрів газопилового потоку і концентрацій забруднюючих речовин в атмосферному повітрі та викидах стаціонарних і пересувних джерел;

9. здійснювати контроль за проектуванням, будівництвом і експлуатацією споруд, устаткування та апаратури для очищення газопилового потоку від забруднюючих речовин і зниження впливу фізичних та біологічних факторів, оснащення їх засобами вимірювальної техніки, необхідними для постійного контролю за ефективністю очищення, дотриманням нормативів гранично допустимих викидів забруднюючих речовин і рівнів впливу фізичних та біологічних факторів та інших вимог законодавства в галузі охорони атмосферного повітря;

10. своєчасно і в повному обсязі сплачувати збори за забруднення навколишнього природного середовища та погіршення якості природних ресурсів відповідно до закону.

Виконання заходів щодо охорони атмосферного повітря не повинно призводити до забруднення грунтів, вод та інших природних об'єктів.

Законодавством України встановлюються нормативи використання природних ресурсів та інші екологічні нормативи.

Екологічні нормативи встановлюють гранично допустимі викиди та скиди у навколишнє природне середовище забруднюючих хімічних речовин, рівні допустимого шкідливого впливу на нього фізичних та біологічних факторів

З метою відвернення і зменшення забруднення атмосферного повітря транспортними та іншими пересувними засобами і установками та впливу пов’язаних з ними фізичних факторів здійснюються:

1. розроблення та виконання комплексу заходів щодо зниження викидів, знешкодження шкідливих речовин і зменшення фізичного впливу під час проектування, виробництва, експлуатації та ремонту транспортних та інших пересувних засобів і установок;

2. переведення транспортних та інших пересувних засобів і установок на менш токсичні види палива;

3. раціональне планування та забудова населених пунктів з дотриманням нормативно визначеної відстані до транспортних шляхів;

4. виведення з густонаселених житлових кварталів за межі міста транспортних підприємств, вантажного транзитного автомобільного транспорту;

5. обмеження в'їзду автомобільного транспорту та інших транспортних засобів та установок у сельбищні, курортні, лікувально-оздоровчі, рекреаційні та природно-заповідні зони, місця масового відпочинку та туризму;

6. поліпшення стану утримання транспортних шляхів і вуличного покриття;

7. впровадження в містах автоматизованих систем регулювання дорожнього руху;

8. удосконалення технологій транспортування і зберігання палива, забезпечення постійного контролю за якістю палива на нафтопереробних підприємствах та автозаправних станціях;

9. впровадження та вдосконалення діяльності контрольно-регулювальних і діагностичних пунктів та комплексних систем перевірки нормативів екологічної безпеки транспортних та інших пересувних засобів і установок.

Проектування, виробництво та експлуатація транспортних та інших пересувних засобів і установок, вміст забруднюючих речовин у відпрацьованих газах яких перевищує нормативи або рівні впливу фізичних факторів, забороняються.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було здійснено аналіз методів дослідження математичної моделі, в результаті якого був здійснений опис предметної галузі, дослідження існуючих методів та критеріїв для знаходження важелів моделі, знаходження аномальних значень та оцінювання доцільності математичної моделі. Також була здійснене удосконалення математичної моделі, шляхом переходу від лінійної моделі до нелінійної, через зворотне десяткове логарифмічне перетворення. В результаті отриманої удосконаленої математичної моделі критерій середньої величини відносної похибки став покращений на 0.083 та рівень прогнозування на 0.038.

Після розробки проекту програмного забезпечення було розроблене програмне забезпечення для оцінювання програмних проектів створених мовою Python, що задовольняє поставленим вимогам технічного завдання.

В результаті виконання роботи було розроблене програмне забезпечення у вигляді веб-додатку, який задовольняє вимогам, які поставлені у технічному завданні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коэффициент регрессии (Coefficient of regression) [Електронний ресурс]. Режим доступу: https://wiki.loginom.ru/articles/coefficient-of-regression.html

2. Линейная регрессия (Linear regression) [Електронний ресурс]. Режим доступу: https://wiki.loginom.ru/articles/linear-regression.html

3. Метод наименьших квадратов (МНК) [Електронний ресурс]. Режим доступу: http://www.cleverstudents.ru/articles/mnk.html

4. Полиномиальная регрессия - Polynomial regression [Електронний ресурс]. Режим доступу: https://ru.qaz.wiki/wiki/Polynomial_regression

5. Transformations of Variables [Електронний ресурс]. Режим доступу: https://stattrek.com/regression/linear-transformation.aspx?tutorial=reg

6. Метод наименьших квадратов (Least-Squares method) [Електронний ресурс]. Режим доступу: https://wiki.loginom.ru/articles/least-squares-method.html

7. Простая линейная регрессия [Електронний ресурс]. – Режим доступу: https://baguzin.ru/wp/prostaya-linejnaya-regressiya/

8. Тема 1. Обзор понятий и формулы вычисления: ковариации, дисперсии и корреляции [Електроний ресурс]. Режим доступу: https://kpfu.ru/portal/docs/F149578393/lekcii.27.pdf

9. Среднеквадратическое отклонение (Mean square deviation) [Електронний ресурс]. Режим доступу: https://wiki.loginom.ru/articles/mean-square-deviation.html

10. Определение среднего значения, вариации и формы распределения. Описательные статистики [Електронний ресурс]. Режим доступу: https://baguzin.ru/wp/opredelenie-srednego-znacheniya-varia/

11. Нормализация данных (Data normalization) [Електронний ресурс]. Режим доступу: https://wiki.loginom.ru/articles/data-normalization.html

12. Обнаружение выбросов и влиятельных наблюдений в регрессионном анализе [Електроний ресурс]. Режим доступу: http://forum.disser.ru/index.php?act=attach&id=284&type=post

13. Расстояния Махаланобиса [Електронний ресурс]. Режим доступу: http://statistica.ru/glossary/general/rasstoyaniya-makhalanobisa/

14. Відстань Кука [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Відстань_Кука

15. Критерий Фишера (F-test) [Електронний ресурс] Режим доступу: https://wiki.loginom.ru/articles/f-test.html

16. Коэффициент детерминации (Coefficient of determination) [Електронний ресурс] Режим доступу: https://wiki.loginom.ru/articles/coefficient-of-determination.html

17. Коэффициент корреляции (Correlation coefficient) [Електронний ресурс] Режим доступу: https://wiki.loginom.ru/articles/correlation-coefficient.html

18. Confidence Interval for the Mean Response [Електронний ресурс] Режим доступу: https://online.stat.psu.edu/stat501/lesson/3/3.2

19. Machine learning: an introduction to mean squared error and regression lines [Електронний ресурс] Режим доступу: https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

1 Вступ

У зв’язку з важким процесом створення правильної архітектури проектів і трудомістким процесом їх розробки та підтримки необхідно поліпшити цей процес шляхом створення програмного забезпечення, які дають змогу оцінити розмір програмних проектів, шляхом удосконалення математичної моделі. Удосконалення математичної моделі дозволить проаналізувати та спрогнозувати усі аспекти розроблюваного програмного проекту, що допоможе при його розробці та у продовж подальшої підтримки вже розробленого проекту.

Назва програмного забезпечення: «Удосконалення математичної моделі для оцінювання розміру програмних проектів, що створені мовою Python».

Створене програмне забезпечення повинно використовувати веб-інтерфейс, що забезпечує роботу з даними, їх обчислення та знаходження лінійної моделі, та її удосконалення. Користувач системи здійснює заносить дані через форму інтерфейсу і використовує методи в системі, через кнопки, які обчислюють необхідні дані та виводять результати на екран.

2 Підстави для розробки

Підставою для розробки цього програмного забезпечення є відсутність повної реалізованої системи оцінювання розміру програмних проектів написаних мовою Python, тому розробка даної системи є доцільною.

3 Призначення розробки

3.1 Функціональне призначення

Функціональнім призначенням програми є знаходження математичної моделі, їх удосконалення, розрахування необхідних обчислень, вивід результатів на екран та на графіки на базі завантажених даних користувачем.

3.2 Експлуатаційне призначення

Програмне забезпечення повинно використовуватись користувачами на персональних комп’ютерах у веб-браузері.

4 Вимоги до програми або програмного виробу

4.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинне забезпечувати можливість виконання перерахованих нижче функцій:

– на основі завантажених даних у JSON файлах зі структурою у вигляді JSON об’єктів, надати дані з критеріями (кількість методів, вага);

– на основі завантажених даних зробити обчислення розподілу даних;

– на основі завантажених даних зробити обчислення лінійної регресії;

– на основі завантажених даних зробити обчислення нелінійної регресії;

– на основі завантажених даних зробити розрахунок метрик рівнянь регресії;

– на основі отриманих даних зробити вивід обчислень регресії на екран;

– на основі отриманих даних зробити вивід обчислень регресії на графік;

– надати можливість розрахувати викиди;

– на основі отриманих даних зробити розрахунок інтервалів для введеної кількості методів.

4.2 Вимоги до організації вхідних та вихідних даних

Введення вхідних даних здійснюється за допомогою завантаження файлу з операційної системи. Усі інші перераховані функціональні можливості виконуються за допомогою кнопок інтерфейсу.

4.3 Вимоги до надійності

4.3.1 Вимоги до забезпечення надійного функціонування програми

Стійким функціонаування програмного забезпечення має набути шляхом забезпеченням виконання сукупності організаційно-технічних задів, а саме:

1) організацією безперебійного живлення для технічних засобів, що застосовуються при використані даного ПЗ;

2) використанням чистого та ліцензійного програмного забезпечення.

4.3.2 Час на відновлення після відмови

Час на відновлення після відмови, або збою, що може бути викликано проблемами з електроживленням на сервері, або проблемами з операційною системою, що встановлена на сервері, час відмов в таких випадках не повинен перевищувати п’яти хвилин.

Час на відновлення після несправності технічних засобів, або у випадку повної несправності системи на сервері, не повинно бути більше години. На протязі цієї години необхідні збої повинні бути у процесі відновлені, а проблеми з системою – усунені шляхом переустановленням системи.

4.3.3 Відмови через некоректність дій користувача

Відмови через дії користувачів можливі, якщо користувач при завантаженні вхідних файлів обирає файли неправильного типу, або якщо обрані файли мають неправильну структуру, яка необхідно для роботи системи. Але усі відмови мають попереджувальний для користувача характер, що не приводить до збоїв системи.

4.4 Умови експлуатації

4.4.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації повинні задовольняти вимогам до технічних засобів:

– температура навколишнього повітря, — від плюс 10°С до плюс 40°С;
– атмосферний тиск, кПа — 101.3;
– відносна вологість повітря при 20 ° С — 45%.

4.4.2 Вимоги до видів обслуговування

Програмне забезпечення не потребує обслуговування.

4.4.3 Вимоги до чисельності та кваліфікації персоналу

Мінімальною кількістю персоналу необхідною для функціонування програми є одна людина, що має бути кінцевим користувачем програми.

Користувач повинен мати навички роботи в мережі інтернет, та мати змогу працювати з веб-інтерфейсом у браузері.

4.5 Вимоги до складу і параметрів технічних засобів

В склад технічних засобів повинен входити комп’ютер з мінімальними характеристиками:

– 2 ГБ оперативної пам’яті;
– 16 GB вільного дискового простору;
– 32-разрядний процесор (x86), 1ГГц, 1 ядро
– підключення до інтернету.

4.6 Вимоги до інформаційної і програмної сумісності

4.6.1 Вимоги до інформаційних структур і методів розв’язку

Вимогою для вхідних файлів стосується типу файлів та структурою, яка використовується цим типом файлів:

– файли повинні мати тип даних «.json»;

– структура файлів має бути JSON object, що має дані вигляду «ключ-значення», та має сувору структуру у фігурних дужках, порушення якої приводить до збоїв програми, що використовує такий тип даних.

4.6.2 Вимоги до вихідних кодів та мов програмування

Вихідний код програми повинен бути реалізований на мові Javascript. У якості інтегрованого середовища для розробки програмного забезпечення повинне бути використане PHPStorm, або WebStorm.

4.6.3 Вимоги до програмних засобів, що використовуються програмою

Системне програмне забезпечення, що використовується для використання розробленого програмного забезпечення, шляхом взаємодії з веб-інтерфейсу у веб-браузері, повинні бути ліцензійними, або безкоштовними.

4.6.4 Вимоги до захисту інформації та програм

Вимог до захисту інформації та програм не висувається.

4.7 Вимоги до маркування

Вимог до маркування не висувається.

4.8 Вимоги до транспортування та збереження

Вимоги до транспортування та збереження програми не пред'являється.

5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програма та методика випробовування
- текст програмних модулів
- опис програми
- інструкція до користувача

6 Техніко-економічні показники

Техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Стадії та етапи розробки представлено в таблиці А.1.

Таблиця А.1 Стадії та етапи розробки

№	Стадії розробки	Етапи розробки	Зміст робіт по етапах	Строки виконання
1	Технічне завдання	розробка, узгодження та затвердження технічного завдання	Постановка задачі; визначення та уточнення вимог до технічних засобів; визначення вимог до програми; визначення стадій, етапів і термінів розробки програми і документації на неї; узгодження і затвердження технічного завдання.	Початок: 10.09.20 Закінчення: 30.09.20
2	Ескізний проект	Розробка та затвердження ескізного проекту	Виявлення вимог, концепція майбутньої системи, складання ескізної частини програмного продукту	Початок: 09.10.20 Закінчення: 19.10.20
3	Технічний проект	Розробка та затвердження технічного проекту	Загальні системні вирішення, алгоритми задач, оцінювання економічної ефективності автоматизованої системи та перелік заходів підготовки об’єкта для провадження	Початок: 20.10.20 Закінчення: 29.10.20
4	Робочий проект	Розробка та затвердження робочого проекту	Деталізовані загальносистемні проекті рішення, програми та інструкції для розв’язку завдань, розробка програмного забезпечення.	Початок: 01.11.20 Закінчення: 15.11.20

8 Порядок контролю та приймання

Для контролю та приймання повинен бути представлений опис програми, а також програма і методика тестування.

Порядок контролю та приймання даного програмного продукту здійснюється представниками замовника за участі представника розробника згідно програмі та методиці тестування.

Якщо програма не пройшла тестування, виконавець зобов’язаний виправити помилки та недоробки в зазначений термін, але не більше 1 місяця з дня випробування.

За результатами приймання складається акт, який підписується представниками замовника та представниками розробника. Потім акт затверджується керівниками організації-замовника та організації-розробника.

У випадку знаходження помилок при прийманні програмного продукту, складається акт про виявлення помилок, який підписується представниками замовника та представниками розробника та керівниками організації-замовника та організації-розробника. Розробник повинен виправити зазначені зауваження протягом 1-го місяця і повідомити про повторне проведення перевірки, але не пізніше ніж за 2 тижня до початку приймання програмного продукту.

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

Лістинг коду програми

```
(this["webpackJsonpchart-app"] = this["webpackJsonpchart-app"] || []).push([[0], {
    25: function(t, e, n) {
        t.exports = {
            RegressionApp: "RegressionApp_RegressionApp__2OT5P",
            Input: "RegressionApp_Input__1vo-m",
            Output: "RegressionApp_Output__2C305",
            RegressionControls: "RegressionApp_RegressionControls__32ngo",
            OutputAndRegressionCBox: "RegressionApp_OutputAndRegressionCBox__1WMOw",
            ChartContainer: "RegressionApp_ChartContainer__16xUQ"
        }
    },
    35: function(t, e, n) {
        t.exports = {
            RegressionControls: "RegressionControls_RegressionControls__3q0mK",
            Button: "RegressionControls_Button__12R9s"
        }
    },
    38: function(t, e, n) {
        t.exports = {
            InputRegressionData: "InputRegressionData_InputRegressionData__3cv6C",
            textarea: "InputRegressionData_textarea__2TLcC",
            Button: "InputRegressionData_Button__1XHw-"
        }
    },
    53: function(t, e, n) {
        t.exports = {
            Controls: "Controls_Controls__1E4qe",
            Button: "Controls_Button__o2mVX"
        }
    },
    63: function(t, e, n) {
        t.exports = {
            "MuiInputBase-formControl": "OutputData_MuiInputBase-formControl__1dcLk",
            "MuiInputBase-multiline": "OutputData_MuiInputBase-multiline__U0oas"
        }
    },
    81: function(t, e, n) {},
    82: function(t, e, n) {},
    93: function(t, e) {},
    96: function(t, e, n) {
        "use strict";
        n.r(e);
        var a = n(5)
          , r = n(0)
          , o = n.n(r)
          , c = n(10)
          , i = n.n(c)
          , s = (n(81),
        n(82),
        n(19))
          , u = n(8)
          , l = n(127)
          , d = n(128)
          , b = n(125)
          , f = n(38)
          , p = n.n(f)
```

```
          , j = n(43)
          , O = n.n(j);
        console.log("styles", p.a);
        var h, g = function(t) {
            var e = t.onLoadData
              , n = t.className
              , o = Object(r.useState)("")
              , c = Object(u.a)(o, 2)
              , i = c[0]
              , s = c[1]
              , f = Object(r.useCallback)((function(t) {
                console.log("update value"),
                s(t.currentTarget.value)
            }
            ), [s])
              , j = Object(r.useCallback)((function() {
                e(i)
            }
            ), [e, i]);
            return Object(a.jsxs)(l.a, {
                className: O()(p.a.InputRegressionData, n),
                children: [Object(a.jsx)(d.a, {
                    onClick: j,
                    variant: "contained",
                    disabled: !i,
                    color: "primary",
                    className: p.a.Button,
                    children:
"\u0417\u0430\u0432\u0430\u043d\u0442\u0430\u0436\u0438\u0442\u0438"
                }), Object(a.jsx)(b.a, {
                    className: p.a.textarea,
                    fullWidth: !0,
                    label: "\u0412\u0445\u0456\u0434\u043d\u0456 \u0434\u0430\u043d\u0456",
                    multiline: !0,
                    rows: 4,
                    placeholder: "Enter input data",
                    variant: "outlined",
                    onChange: f
                })]
            })
        }, v = n(63), m = n.n(v), C = function(t) {
            var e = t.value
              , n = t.className
              , o = Object(r.useCallback)((function(t) {
                t.preventDefault()
            }
            ), []);
            return Object(a.jsx)(l.a, {
                className: n,
                children: Object(a.jsx)(b.a, {
                    fullWidth: !0,
                    label:  "\u0406\u043d\u0444\u043e\u0440\u043c\u0430\u0446\u0456\u044f",
                    "aria-readonly": !0,
                    multiline: !0,
                    rows: 10,
                    variant: "outlined",
                    onKeyPress: o,
                    value: e,
                    className: m.a.OutputTextInput,
                    style: {
                        height: 200
                    }
                })
            })
```

```
}, x = n(25), R = n.n(x), _ = n(12), y = n(124), k = n(34), w = function(t, e) {
    var n = e.map((function(t) {
        return t[0]
    }
    ))
      , a = y.a.apply(void 0, Object(_.a)(n))
      , r = y.c.apply(void 0, Object(_.a)(n));
    return [{
        x: r,
        y: t.fx(r)
    }, {
        x: a,
        y: t.fx(a)
    }]
}, B = function(t, e) {
    return t.map((function(n, a) {
        var r = Object(_.a)(t);
        return e(r.filter((function(t, e) {
            return e !== a
        }
        )), n, a)
    }
    ))
}, S = function(t) {
    return Math.sqrt(t.reduce((function(t, e) {
        return t + e * e
    }
    ), 0) / (t.length - 2))
}, I = function(t) {
    var e = y.b.apply(void 0, Object(_.a)(t));
    return t.map((function(t) {
        return Math.pow(e - t, 2)
    }
    )).reduce((function(t, e) {
        return t + e
    }
    ), 0) / t.length
}, N = function(t, e, n, a) {
    return 1 / n + Math.pow(t - e, 2) / ((n - 1) * a)
}, E = function(t) {
    return 4 / t
}, T = function(t) {
    var e = function(t) {
        return Math.log10(t)
    };
    return t.map((function(t) {
        return e(t)
    }
    ))
}, D = function(t) {
    var e = arguments.length > 1 && void 0 !== arguments[1] ? arguments[1] : 3
      , n = t.getInputData();
    console.log("inputData", n);
    var a = n.map((function(t) {
        return t[0]
    }
    ))
      , r = a.length
      , o = []
      , c = t.getResidual();
    console.log("residual", c);
    B(n, (function(t, e) {
        var n = Object(k.linear)(t);
        return o.push(n),
```

```
            e[1] - n.predict(e[0])[1]
        }
        ));
        var i = S(c)
          , s = Object(y.b)(a)
          , u = I(a)
          , l = (E(r),
        a.map((function(t) {
            return N(t, s, r, u)
        }
        )))
          , d = l.map((function(t) {
            return i * Math.sqrt(1 - t)
        }
        ))
          , b = (c.map((function(t, e) {
            return t / d[e]
        }
        )),
        B(n, (function(t, e, n) {
            var a = S(t.map((function(t, e) {
                return t[1] - o[n].predict(t[0])[1]
            }
            )));
            return c[n] / (a * Math.sqrt(1 - l[n]))
        }
        )))
          , f = [];
        return console.log("externallyRt", b),
        b.forEach((function(t, n) {
            Math.abs(t) > e && f.push(n)
        }
        )),
        f
    }, L = n(51), M = n(52), A = function() {
        function t(e) {
            var n = this;
            Object(L.a)(this, t),
            this.data = void 0,
            this.linear = void 0,
            this.getInputData = function() {
                return Object(_.a)(n.data)
            }
            ,
            this.getR2 = function() {
                return n.linear.r2
            }
            ,
            this.data = Object(_.a)(e),
            this.linear = Object(k.linear)(this.data, {
                precision: 5
            })
        }
        return Object(M.a)(t, [{
            key: "getRtiThreshold",
            value: function() {
                return 3
            }
        }, {
            key: "fx",
            value: function(t) {
                return console.log("call fx original!"),
                this.linear.predict(t)[1]
            }
```

```
            }, {
                key: "getResidual",
                value: function() {
                    var t = this;
                    return this.data.map((function(e) {
                        return e[1] - t.fx(e[0])
                    }
                    ))
                }
            }, {
                key: "getStringForm",
                value: function() {
                    return this.linear.string
                }
            }, {
                key: "getMSE",
                value: function() {
                    return                          y.d.apply(void                          0,
Object(_.a)(this.getResidual().map((function(t) {
                        return t * t
                    }
                    )))) / this.data.length
                }
            }, {
                key: "getConfidenceInterval",
                value: function() {
                    return null
                }
            }]),
            t
        }(), P = n(64);
        !function(t) {
            t.SCATTER               =               "\u0420\u043e\u0437\u043a\u0438\u0434
\u0434\u0430\u043d\u0438\u0445",
            t.LINE = "\u041b\u0456\u043d\u0456\u0439\u043d\u0430",
            t.OUTLIERS = "\u0412\u0438\u043a\u0438\u0434\u0438",
            t.QBORDER_LOW_1 = "Q Border low 1",
            t.QBORDER_LOW_2 = "Q Border low 2",
            t.QBORDER_HIGHT_1 = "Q Border hight 1",
            t.QBORDER_HIGHT_2 = "Q Border hight 2"
        }(h || (h = {})),
        window.naxer = [];
        var F = function(t) {
            var e = t.linePoints
              , n = t.scatterPoints
              , o = t.qBorderLine
              , c = t.outliersPoints
              , i = t.lineName
              , s = Object(r.useState)(null)
              , l = Object(u.a)(s, 2)
              , d = l[0]
              , b = l[1]
              , f = Object(r.useState)()
              , p = Object(u.a)(f, 2)
              , j = p[0]
              , O = p[1];
            return Object(r.useEffect)((function() {
                var t = null === d || void 0 === d ? void 0 : d.getContext("2d");
                if (t) {
                    var e = new P.Chart(t,{
                        type: "scatter",
                        data: {
                            datasets: [{
                                pointBackgroundColor: "#106CC8",
```

```
                                label: h.SCATTER,
                                backgroundColor: "#106CC8",
                                data: []
                            }, {
                                pointBackgroundColor: "#9517f3",
                                label: h.OUTLIERS,
                                backgroundColor: "#9517F3",
                                data: [],
                                pointStyle: "cross",
                                borderColor: "#9517F3",
                                pointRadius: 6
                            }, {
                                type: "line",
                                label: i,
                                data: [],
                                fill: !1,
                                borderColor: "#106CC8"
                            }]
                        },
                        options: {
                            responsive: !0,
                            maintainAspectRatio: !1,
                            tooltips: {
                                callbacks: {
                                    label: function(t, e) {
                                        console.log("tooltipItem", t),
                                        console.log("data", e);
                                        var n = e.datasets[t.datasetIndex].label || "";
                                        if (n && (n += ": "),
                                        e.datasets && void 0 !== t.datasetIndex && void
0 !== t.index) {
                                            var a = e.datasets[t.datasetIndex].data;
                                            if (Array.isArray(a) && a[t.index]) {
                                                var r = a[t.index];
                                                if (console.log("item", r),
                                                r.label && r.classCount)
                                                    return window.naxer.push(r.label),
                                                    "".concat(r.label, ": [").concat(t.x,
", ").concat(r.classCount, "]")
                                            }
                                        }
                                        return n
                                    }
                                }
                            }
                        }
                    });
                    O(e)
                }
            }
            ), [O, d, i]),
            Object(r.useEffect)((function() {
                j && j.data.datasets && (j.data.datasets.forEach((function(t) {
                    switch (t.label) {
                    case h.SCATTER:
                        t.data = n;
                        break;
                    case h.OUTLIERS:
                        t.data = c;
                        break;
                    case i:
                        t.data = e
                    }
                }
```

```
        )),
        j.update())
    }
    ), [n, e, c, j]),
    Object(r.useEffect)((function() {
        o && j && j.data.datasets && (j.data.datasets.forEach((function(t) {
            if (t.label)
                switch (t.label) {
                case h.QBORDER_HIGHT_1:
                    t.data = o.highBorder1;
                    break;
                case h.QBORDER_HIGHT_2:
                    t.data = o.highBorder2
                }
        }
        )),
        j.update())
    }
    ), [o, j]),
    Object(a.jsx)("canvas", {
        ref: b
    })
}
  , Q = function(t) {
    var e = t.scatter
      , n = t.line
      , o = t.outliers
      , c = t.lineName
      , i = Object(r.useState)([])
      , s = Object(u.a)(i, 2)
      , l = s[0]
      , d = s[1]
      , b = Object(r.useState)([])
      , f = Object(u.a)(b, 2)
      , p = f[0]
      , j = f[1]
      , O = Object(r.useState)([])
      , h = Object(u.a)(O, 2)
      , g = h[0]
      , v = h[1];
    return Object(r.useEffect)((function() {
        d(e)
    }
    ), [e, d]),
    Object(r.useEffect)((function() {
        j(n)
    }
    ), [n, j]),
    Object(r.useEffect)((function() {
        v(o)
    }
    ), [o, v]),
    Object(a.jsx)(F, {
        outliersPoints: g,
        linePoints: p,
        scatterPoints: l,
        lineName: c
    })
}
  , H = n(53)
  , q = n.n(H)
  , U = function(t) {
    var e = t.fixDistribution
      , n = t.onTransformData
```

```
              , r = t.onCalcLinear;
            t.onFindOutliers;
            return Object(a.jsxs)(l.a, {
                classNam: q.a.Controls,
                children: [Object(a.jsx)(d.a, {
                    className: q.a.Button,
                    variant: "contained",
                    disabled: !e,
                    color: "primary",
                    onClick: r,
                    children:
"\u0420\u043e\u0437\u0440\u0430\u0445\u0443\u0432\u0430\u0442\u0438
\u0440\u043e\u0437\u043f\u043e\u0434\u0456\u043b \u0434\u0430\u043d\u0438\u0445"
                }), Object(a.jsx)(d.a, {
                    className: q.a.Button,
                    variant: "contained",
                    disabled: !e,
                    color: "primary",
                    onClick: n,
                    children:
"\u041f\u0435\u0440\u0435\u0442\u0432\u043e\u0440\u0435\u043d\u043d\u044f
\u0434\u0430\u043d\u0438\u0445"
                })]
            })
        }
          , W = n(35)
          , G = n.n(W)
          , J = function(t) {
            var e = t.className
              , n = t.disabled
              , r = t.calcLinear
              , o = t.findOutliersLinear
              , c = t.calcPower
              , i = t.findOutliersPower;
            return Object(a.jsxs)(l.a, {
                className: O()(G.a.RegressionControls, e),
                children: [Object(a.jsx)(d.a, {
                    className: G.a.Button,
                    variant: "contained",
                    disabled: n,
                    color: "primary",
                    onClick: r,
                    children:
"\u0420\u043e\u0437\u0440\u0430\u0445\u0443\u0432\u0430\u043d\u043d\u044f
\u043b\u0456\u043d\u0456\u0439\u043d\u043e\u0457
\u0440\u0435\u0433\u0440\u0435\u0441\u0456\u0457"
                }), Object(a.jsx)(d.a, {
                    className: G.a.Button,
                    variant: "contained",
                    disabled: n,
                    color: "primary",
                    onClick: o,
                    children:                                   "\u041f\u043e\u0448\u0443\u043a
\u0432\u0438\u043a\u0438\u0434\u0456\u0432                                  \u0434\u043b\u044f
\u043b\u0456\u043d\u0456\u0439\u043d\u043e\u0457"
                }), Object(a.jsx)(d.a, {
                    className: G.a.Button,
                    variant: "contained",
                    disabled: n,
                    color: "primary",
                    onClick: c,
                    children:
"\u0420\u043e\u0437\u0440\u0430\u0445\u0443\u043d\u043e\u043a
```

```
\u043d\u0435\u043b\u0456\u043d\u0456\u0439\u043d\u043e\u0457
\u0440\u0435\u0433\u0440\u0435\u0441\u0456\u0457"
                }), Object(a.jsx)(d.a, {
                    className: G.a.Button,
                    variant: "contained",
                    disabled: n,
                    color: "primary",
                    onClick: i,
                    children:                                "\u041f\u043e\u0448\u0443\u043a
\u0432\u0438\u043a\u0438\u0434\u0456\u0432                                \u0434\u043b\u044f
\u043d\u0435\u043b\u0456\u043d\u0456\u0439\u043d\u043e\u0457"
                })]
            })
        }
          , K = n(66)
          , X = n(65)
          , V = function(t) {
            Object(K.a)(n, t);
            var e = Object(X.a)(n);
            function n(t) {
                return Object(L.a)(this, n),
                e.call(this, t.map((function(t) {
                    var e = Object(u.a)(t, 2)
                      , n = e[0]
                      , a = e[1];
                    return [Math.log10(n), Math.log10(a)]
                }
                )))
            }
            return Object(M.a)(n, [{
                key: "fx",
                value: function(t) {
                    return console.log("call fx power!"),
                    Math.pow(10, A.prototype.fx.call(this, Math.log10(t)))
                }
            }, {
                key: "getRtiThreshold",
                value: function() {
                    return 2
                }
            }, {
                key: "getResidual",
                value: function() {
                    var t = this;
                    return this.data.map((function(e) {
                        var n = Object(u.a)(e, 2)
                          , a = n[0];
                        return n[1] - A.prototype.fx.call(t, a)
                    }
                    ))
                }
            }, {
                key: "getStringForm",
                value: function() {
                    return                                    "y                                    =
10^".concat(A.prototype.getStringForm.call(this).replace("x",  "log(x)").replace("y  =  ",
""))
                }
            }]),
            n
        }(A)
          , z = (n(95).jStat,
        function(t, e) {
            return t.map((function(t) {
```

```
        return {
            x: t.count,
            y: t.bytes,
            label: t.name,
            classCount: e.get(t.name) || t.count
        }
    }
    ))
}
)
  , Y = function() {
    var t = Object(r.useState)("")
      , e = Object(u.a)(t, 2)
      , n = e[0]
      , o = e[1]
      , c = Object(r.useState)([])
      , i = Object(u.a)(c, 2)
      , d = i[0]
      , b = i[1]
      , f = Object(r.useState)([])
      , p = Object(u.a)(f, 2)
      , j = p[0]
      , O = p[1]
      , h = Object(r.useState)([])
      , v = Object(u.a)(h, 2)
      , m = v[0]
      , x = v[1]
      , _ = Object(r.useState)([])
      , y = Object(u.a)(_, 2)
      , k = y[0]
      , B = y[1]
      , S = Object(r.useState)([])
      , I = Object(u.a)(S, 2)
      , N = I[0]
      , E = I[1]
      , L = Object(r.useState)([])
      , M = Object(u.a)(L, 2)
      , P = M[0]
      , F = M[1]
      , H = Object(r.useState)()
      , q = Object(u.a)(H, 2)
      , W = q[0]
      , G = q[1]
      , K = Object(r.useState)()
      , X = Object(u.a)(K, 2)
      , Y = X[0]
      , Z = X[1]
      , $ = Object(r.useState)()
      , tt = Object(u.a)($, 2)
      , et = tt[0]
      , nt = tt[1]
      , at = Object(r.useState)(new Map)
      , rt = Object(u.a)(at, 2)
      , ot = rt[0]
      , ct = rt[1]
      , it = Object(r.useState)()
      , st = Object(u.a)(it, 2)
      , ut = (st[0],
    st[1],
    Object(r.useCallback)((function() {
        o(""),
        x([]),
        E([])
    }
```

```
            ), [o, Z, x, E]))
              , lt = Object(r.useCallback)((function(t) {
                o((function(e) {
                    return "".concat(e, "\n").concat(t)
                }
                ))
            }
            ), [o]);
            Object(r.useEffect)((function() {
                Y && b(z(Y, ot))
            }
            ), [Y, b, ot]),
            Object(r.useEffect)((function() {
                et && O(z(et, ot))
            }
            ), [et, b, ot]);
            var dt = Object(r.useCallback)((function(t) {
                try {
                    var e = JSON.parse(t);
                    console.log("data length", e.data.length);
                    var n = e.data.filter((function(t) {
                        return t.count
                    }
                    )).filter((function(t) {
                        return !1 === [""].includes(t.name)
                    }
                    )).map((function(t) {
                        return Object(s.a)(Object(s.a)({}, t), {}, {
                            bytes: t.bytes
                        })
                    }
                    ));
                    G(n),
                    ct(new Map(n.map((function(t) {
                        return [t.name, t.count]
                    }
                    )))),
                    Z(n),
                    nt(n),
                    ut()
                } catch (a) {
                    ut(),
                    o("Occur error while trying to parse json. Try to check data for
correctness.")
                }
            }
            ), [ut, G, ct, nt])
              , bt = Object(r.useCallback)((function() {
                Y && Z(Y.map((function(t) {
                    return Object(s.a)(Object(s.a)({}, t), {}, {
                        count: T([t.count])[0],
                        bytes: T([t.bytes])[0]
                    })
                }
                )))
            }
            ), [Y, Z])
              , ft = Object(r.useCallback)((function() {
                if (Y) {
                    lt("Try to find linear regression.");
                    var t = Y.map((function(t) {
                        return [t.count, t.bytes]
                    }
                    ))
```

```
                      , e = new A(t);
                  lt("Linear regression have a form: ".concat(e.getStringForm())),
                  lt("R2: ".concat(e.getR2())),
                  E(w(e, t))
              }
          }
          ), [lt, Y])
            , pt = Object(r.useCallback)((function(t, e, n, a) {
              console.log("call find out!");
              var r, o = function(e) {
                  var n = e.map((function(t) {
                      return [t.count, t.bytes]
                  }
                  ));
                  return new t(n)
              }, c = [], i = [], s = a;
              do {
                  r = o(s),
                  (c = D(r, r.getRtiThreshold())).length > 0 && (s = s.filter((function(t,
e) {
                      return !c.includes(e) || (i.push(t),
                      !1)
                  }
                  ))),
                  console.log("step outliers", c)
              } while (0 !== c.length);console.log("Found outliers", i),
              e(z(i, ot)),
              n(a.filter((function(t) {
                  return !i.includes(t)
              }
              )))
          }
          ), [Z, ot])
            , jt = Object(r.useCallback)((function() {
              return Y && pt(A, x, Z, Y)
          }
          ), [pt, x, Y, Z])
            , Ot = Object(r.useCallback)((function() {
              return et && pt(V, B, nt, et)
          }
          ), [pt, nt, et])
            , ht = Object(r.useCallback)((function() {
              if (et) {
                  lt("Try to find power regression.");
                  var t = et.map((function(t) {
                      return [t.count, t.bytes]
                  }
                  ))
                    , e = new V(t);
                  lt("Power regression have a form: ".concat(e.getStringForm())),
                  lt("R2: ".concat(e.getR2())),
                  F(w(e, t))
              }
          }
          ), [lt, et, F]);
          return Object(a.jsxs)(l.a, {
              padding: "20px",
              className: R.a.RegressionApp,
              children: [Object(a.jsx)(g, {
                  className: R.a.Input,
                  onLoadData: dt
              }), Object(a.jsx)(U, {
                  onTransformData: bt,
                  fixDistribution: !!W,
```

```
                    onCalcLinear: ft,
                    onFindOutliers: jt
                }), Object(a.jsxs)(l.a, {
                    className: R.a.OutputAndRegressionCBox,
                    children: [Object(a.jsx)(C, {
                        className: R.a.Output,
                        value: n
                    }), Object(a.jsx)(J, {
                        calcLinear: ft,
                        findOutliersLinear: jt,
                        calcPower: ht,
                        disabled: !W,
                        className: R.a.RegressionControls,
                        findOutliersPower: Ot
                    })]
                }), Object(a.jsxs)(l.a, {
                    position: "relative",
                    className: R.a.ChartContainer,
                    height: "47vh",
                    width: "96vw",
                    children: [Object(a.jsx)(l.a, {
                        width: "48vw",
                        children: Object(a.jsx)(Q, {
                            line: N,
                            scatter: d,
                            outliers: m,
                            lineName: "\u041b\u0456\u043d\u0456\u0439\u043d\u0430"
                        })
                    }), Object(a.jsx)(l.a, {
                        width: "48vw",
                        children: Object(a.jsx)(Q, {
                            line: P,
                            scatter: j,
                            outliers: k,
                            lineName: "\u0421\u0442\u0435\u043f\u0456\u043d\u043d\u0430"
                        })
                    })]
                })]
            })
        };
        var Z = function() {
            return Object(a.jsx)("div", {
                className: "App",
                children: Object(a.jsx)(Y, {})
            })
        }
          , $ = function(t) {
            t && t instanceof Function && n.e(3).then(n.bind(null, 131)).then((function(e)
{
                var n = e.getCLS
                  , a = e.getFID
                  , r = e.getFCP
                  , o = e.getLCP
                  , c = e.getTTFB;
                n(t),
                a(t),
                r(t),
                o(t),
                c(t)
            }
            ))
        };
        i.a.render(Object(a.jsx)(o.a.StrictMode, {
            children: Object(a.jsx)(Z, {})
```

```
        }), document.getElementById("root")),
        $()
    }
}, [[96, 1, 2]]]);
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1 Загальні відомості

Найменування даної програми – "Удосконалення вибору цільової аудиторії приватного медичного центру модифікованим методом аналізу ієрархій".

Програмне забезпечення, що розроблялося, повинно працювати у вигляді web-додатка під управлінням двох розповсюджених операційних систем Windows і Unix. Отже, для його роботи потрібен веб-сервер і база даних.

В якості web-сервера використовується Apache Tomcat, що вільно поширюється й постійно оновлюється. Існують версії даної програми як для операційних систем на базі Win32/64, так і для POSIX-сумісних (Linux, FreeBSD).

Web-сервер Apache Tomcat може обслуговувати віртуальні хости, підтримує CGI-інтерфейс і володіє гнучкими та зручними налаштуваннями.

В якості бази даних використовується H2 - відкрита СУБД, повністю написана на мові Java. є доволі популярним рішенням для середнього та малого бізнесу. H2 працює майже на будь-якій відомій Unix-платформі та під управлінням Windows Vista/7/8/10.

При розробці баз даних, доступ до яких здійснюється через web-сайти, написані за допомогою технологій Java, вибір краще усього зупинити на СУБД H2 за наступними причинами:

- легка інтеграція з Java додатками;
- простота у використанні;
- кросплатформеність;
- користувацькі функції і тригери;
- більша захищеність, ніж у нативних додатків;

- підтримка Unicode.

Для забезпечення web-інтерфейсу до бази даних H2 використовується мова програмування Java, що відповідає за серверну частину. Web-сервер обробляє такі файли, і виводить користувачеві результати обробки у вигляді HTML-файлів.

Логічна структура програми наступна: після авторизації користувач, в залежності від прав доступу, має доступ до відповідних станів роботи програмного забезпечення. Крім того, в залежності від прав доступу від може виконувати відповідні дії над даними таблиць бази даних.

Для виходу з web-додатку користувач може закрити вікно або скористатись кнопкою "Вихід".

При написанні даного програмного продукту використовувався комп'ютер з наступними технічними характеристиками:

Intel(R) Core(TM) i5-2410U CPU @ 2.30GHz;

4 ГБ ОЗУ;

операційна система: Windows 10 Pro.

2 Функціональне призначення

Програмне забезпечення "Удосконалення вибору цільової аудиторії приватного медичного центру модифікованим методом аналізу ієрархій" призначене для автоматизації роботи відділу дослідження цільової аудиторії приватного медичного центру. Програма дозволяє на основі вхідних даних та заданої діяльності суб’єкта бізнесу створити кампанію по дослідженню цільової аудиторії у відповідності до облікового запису, на основі внутрішніх критеріїв відбору (кількість показників «Like», «Share», «Comments») проводити збір інформації, на основі зібраних даних та критеріїв, що визначають вигоди, витрати,

ризики проводити етапи аналізу цільової аудиторії, взаємодіяти з API модулями соціальних мереж та проводити обробку можливих помилок, пов’язаних з API модулями, зберігати дані в базі, корегувати данні, отримувати звіти що відображають портрет цільової аудиторії за кампаніями.

3 Налаштування програмного комплексу

Вхідною точкою до програмного забезпечення для дослідження цільової аудиторії приватного медичного центру є налаштування хостингу для запуску віртуальної машини Java. Після цього користувач може приступити до створення облікового запису та кампанії з дослідження цільової аудиторії.

4 Вхідні дані

Вхідними даними є дані початкових конфігурацій до кожної кампанії (критерії для аналізу вносяться у відповідні поля вводу, альтернативи знаходяться в процесі роботи модуля аналізу цільової аудиторії з програмними інтерфейсами модулів соціальних мереж).

5 Вихідні дані

Вихідними даними програми є звіти з дослідження цільової аудиторії, що відображають отримані дані за заданими критеріями.

ДОДАТОК Г – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення для удосконалення оцінювання розміру програмних проектів написаних мовою Python.

Галуззю застосування програмного забезпечення є аналіз програмних проектів для оптимізації процесу проектування програмного забезпечення, що дозволить зберегти час та зусилля при розробці і розширювані ПЗ.

2 Мета випробувань

Перевірка придатності програмного забезпечення до використання за призначенням, її відповідність заданим вимогам і документації. Перевірка працездатності та правильності роботи програмного забезпечення.

3 Вимоги до програми

При проведенні випробувань, функціональні характеристики програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до складу виконуваних функцій» Технічного завдання.

4 Вимоги до програмної документації

В комплект програмної документації повинні входити наступні документи:

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

5 Склад та порядок випробувань

Вимоги до складу та параметрів технічних і програмних засобів вказані в Додатку В.

Порядок проведення випробувань:

– виконуються контрольні тести в довільному порядку;

– робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

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» технічного завдання.

Перевірка вважається завершеною у разі відповідності складу і послідовності дій, при виконанні даної перевірки, вказаною вище пункту технічного завдання.

В процесі тестування була перевірена функціональність програмного забезпечення для удосконалення математичної моделі оцінювання розміру програмних проектів.

У таблиці, що наведено нижче, вказано перелік випробувань для завантаження вхідних даних, тому що це функціонал програмного забезпечення де користувач може помилитися.

Таблиця Г.1 – Методика випробування

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	2	3	4	5
1	Завантаження файлу не типу json	Усі функції обробки даних стали неактивними, а також була виведене повідомлення про неправильні дані	Виконано	
2	Завантаження json файлу із неправилною структурою json об’єкту	Усі функції обробки даних стали неактивними, а також була виведене повідомлення про неправильні дані	Виконано	
3	Завантаження json файлу із правильною структурою json об’єкту	Усі функції обробки даних стали активними і доступними для викостання	Виконано	

Таким чином були наведені програма та методика випробувань.

ДОДАТОК Д – ІНСТРУКЦІЯ КОРИСТУВАЧА

На початку роботи з програмним забезпеченням користувачу надається інтерфейс для виконання регресійного аналізу та виводу результатів на екран та графік.

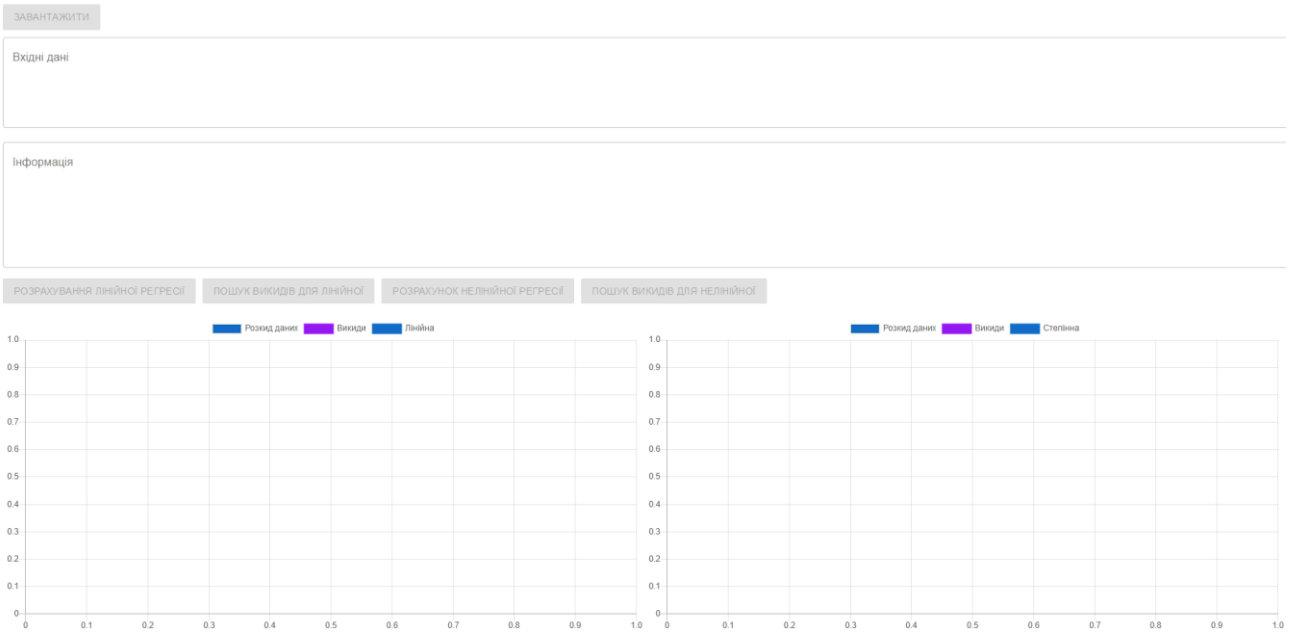

Рисунок Д.1 – Інтерфейс програми

На зображенні можна побачити усі необхідні для аналізу та оцінювання розміру програмних проектів функції. А саме: завантаження даних, розрахунок розподілу даних, перетворення вхідних даних, обчисленні лінійної регресії, метрики лінійної регресії, графік лінійної регресії, викиди лінійної регресії, обчисленні нелінійної регресії, метрики нелінійної регресії, графік нелінійної регресії, викиди нелінійної регресії.

Перше що користувач повинен здійснити – завантажити дані, для цього необхідно натиснути кнопку «Завантажити», після цього на графіку рівняння регресії будуть відображені усі точки завантажених даних.

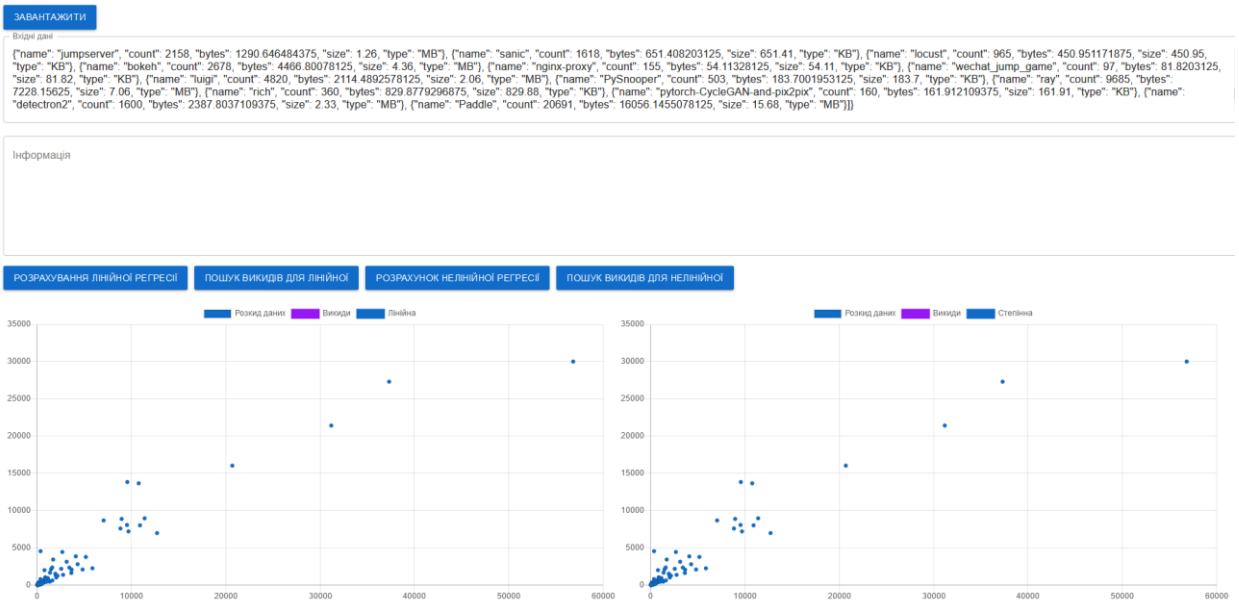

Рисунок Д.2 – Вхідні дані були завантажені

Зазвичай вхідні дані мають ненормальний розподіл, та багато аномалій. Для перетворення даних до нормального розподілу, необхідно провести нормалізацію, для цього необхідно застосувати функції розрахунку розподілу та перетворення даних. Для цього необхідно натиснути кнопки «Розрахувати розподіл даних» та «Перетворення вхідних даних». Після перетворення даних, графік приймає іншу форму розподілу, що зображено на рисунку Д.3.

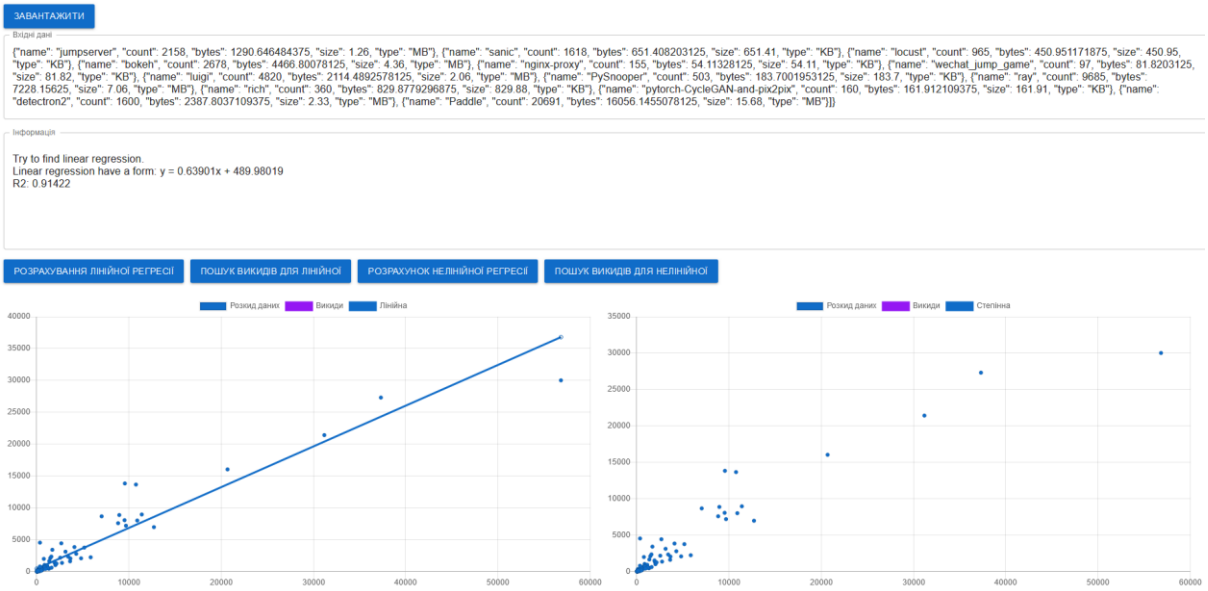

Рисунок Д.3 – Графік лінійної регресії

Після розрахованого рівняння регресії і метрик лінійної регресії, можна побачити поганий показник коефіцієнту кореляції, що значить, що дані мають аномалії, або значить що регресійна модель може бути вдосконаленою. Тому перше, що необхідно зробити – знайти викиди. Для цього необхідно натиснути на кнопку «Викиди лінійної регресії». Використання функції для викиду аномальних даних, покаже викиди на графіку іншим кольором, що зображено на рисунку Д.4.

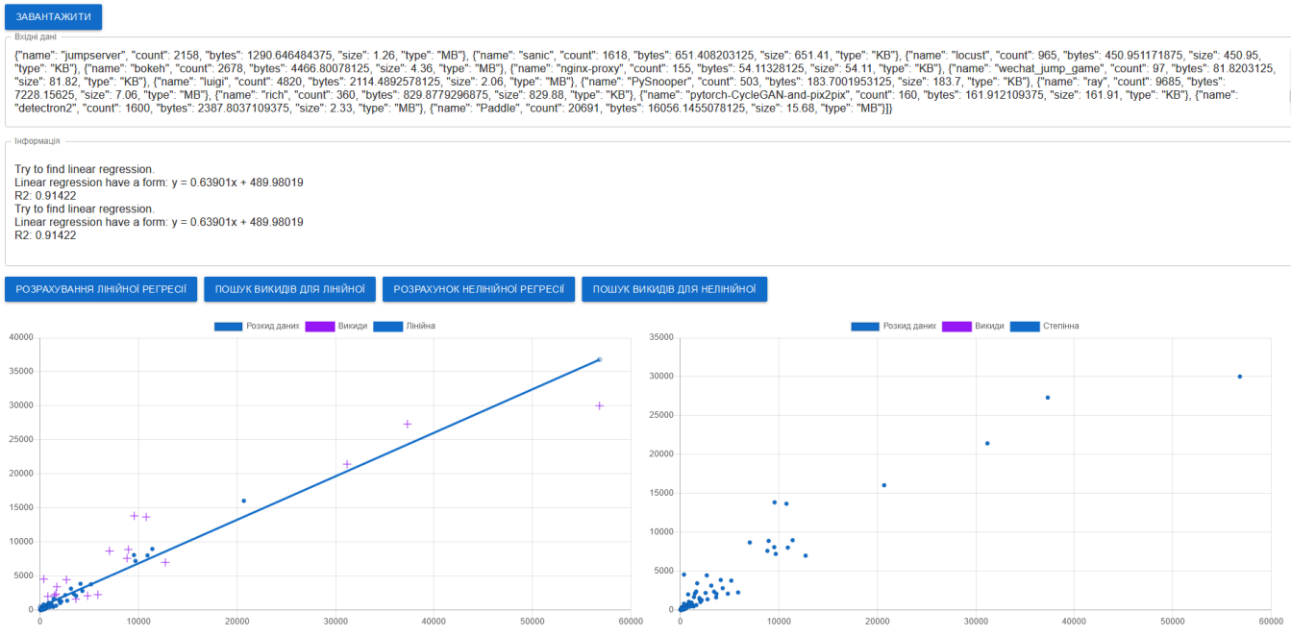

Рисунок Д.4 – Викиди лінійної регресії

Після визначення викидів графік лінійної регресії є неправильним, оскільки обчислення лінійної регресії має бути перераховано на нових даних без викидів. Для цього необхідно знову натиснути на кнопку «Обчислення лінійної регресії», після чого графік буде оновлено, а графік лінійної регресії буде перераховано на основі нових даних. Що зображено на рисунку Д.5.

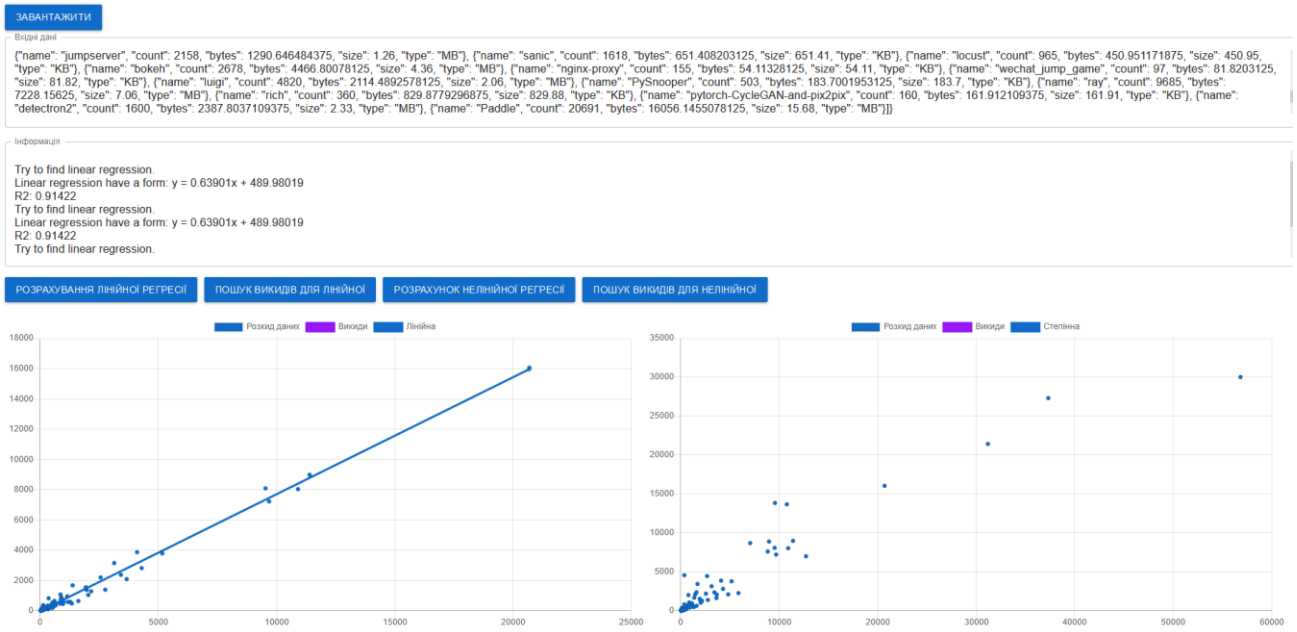

Рисунок Д.5 – Обчислення лінійної регресії без викидів

Після знайдених викидів можна перейти до моделі нелінійної регресії і розрахувати нелінійну регресію і вивести її на графік, що зображено на рисунку Д.6.

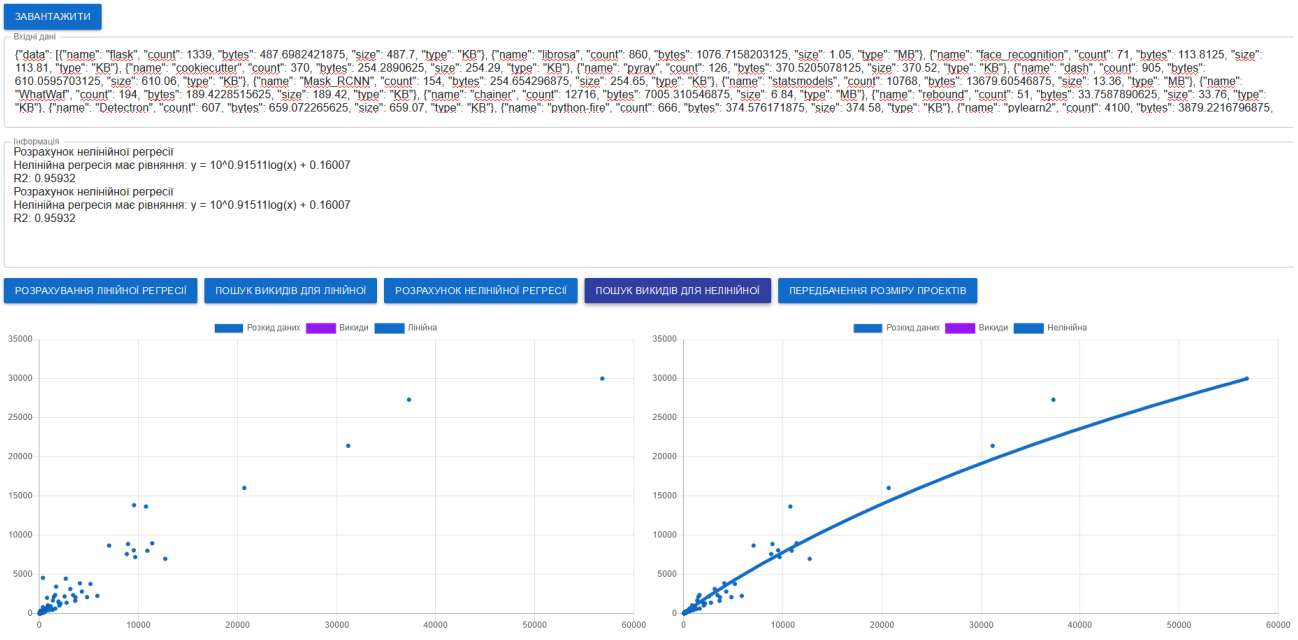

Рисунок Д.6 – Вивід нелінійної регресії

Для розрахунку розміру програмного проекту необхідно натиснути кнопку «Розрахунок розміру проектів» і ввести кількість методів, що зображено на рисунку Д.7:

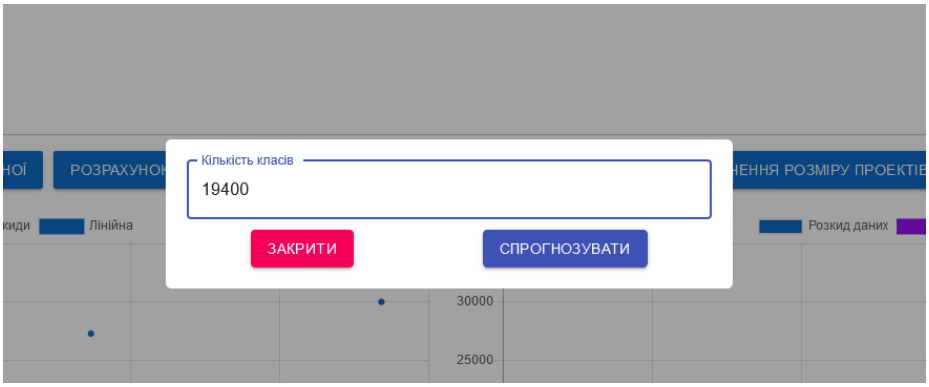

Рисунок Д.7 – Ввід кількості методів

Після розрахунку нелінійної регресії можна виконати розрахунок розміру програмного проекту, для цього не обхідно ввести кількість методів, після натискання кнопки «Розрахунок розміру проектів», що зображено на рисунку Д.8:

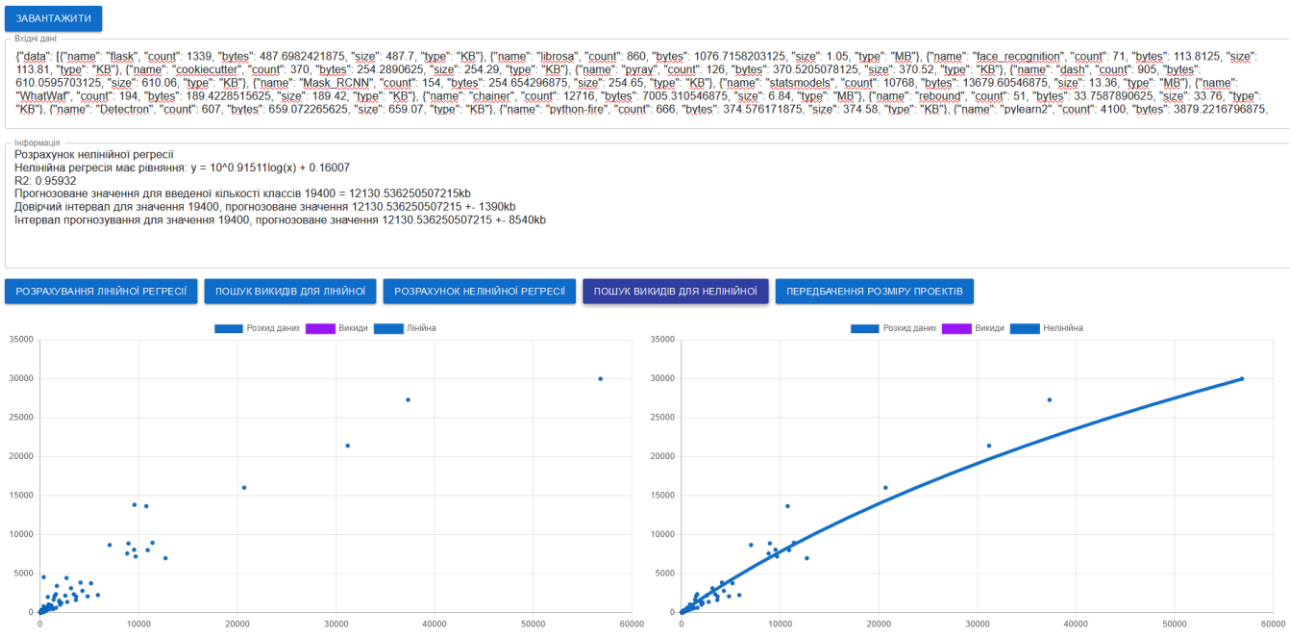

Рисунок Д.8 – Розрахований розмір для кількості методів програмного проекту

Таким чином була приведена інструкція користувача та представлені приклади з програми у лістингу Д.