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 рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на Java та розробка програми для його реалізації

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

Веретницький М.О
(підпис, прізвище та ініціали)

Керівник Устенко І.Р.
(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.

РЕФЕРАТ

Веретницький Михайло Олексійович

«Удосконалення рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на Java та розробка програми для його 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: 154 стор., 24 табл., 30 рис., 29 використаних джерел, 5 додатків.

Актуальність теми роботи: Передбачення розміру програмного забезпечення в кілобайтах є важливою рисою при проектуванні програмного забезпечення коли ми маємо фіксовані розміри простору для програми. Правильно спроектоване програмне забезпечення має вирішальну роль в успіху проекту. В подальшому планується переведення кілобайт до кількості строк коду, що може бути використано в оцінюванні трудомісткості та вартості проекту на основі моделі COCOMO II.

Мета та завдання дослідження. Метою є вдосконалення рівняння регресії та покращення її оцінок, для оцінювання інформаційних систем на java за рахунок зворотного перетворення до нелінійної моделі.

Завдання дослідження: проаналізувати існуючі регресійні моделі для оцінювання розміру програмного забезпечення на java; удосконалити регресійне рівняння; розробити програму для розрахування регресійного рівняння для оцінювання розміру інформаційних систем на мові java.

Об'єктом дослідження є процес передбачення розміру програмного забезпечення інформаційних систем з відкритим кодом на java за рахунок регресійного рівняння.

Предметом дослідження є рівняння регресії з допомогою якого можливо передбачити розмір програмного забезпечення на інформаційних систем на мові java.

Методи дослідження є метод знаходження важелів моделі, метод перетворення лінійної регресії до нелінійної, та статистичні методи оцінки моделі.

Наукова новизна одержаних результатів. Удосконалено лінійну регресійну модель для оцінювання розміру програм інформаційних систем на java на основі нормалізуючого перетворення у вигляді десяткового логарифму, яка в порівнянні з лінійною регресійною моделлю має більший коефіцієнт детермінації, менше значення середньої величини відносної похибки.

Практичне значення одержаних результатів. Розроблено регресійну модель, що можна використовувати для оцінювання розміру інформаційних систем на мові java у кілобайтах, з ціллю оцінювання бюджету на оренді простору для програмного продукту.

Ключові слова: зворотне перетворення, оцінювання розміру програмного забезпечення, видалені залишки.

ABSTRACT

Veretnitskii Mikhail Alekseevich

«Improving the regression equation to estimate a software size of open source Java-based information systems and developing the software for its implementation»

The qualification work in obtaining a master's degree in specialty 121 - "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolayiv, 2020.

The qualification work is presented on the 154 pages of typewritten text, contains 24 tables, 30 figures, 5 appendices and 29 references.

Relevance of the topic of the work: Predicting software size in kilobytes is an important feature when designing software when we have a fixed amount of space for the program. Properly designed software plays a crucial role in the success of a project. In the future, it is planned to convert kilobytes to the number of lines of code that can be used to estimate the complexity and cost of the project based on the COCOMO II model.

The goal and objectives of the study. The aim is to improve the regression equation and improve its estimates, to evaluate information systems in java by inversely converting to a nonlinear model.

The object of the study is a process of predicting the size of open source information systems software in java due to the regression equation.

The subject of the study is a regression equation with which it is possible to predict the size of software on information systems in java.

The methods of the study are a method of finding the levers of the model, a method of converting linear regression to nonlinear, and statistical methods of estimating the model.

The scientific novelty of obtained results. The linear regression model for estimating the size of java information systems programs has been improved by basis of normalizing transformation in the form of a decimal logarithm, which in comparison with the linear regression model has a higher percentage of determination, less than the value of the average relative error.

The practical significance of obtained. Developed a regression that can be used to predict the size of information systems in java in kilobytes, in order to estimate the budget for renting space for the software product.

Keywords: nonlinear regression model, regression equation transformation, deleted residual.

Зміст

ВСТУП

Актуальність теми. У нас час існує достатня кількість різних систем та метрик, які можуть аналізувати дані та допомагати прийняти рішення на основі цього. Розмір програмного продукту на мові java – це важливий фактор систем, які не мають змогу легко розширюватись та перевищувати заданий розмір [1], тому, є важливо на етапі проектування програми мати змогу оцінити її розмір на основі кількості класів та отримати результат в кілобайтах, і завдяки цьому прийняти вчасно рішення для перепроектування програмного продукту, якщо прогнозований розмір не буде влаштовувати. Це буде доцільним, тому що процес проектування коштую набагато дешевше ніж подовшений процес кодування в результаті несподіваних змін та подальші витрати в результаті переписування модулів програми [2]. В подальшому планується переведення кілобайт до кількості строк коду, що може бути використано в оцінюванні трудомісткості та вартості проекту на основі моделі COCOMO II [3]. Виходячи з цього, наукова новизна полягає в удосконалені рівняння регресії для оцінювання розміру інформаційних систем на java, в результаті чого будуть покращенні результати передбачення розміру пз, та метрики оцінки регресійної моделі, такі як коефіцієнт детермінації та показник передбачення.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є підвищення достовірності оцінювання рівняння регресії для оцінювання розміру інформаційних систем на java.

Завдання дослідження: проаналізувати існуючі регресійні моделі для оцінювання розміру програмного забезпечення на java; удосконалити регресійне рівняння; розробити програму для розрахування регресійного рівняння для оцінювання розміру інформаційних систем на мові java.

Об’єкт дослідження. Об’єктом дослідження є процес передбачення розміру програмного забезпечення інформаційних систем з відкритим кодом на java за рахунок регресійного рівняння.

Предмет дослідження. Предметом дослідження є рівняння регресії з допомогою якого можливо передбачити розмір програмного забезпечення інформаційних систем з відкритим кодом на java.

Методи дослідження. Методами дослідження є метод знаходження важелів моделі через метод найменших квадратів, метод перетворення лінійної регресії до нелінійної за допомогою зворотного та метод знаходження викидів за допомогою видалених залишків.

Наукова новизна. Наукова новизна роботи полягає в удосконаленні лінійного регресійного рівняння для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на java за рахунок десяткового логарифмічного нормалізуючого перетворення та знаходження викидів через метод стьюдентизованих зовнішних залишків, завдяки чому були покращені: коефіцієнт детермінації моделі, показник прогнозування та середня величина відносної похибки.

Практична цінність. Практична цінність результатів дослідження може бути виражена в аналізі та оцінюванні розміру програмного продукту інформаційних систем з відкритим кодом на java, завдяки чому може бути вирахувані потрібні витрати на придбання або оренді простору для збереження програмного продукту або прийняття рішенні щодо перепроектування окремих модулів програми для зменшення її розміру.

Особистий внесок здобувача. Дана робота є самостійно виконаною науковою працею.

1 АНАЛІЗ МЕТОДІВ ДОСЛІДЖЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ТА ЇЇ ПРИЗНАЧЕННЯ

1.1 Опис основної концепції регресійного аналізу та аналізу існуючих регресійних моделей для аналізу розміру інформаційних систем на мові java

Насамперед рівняння регресії – це математична формула (модель) яка застосовується к незалежним змінним, для прогнозування та аналізу залежної змінною, яку необхідно змоделювати. Залежну зміну позначають як Y, це повинна бути якась величина або процес якої потрібно спрогнозувати. Незалежна зміна позначається як X, незалежною змінною виступають якісь інші характеристики які на перший погляд, можливо, і не пов’язані з Y, їх може буду декілька, але залежна зміна завжди одна [4].

Головною частинної регресійного аналізу є знаходження коефіцієнтів для рівняння регресії, які розраховуються в результаті аналізу. Визначаються ваги для кожної незалежної змінною, які репрезентують тип и силу впливу незалежної зміною на залежну. Якщо співвідношення позитивне знак пов’язаного коефіцієнту також буде позитивний, також може бути зворотний зв’язок. Змінні з коефіцієнтами близько до 0 не допомагають передбачити або змоделювати залежні величини, вони практично завжди видаляються з регресійного рівняння, якщо тільки немає вагомих причин зберегти їх [5].

Якщо аналізувати інформаційні системи на мові java, можна знайти велику кількість програм для виконання різних задач, всі вони використовують різні підходи до проектування та написання програмного коду, та всі вони використовують конструкції коду які передбачені мовою програмування java. Є очевидним, що розмір програми може бути виражений та прогнозований через параметр, який може бути порахований на етапі проектування інформаційної системи. В такому випадку використання регресійного рівняння буде найбільш зручним шляхом знаходження такої закономірності. Для побудування

регресійного рівняння потрібно буде проаналізувати різні параметри мови програмування java та зробити вибір найбільш репрезентативних параметрів до даної мови та її архітектурній особливості, яке буде загально використовуваним у всіх видах програм на java.

Побудова регресійного рівняння не є кінцевою задачею, після цього треба довести, що дана модель, може задовольняти критеріями передбачення даних, та може пояснити існуючи спостереження.

Для оцінки на скільки влучно регресійна модель може пояснити існуючі спостереження існують спеціальні метрики для оцінювання рівняння регресії. Одна з най поширених є R-квадрат [6]. R-квадрат обчислюються з регресійного рівняння, щоб якісно оцінити модель. Більш математичними словами, R-квадрат – це статистичний метод аналізу моделі, який аналізую вибіркову дисперсію залишків. Тобто чим більша виборча дисперсія залишків, тим менше значення R-квадрат буде отримано. Треба звернути увагу, що R-квадрат не може бути більше 1.0 і він дорівнює 1 коли виборча дисперсія залишків дорівнює нулю – це коли вся наша модель без помилок може пояснити залежну зміну. Та коли вона більша за нуль – зменшується і якість детермінації. Та якщо R-квадрат дорівнює нулю, це означає що наша модель ні більш точна ніж використання вибіркового середнього вхідних даних. Наприклад, якщо значення R-квадрат було вирахуване як 0.71, можна пояснити це як «Побудована модель пояснює 71% варіантів залежної змінної». Щоб зрозуміти, як працює R-квадрат, побудуємо графік, що відображає спостережувані і оцінювані значення Y, які відсортовані по оцінюваним величинам. Таким чином, R-квадрат являє собою мірою наскільки близько побудована модель співпадає з даними, і цим ближче це значення до одиниці – тим ближча наша модель до ідеалу, і в разі якщо вона менша – тим гірше [7].

Важливим результатом кваліфікаційної роботи буде отримання моделі, яка зможе задовольнити прийнятний рівень детермінації та прогнозування. Оскільки віднайдене рівняння регресії ніколи не зможе описати точний розмір

програмного забезпечення на основі параметра, тим більш одного. Але завдяки статистичних методів є можливість знайти найбільш вірогідний інтервал передбачення таких значень. В регресійному аналізу для таких випадків використовують інтервал передбачення та довірчий інтервал [8].

Більшість регресійних методів також обчислюють p-значення, для коефіцієнтів, та вхідних даних, пов'язаної з кожної незалежної змінної. P-значення використовуються доволі часто, коли виконуються тест статистичної значимості, наприклад t-тест, критерій хи-квадрат і т.д. [9]. Зазвичай ці тести дають нам вирахувану статистику тесту і пов’язане p-значення. Ці значення використовуються для встановлення статистичних важливості вхідних даних. Маленькі значення p-значень кажуть о маленькій ймовірності і можуть припускати, що коефіцієнт важливий для моделі з значенням, що сильно може відрізняється від 0 (тобто, маленькі величини p-значень свідчать про те, що коефіцієнт не дорівнює 0).

Також при будуванні моделі регресії, є вірогідність зітнутися з точками які дуже сильно впливають на модель, і сильно тягнуть функцію на себе, такі точки можуть бути викидами, які треба ідентифікувати, та вилучити з вибірки даних [10].

Існують різні методи знаходження викидів, використання яких буде залежите від даних та моделі яка використовуються. Треба пам’ятати про аномальні дані та їхній вплив на статичні методи аналізу, такі як середнє значення, дисперсію та стандартне відхилення. Існують наступні методи пошуку таких відхилень:

- Для незмінних використовується Z-значення або міжквартільний діапазон;
- Для лінійної регресії використовують «leverage statistics» , дистанція Кука та «DFFITS» метод, стьюдентизовані зовнішні залишки;

•При використані двовимірних і більш багато вимірних варіантів використовують «Robust covariance», локальні фактор викиду або «Isolation forest».

Пошук викидів є важливою задачею для побудови рівняння регресії яке зможе задовольняти високим метрикам оцінки моделі. Для цього потрібно обрати ефективний спосіб знаходження викидів, який будить підходити під модель.

1.2 Огляд різних типів регресійних моделей

Існує багато різних видів регресії, кожна з них має свої переваги і недоліки.

Приклади регресійних моделей: лінійні функції, алгебраїчні поліноми, нейронні мережі без зворотного зв'язку (наприклад, одношаровий персептрон Розенблатта), радіальні базисні функції, степеневі, Гребньові, Лассо регресії, логічні та інші.

Лінійна регресія

Лінійна регресія – це статистичний метод, який відноситься до прогнозу числового цільового об’єкту, її основна мета полягає в пошуку лінійній функції, яка найближче чином інтерполює вибірку. Геометрично ця функція являє собою гіперповерхню, котра зближує точки y_i по предикатах $\overline{x_i}$ при $i = 1, \dots, l$. Така регресія намагається змоделювати взаємозв’язок між двома змінними. Наприклад, використати лінійну модель можливо для оцінювання ваги людини з її ростом.

Задача пошуку рівняння регресії вже була розв’язана ще в 17 віці Гауссом та Лежандром за допомогою мінімізації суми квадратів різниць значень функції $f = (\overline{x_i})$ і точок y_i. Більш загальна теорія метода була цілком описана в математичній статистиці, для випадку лінійної моделі з моделювання даних с гаусовим випадковим шумом [11].

Сама найпростіша лінійна регресія – це парна, модель, що може моделювати взаємозв'язок між значеннями однієї вхідний незалежної і однієї вихідний залежною змінними за допомогою лінійної моделі, наприклад, рівняння прямій. Більш розповсюдженою моделлю є множинна лінійна регресія, тобто яка має більш ніж одну незалежну змінну. Модель лінійної регресії виглядає наступним чином [12]:

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k.$$

До переваг моделі відноситься: гнучкість, простота, однаковість їх проектування та аналізу. Використовуючи таку модель в результаті буде отримана більш людино зрозумілу система, розрахування якої не буду займати дуже багато часу, як у випадку якщо використовувати більш складну модель. С цього випливає те – що модель є більше зрозумілою для людини і в багатьох випадках достатньо поглянути на модель для того щоб оцінити її коректність та адекватність. Робити зміни в цій моделі для корегування також не складно.

Але в цій простоті є і недоліки, такі як чутливість до викидів, так і низьку адаптивність. Також у випадках нелінійних взаємовідношень модель не використовується, тому що не може моделювати такі процеси.

Для поліпшення лінійної регресії можна використовувати її перетворення до нелінійної, це легко зробити завдяки перетворенню вхідних спостережень, та далі зробити зворотне перетворення отриманого рівняння. Найчастіше використовується логарифмічне перетворення. Рівняння буде мати наступний вигляд [13]:

$$\log(y) = b + a * \log(x) . \quad (1.1)$$

Дане рівняння має лінійну форму запису, та його можна перетворити на нелінійне рівняння наступного виду [13]:

$$y = 10^{b+a*\log(x)} . \quad (1.2)$$

Дане рівняння вже є нелінійним, в даному випадку ми лише виразили Y через визначення логарифму. Таку форму регресії застосовують коли мають вхідні дані які не мають нормальний розподіл даних [14]. Оскільки

перетворення логарифму може виправити зсув даних на гістограмі розподілу та наблизити даних до нормального розподілу.

Поліноміальна регресія

Для випадку нелінійних співвідношень можна використовувати і поліноміальну регресію. Функція моделює значення на всій області визначення і є парною. Як і у випадку з лінійною регресією вона використовує взаємовідношення між залежної та незалежної змінними, для пошуку найближчої функції, яка буде описувати дані. Головний концепт поліноміальної функції є те що зміна визначається ступенем.

Моделювання нелінійних даних є важливої перевагою використання поліноміальної регресії на підміну від лінійної регресії, тому така регресія більш гнучка та має змогу працювати з більш складними вхідними даними. Але за це потрібно сплачувати більшою складністю будування такої моделі, вона потребує окремих знань про вхідні дані, а також ретельної уваги щодо підбору найбільш точніших значенням ступеня, у випадку неправильного ступеня буде отримана неадекватна модель, тому перед використанням даної моделі потрібно ретельно вивчати досліджуємий об’єкт.

Функція поліноміальної регресії виглядає наступним чином [15]:

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2^2 + \cdots + b_n x^n + \varepsilon\,.$$

Гребньова регресія або Рідж регресія

У випадку якщо лінійна та поліноміальна регресії є недостатньо ефективними та незалежні зміні колінеарні, тобто співвідношення незалежних змінних може буте виражено як лінійне. В цьому випадку можна використати гребньову регресію (Рідж регресію).

Гребньова регресія – це модель яка має міру корегування для зменшення колініарності між вхідними даними.

Якщо зміні будуть корелювати між собою буде отримана велика дисперсія. Для цього гребньова регресія використовує квадратичне зміщення

для збалансування дисперсії. Приведений порядок формул для отримання моделі [16]:

Розглянемо залежність

$$f(x, \beta) = \langle \beta, x \rangle,$$

Знаходимо вектор

$$Q(\beta) = \|F\beta - y||^2,$$

$$\beta^* = arg \min_{\beta} Q(\beta) \ .$$

Знаходимо розв’язок за допомогою метода найменших квадратів

$$\beta^* = (F^T F)^{-1} F^T y \ .$$

В умовах мультіколінеарності матриця $F^T F$ стає погана обумовленою. Для розв’язку цього застосується обмеження на величину коефіцієнтів

$$\beta: ||\vec{\beta}||_2^2 \leq t^2.$$

Функціонал Q з урахуванням обмежень приймає вид:

$$Q(\beta) = \left||Y - X\beta\right||^2 + \lambda \left||\beta\right||^2.$$

Де $\lambda -$ не негативний параметр, розв’язок таким чином має вигляд:

$$\beta^* = (F^T F + \ \lambda I_n)^{-1} F^T y \ .$$

Діагональна матриця λI_n зветься гребнем.

Лассо регресія

Також існує регресія по методу лассо, вона також використовує значення для зміщення функція для її оптимізації та зменшення дисперсії моделі. Але замість використання квадратичного зміщення, використовується зміщення абсолютним значенням β. Наведена формула [16]:

$$||\vec{\beta}||_1 \leq t \ .$$

Фунціонал Q приймає наступний вид:

$$Q\lambda(\beta) = \left||F\beta - y\right||^2 + \lambda \left||\beta\right|| \ .$$

Виділяючи різницю між двома моделями, як гребньової та лассо, можна зробити висновок, що перша може привести вхідні незалежні зміні до нуля, а друга зменшує їх до значень близьких нулю.

Логістична регресія

Логістична регресія – узагальнений метод регресії використовується у випадках, коли вхідні змінні є двійковими, але при цьому залежна зміна не повинна бути двійковою.

Логічну регресію зручно використовувати для знаходження незалежних змінних, якщо задачу можливо вирішити за рахунок булевих функцій від інших незалежних змінних. Можна побачити, що часто методи регресії не зважають на зв'язок між іншими змінними, це можна зрозуміти як: якась конкретна зміна не має впливу на розрахування результатів на основі інших змінних – існують випадки коли ця умова невірна [16].

Наприклад x1, x2, ..., xk – набір змінних які мають двійкову природу та є незалежними змінними, тоді y – буде залежною змінною. Після цього треба розв'язати та побудувати модель регресії виду [16]:

$$g\big(E(y)\big) = b_0 + b_1 L_1 + \cdots + b_n L_n \,.$$

Де $L_g -$ булева представлення функція від незалежної змінною x_i. Далі для кожного типу моделі необхідно визначити функцію, яка найближче відображає сутність даної моделі. Наприклад, для лінійної регресії такою функцією може бути залишкова сума квадратів. Метою методу логічної регресії є мінімізація обраної функції якості за допомогою налаштування параметрів b_j одночасно з булевими виразами L_j.

Для даної кваліфікаційної роботи, у базі програм на java дуже складно знайти вхідні дані які будуть мати нормальність розподілу, тому буде доцільно використовувати перетворення вхідних даних для виправлення цього. Використання десяткового або натурального логарифму не є суттєвим, розподіл даних після їх використання виглядає майже однаковим. Оскільки буде доцільним використовування перетворення даних, та далі знаходження

лінійної рівняння регресії, потрібно застосовувати і зворотне перетворення для рівняння для того, щоб модель враховувала перетворення даних, в результаті буде отримано нелінійне рівняння регресії, яке і буде використовуватися для вдосконалення.

1.3 Розгляд методів знаходження викидів для регресійної моделі

Викид – це таке аномальне значення, яке знаходиться далеко від більшості інших спостережень та лежать значніш далі ніж інші спостерігання. Поява таких аномалій частіше всього пов’язане з впливу на сутність рідких або не врахованих чинників, також є можливість помилки на стадії вимірювання.

Для регресії викидом називають запис, фактичне значення якої знаходиться далеко від передбаченого значення. Візуально викиди визначаються як точки на графіку, які значніш віддалені від всього іншого набору точок. На рисунку 1.1 [17], представленому нижче, викид виділений в кругову область.

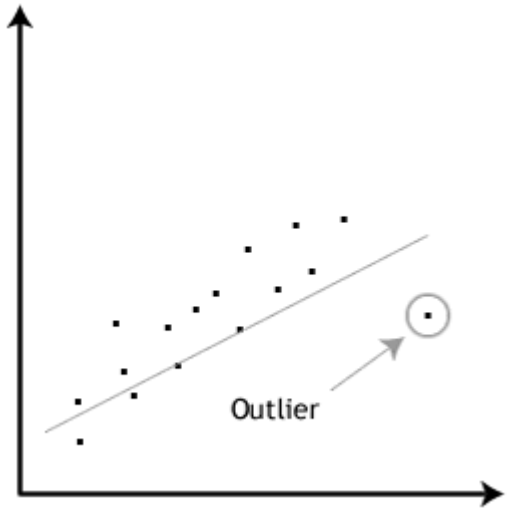


Рисунок 1.1 – Приклад аномального значення

Часто викиди (тобто аномальні значення) можуть привести до серйозного зсуву оцінок, переміщаючи лінію регресії в певному напрямку і таким чином,

викликаючи дислокацію регресійних коефіцієнтів і тим самим виняток всього одного аномального значення призводить до зовсім іншого результату [18].

Для майже усіх методів пошуку використовується так звані «залишки», це означає різницю для усіх $i = 1, \dots, n$ між спостерігаєм значенням (фактичним) та прогнозованим значенням на основі регресійної моделі. Наведена формула залишків моделі [18]:

$$e_i = y_i - \widehat{y_i}\,.$$

Для прикладу наведемо таблицю, де позначені фактичні значення залежною зміною та прогнозовані на основі моделі (див. табл. 1.1).

Таблиця 1.1 – Зразкові дані

x	y	прогнозоване	різниця
1	2	2.2	-0.2
2	5	4.4	0.6
3	6	6.6	-0.6
4	9	8.8	0.2

Головна проблема звичайних залишків складається в тому, що їх значення залежить від одиниці виміру, що ускладнює застосування залишків як способу виявлення викидів. Для того щоб виключити одиниці виміру, потрібно поділити залишки на оцінку їх стандартного відхиленням, тим чином отримати так названі «стандартизовані залишки».

Стандартизовані залишки

Стандартизовані залишки (також відомі як внутрішні стьюдентізіровані залишки) визначаються для кожного спостерігання, $i = 1, \dots, n$ як звичайний залишок поділений на оцінку його стандартного відхилення [19]. Наведена формула стандартизованих залишків [19]:

$$r_i = \frac{e_i}{s(e_i)} = \frac{e_i}{\sqrt{MSE(1 - h_{ii})}},$$

Формула MSE [20]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_y)^2 . \quad (1.3)$$

На таблиці можна побачити, що стандартизований залишок для точці не залежить тільки від звичайного залишку, але також від розміру середнє квадратичної помилки і показнику спостерігання (Leverage) h_{ii}.

В таблиці 1.2 приведений зразок розрахунку залишків.

Таблиця 1.2 – Зразкові дані стандартного відхилення

x	y	прогнозоване	різниця	оцінка	Стандартизоване відхилення
1	2	2.2	-0.2	0.7	-0.57735
2	5	4.4	0.6	0.3	1.13389
3	6	6.6	-0.6	0.3	-1.13389
4	9	8.8	0.2	0.7	0.57735

Хороші риси стандартизованих залишок є в тому, що вони кількісно визначають, наскільки великі залишки в одиницях стандартного відхилення, тому їх можна легко використовувати для виявлення викидів. Значення стандартного залишку яке перевищує 3 по абсолютній величині можливо вважати викидом, деякі статистичні програми вважають викидом будь яке спостерігання стандартизованої помилки більше ніж 2.

Використання значення для викиду як 2 є трохи поспішним, тому пропонується краще помічати ці спостереження як червоні прапорці, та продовжити спостерігати дані точки.

Стьюдентізіровані залишки

Існують такі випадки коли потенційний викид впливає на регресійну модель таким чином, що функція регресії достатньо зміщується к викиду, тому цей викид фактично не знаходиться. Для пошуку таких викидів не підходить метод стандартизованих залишків, для вирішення цієї проблеми є стьюдентізіровані залишки, які пропонують альтернативний критерій для

виявлення викидів. Головна мета складається в тому, щоб видаляти спостерігання по одному, і кожний раз перераховувати регресійну модель для залишившихся n – 1 спостерігань. Потім порівнюється фактичні значення з їх прогнозованими значення на основі моделі з видаленим і-м спостеріганням. Це також називають видалені залишки. Стандартизація видалених залишків дає стьюдентізіровані залишки.

Наведемо формулу видалених залишків [21]:

$$r_i = y_i - \widehat{y_{(i)}},$$

Де y_i – фактичне спостережене значення, а $\widehat{y_{(i)}}$ – значення яке було отримано на основі побудованої регресійної моделі, без врахування i спостерігання.

Для того щоб порахувати видалені залишки, потрібно знайти залишок для кожної нової моделі регресії від видаленого значення. В підсумку отримуються значення для кожного видаленого значення, і в цьому випадку можливо отримати будь які значення різниці. Але як оцінити ці дані, та зрозуміти що це є викид? Точної відповіді немає, вихідні данні залежать від одиниць виміру, як і прості залишки. Але є шлях який допомагає вирішити цю проблему – поділив кожний видалений залишок на оцінку його стандартного відхилення, де в цьому і допомагає стьюдентізіровані залишки.

Стьюдентізіровані залишками (також їх можна назвати зовнішні стьюдентізіровані залишки) є [21]:

$$t_i = \frac{d_i}{s(d_i)} = \frac{e_i}{\sqrt{MSE_{(i)}(1-h_i)}}, \tag{1.4}$$

$$h_i = \frac{1}{n} + \frac{(x_i-\bar{x})^2}{(n-1)s_x^2}, \tag{1.5}$$

Де, MSE визначається з формули (1.3).

Тобто стьюдентізіровані залишки – це лише видалений залишок, поділений на його оціночне стандартне відхилення. Виявляється, це еквівалентно простому залишку, поділеному на коефіцієнт, котрий включає середнє квадратичну помилку на основі оціненою моделі з видаленим і-м

елементом. Треба звернути увагу на те, що єдина різниця між стандартизованими залишками і стьюдентізірованими залишками, є в тому – що стандартизовані залишки використовують середнє квадратичну похибку моделі, в основі всіх спостерігань, в той час як стьюдентізіровані залишки використовують середнє квадратичну похибку основану на вирішеною моделлю з видаленим і-м спостеріганням.

В цілому, стьюдентізіровані залишки є більш ефективними для пошуку викидів залежною змінною Y, ніж стандартизовані залишки. Якщо в спостеріганні є стьюдентізірованна помилка яка перевищує 2 або 3, ми маємо змогу назвати це викидом [21].

Статистика впливу

Також для пошуку викидів можна використовувати статистику впливу, за допомогою цього можна оцінити вплив окремих даних на всю регресійну модель. В більшість випадків впливові спостереження мають сильний вплив на вихідну модель та визначають залежність регресії, тому рекомендується проводити також розрахунок оцінок впливу для даних.

В цілому методи пошуку впливу можна поділити на дві групи, загальні та специфічні. Якщо казати про загальні методи, то їх ідея полягає в оцінці як кожне окреме спостереження може мати вплив на регресійну модель та її зображення. До загальної групи можна віднести наступні методи:

- відстань Кука;
- коваріативне відношення;
- «DFFITS»;
- відстань Махаланобіса.

Друга група методів це специфічні, їх ідея вирахувати вплив окремого спостереження на окремі параметри моделі. До цієї групи відносять:

- «DFBETAS».

Також можна виділити те що кожний метод за виключенням відстані Махаланобіса, використовують спосіб виключення кожного вхідного значення для пошуку значень впливу.

Відстань Махаланобіса

Це є найбільш використовуваний метод видалення викидів в багатомірному просторі, тобто значень які мають аномальні комбінації змінних. Тобто Відстань Махаланобіса – це відстань між точками у багатомірному просторі, відстань відміряється відносно центроїду, тобто така точка, яка може бути як середнє для багатовимірних вхідним величинам, іншими словами це точка де перетинаються усі значення середніх значень. І чим більша буде значення відстані Махаланобіса тим далі від центроїду точка знаходиться.

Формула для вектору $x = (x_1, x_2, \dots, x_n)$ [22]:

$$Dm(x) = \sqrt{(x-\mu)^T S^{\,1}(x-\mu)},$$

Де $x = (x_1, x_2, \dots, x_n)^T$, та $\mu = (\mu_1, \mu_2, \dots, \mu_n)^T$.

Формула для виміру схожістю між двома випадковими векторами $\vec{x}$ та $\vec{y}$ [22]:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{\,1}(\vec{x} - \vec{y})}.$$

Відстань Кука

Для пошуку значних викидів у моделі регресії можна використовувати відстань Кука, тобто метод здатний знайти точки які негативно впливають на регресійну модель. Вирахування складається с комбінації розрахування «Leverage» (показник впливу) для кожного спостереження та його залишків, чим більше буде значення розрахунків тим більша відстань Кука. Наведена формула [23]:

$$D_i = \frac{\sum_{j=1}^{n} (\widehat{y_j} - \widehat{y_{J(i)}})^2}{P \times MSE^2},$$

$\widehat{y_{J(\iota)}}$ – значення яке було отримано на основі побудованої регресійної моделі, P – коефіцієнт регресійної моделі, MSE – середньо квадратична похибка, формула (1.3).

Для знаходження критичного значення відстані Кука, розраховується медіана F-розподілу з числом ступенів свободи $v_1 = k, v_2 = n - k$, k – кількість параметрів системи.

Коваріативне відношення

Уявляє собою відношення детермінанта коваріативної матриці з умовою видалення кожним спостереженням к детермінанту коваріативної матриці набору з всіх даних. При роботі з парною регресією зручно використовувати таку формулу [24]:

$$CR_i = (\frac{S^2_{e(i)}}{S^2_e})^2 * \frac{1}{1 - h_i},$$

h_i – показник впливу (формула 5).

Для визначення значення впливова точка чи ні, вважаються результати які суттєво відрізняються від 1. Щоб виконати перевірку необхідно виконати наступне порівняння [24]:

$$|CR_i - 1| > \frac{3k}{n}.$$

Якщо воно виконується спостереження можна вважати впливовим на регресійну модель, видалення таких значень буде мати вплив на регресійне рівняння та збільшить довірчий інтервал.

DFFIT міра та стандартизований DFFIT

DFFIT – це зміна прогнозованого значення для якоїсь точки, яке було отримано коли ця точка не враховується при розрахуванні регресії. Приведена формула [24]:

$$DFFIT_i = \widehat{y_i} - \widehat{y_{(i)}}$$

Для того щоб розрахувати стандартизований DFFIT існує декілька способів, в першому випадку стає зрозуміло чому це називають

стандартизованим, а іншому випадку що стандартизований DFFIT об’єднує в собі стьюдентізіровані залишки та показник впливу спостереження, це схоже як визначається відстань Кука, але треба зауважити, що відстань Кука використовує внутрішні стьюдентізіровані залишки, а DFFITS (стандартизований DFFIT) використовує зовнішні стьюдентізіровані залишки що є більш точної мірою визначення. Для пошуку значення використовується формула [24]:

$$DFFITS_i = \frac{DFFIT_i}{S_{e(i)}\sqrt{h_i}} = t_i\sqrt{\frac{h_i}{1-h_i}}\ .$$

Для не великої кількості спостережень критичним значенням вважаються більше 1, а для великої кількості значення більше ніж $2 * \sqrt{\frac{k}{n}}$.

При аналізі методів пошуку викидів найбільш ефективним виявились видалені стьюдентизовані залишки та DFFITS, вони знаходять найбільшу кількість викидів, після виключення яких якість моделі покращуються. Відстань Кука та стандартизовані залишки є менш ефективними, тому що при розрахунку викидів, дані методи використовують звичайні залишки, тобто вони виділяють викиди за рахунок відстані від первісно вирахуваної моделі та вони не можуть визначати на скільки сильно спостереження оказує вплив на регресійне рівняння. Коваріативне відношення не є доцільним для пошуку викидів оскільки виражає загальну міру впливу, тому в результаті найчастіше можуть бути позначені як впливові велика кількість значень, та деякі з них не є викидам, але мають вплив на регресію. Стьюдентизовані залишки та DFFITS використовують видалені залишки, які виражають різницю між моделлю з урахуванням спостереження та без, тому вони показують впливовість спостереження на регресійне рівняння, та якщо цей вплив дуже високий, дане спостереження позначається як викид. В цілому стьюдентизовані залишки та DFFITS дуже схожі і можуть бути виражені один з одного. Тому є доцільним використовувати в якості інструмента пошуку викидів в регресійному рівнянні

для оцінюванні розміру програми метод стьюдентизованих зовнішніх залишків.

1.4 Удосконалення лінійної регресії

Математика, яка є фундаментом лінійної регресії, утворює деякі найважливіші припущення про дані для визначення моделі:

1) Відсутність гетероскедастичності;
2) Відсутність мультиколінеарності;
3) Нормальний розподіл;
4) Лінійність.

Відсутність гетероскедастичності

Це означає що, лінійна регресія передбачає, що дисперсія між точками не збільшується і не зменшується від впливу залежною змінною

Тобто, лінійна регресія уявляє, що дисперсія між двома точками не збільшується і не зменшується в залежності від залежної змінної. В цьому випадку стандартна помилка лінійної моделі не буде надійною.

Способи віднайти помилку можуть бути такими:

1. Тест Голдфелда – Куандта

Для цього потрібно відсортувати дані за незалежними змінними, та поділити дані, тобто потрібно взяти перші i та останні m спостерігань, де $m < n / 2$. Середні $n - 2m$ спостерігань видаляються, і в цілому видаляються біля четверті від усієї вибірки [25].

$$F = \frac{\frac{RSS_1}{m-k}}{\frac{RSS_2}{m-k}} = \frac{RSS_1}{RSS_2}$$

Де RSS_1 – сума квадратів остатків першої частини вибірки, а RSS_2 – другою.

2. Тест Бройша – Пагана.

Потрібно знайти конзистентну оцінку

$$\widehat{\sigma^2} = RSS/n$$

Де RSS – сума квадратів остатків, n – об’єм вибірки.

Далі знаходяться квадрати стандартизованих залишків [26]:

$$g_i = \frac{e_t^2}{\widehat{\sigma^2}} = \gamma_0 + z_t^T\gamma + v_t,$$

$$LM = (TSS - SSR)/2,$$

Де TSS – сума квадратів остатків g_i від їх середнього 1 і SSR – сума квадратів залишків від допоміжної регресії.

Відсутність мультиколінеарності

Розглянемо відсутність мультиколінеарності на прикладі задачі, в котрій потрібно спрогнозувати вартість об’єкту нерухомості, виходячи з довжини ділянки, площа земельної ділянки, а також близькості до школи і громадської інфраструктури. Тут очевидно, що 2 незалежні змінні (довжина і площа ділянки) безпосередньо пов'язані між собою. У міру збільшення довжини збільшується і площа. Така кореляція впливає на ефективність лінійної регресії.

Це можна визначити завдяки використанню кореляції Пірсона. А виправити це можна відкинувши одну з змінних або створивши функцію для нової незалежної змінної з використанням корелюється функцій і відкинувши їх.

Нормальне розподіл

Нормальність розподілу є одним з найбільш важливих факторів, який не враховують перед побудовою лінійної моделі. Важливо, щоб безперервні змінні в наборі даних були розподілені по Гауса.

Гаусове розподілення – це розподіл ймовірностей, симетричне щодо середнього, що показує, що дані, близькі до середнього, зустрічаються частіше, ніж дані, далекі від середнього.

У графічній формі нормальний розподіл буде виглядати як дзвоно образна крива, як показано на рисунку 1.2 [27].

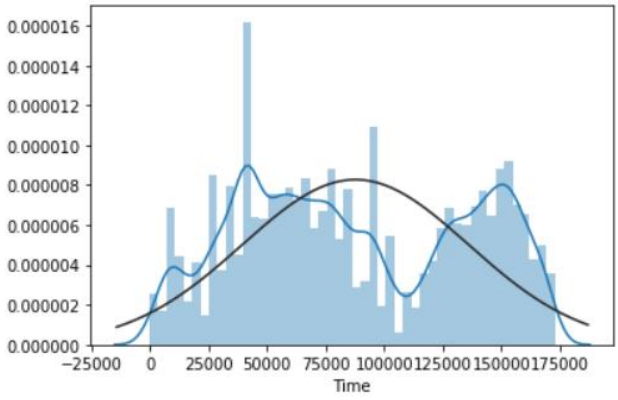


Рисунок 1.2 – Дзвонообразна крива

На наведеному вище рисунку чорна лінія відноситься до гауссового розподілу, яке можна досягти, а синя лінія представляє оцінку щільності ядра даних перед перетворенням.

Усунення асиметрії даних може значно підвищити точність лінійних моделей.

Перетворення, які можна застосувати для усунення перекосу:

• Логарифмічні перетворення: найкраще працює, якщо дані нахилені вправо, тобто розподіл має довгий хвіст на правому кінці.

• Експоненціальне перетворення: збільшення розподілу в ступеня «c», де «c» - довільна константа (зазвичай від 0 до 5).

• Перетворення Бокса-Кокса: це перетворення усуває перекіс в най меншій мірі в 80% випадків. Це, мабуть, найпотужніший інструмент для усунення перекосу.

Лінійність

Лінійна модель проводить пряму лінію через задані їй точки даних, як показано на рисунку 1.3 [27] нижче.

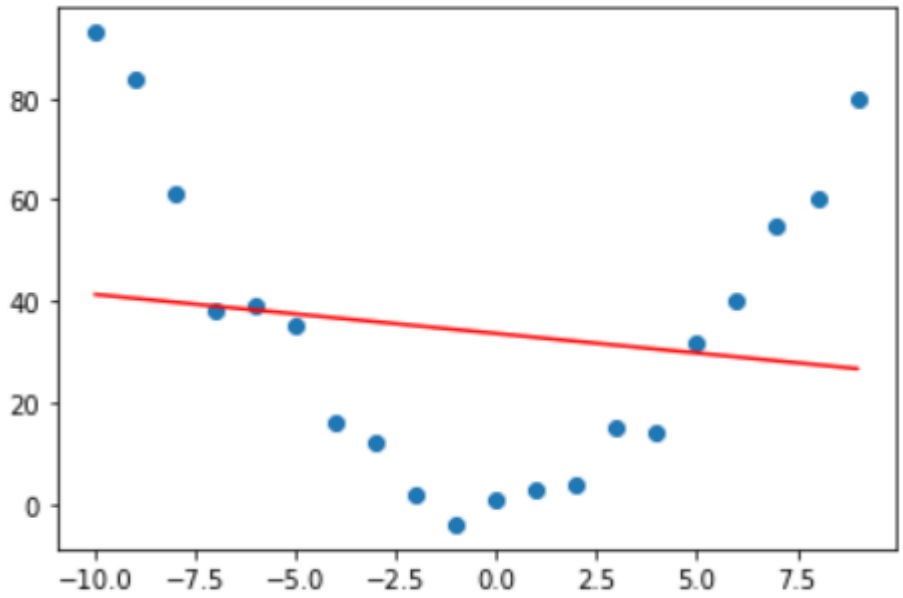


Рисунок 1.3 – Приклад недоцільності використання лінійної моделі

Тим не менш така модель не може відповідати точкам, які побудовані не лінійно. Розглянемо рівняння:

$$y = x^2 + c \pm \varepsilon$$

Графік для цієї функції є параболічним. Якщо провести лінію через цей графік, це не призведе до хорошого відповідності.

Можна перетворити незалежні змінні, застосовуючи експоненціальні, логарифмічні або інші перетворювачі, щоб отримати функцію, яка буде максимально наближена до прямої.

Це означає, що, змінивши незалежну змінну з x на x^2 шляхом зведення в квадрат кожного члена, можна провести пряму лінію через точки даних, зберігаючи при цьому добре середньоквадратичне відхилення.

Таким чином, необхідно з'ясувати, чи пов'язана незалежна змінна безпосередньо з кожної залежної змінної, перш ніж будувати кінцеву модель. Для цього також буде зручно розрахувати залишки та вивести їх на графік, та

якщо дані будуть розташовані хаотично це буде свідчати про лінійну залежність між даними.

1.5 Опис мови програмування Java

Мова програмування java спочатку проектувалась для використання в інтерактивному телебаченні, але в той час ця технологію було через мірно розвинута для кабельного телебачення. Історія появи java розпочинається з команди розробників «Green team», їх цілю була розробка зручної мови яка може працювати на великій кількості цифрових пристроїв, таких як телевізійні приставки, телевізори і інші. Але він також мав змогу працювати для програмування веб застосунків. Пізніше цю технологію було додано до Netscape.

Основні принципі якими керувалися команда розробки java, були: простий, надійний, переносний, платформо-незалежний, захищений, високо-навантажуваний, багатопоточний, архітектурно нейтральний, об’єкто орієнтованим та динамічним. На початку 90-х років Джеймс Гослінг та його команда почала розробляти цю мову програмування, та в 1995 році була представлена світу.

Основними позитивними рисами цієї мови програмування можна назвати те, що вона може працювати на будь якому пристрої, тому що вона не залежить від якоїсь платформи, та для виконання коду використовується віртуальна машина java, такий архітектурний підхід дозволяє легко портувати лише java віртуальну машину під різні платформи, не чіпаючи при цьому саму мову програмування [28]. Але із-за цього мову не можна назвати як найбільш продуктивну по швидкості у виконанні, порівнюючи з такими мовами як c++ чи c#. Мабуть це є найбільш великий мінус java, в іншому ж, це є найбільш потужний інструмент для побудування високо-навантажуємих та складних

систем, де потрібна відказу стійкість та потужна модель для проектування системи.

Java виконує дві основні дії для запуску програми: компіляцію та інтерпретацію. Компілятор приводить вхідний код до байт коду, це і буду основним проміжним файлом, який можливо буде запускати на будь-якому технічному обладнанні с підтримкою віртуальною машинною java. Далі віртуальна машина java перетворює байт код до машинного коду під окрему середу виконання.

В основі написання програм на java лежить об’єктно орієнтований підхід проектування, майже все описується завдяки різноманітним класам. Тому доцільно вибрати одним з критеріїв оцінюванні розміру програми кількість класів в програмі, тому що вони найближчим чином можуть оцінити цю міру.

1.6 Висновки до розділу 1

Було проаналізована література про побудування рівняння регресії, розглянути основна концепція регресійного аналізу та методи оцінки регресійної моделі. Розглянуті основні типи регресійних рівнянь, був обґрунтований вибір нелінійного рівняння за рахунок зворотного логарифмічного перетворення. Проведено порівняння методів для пошуку викидів, та обраний метод зовнішніх стьюдентизованих видалених залишків. Приведені основні засади при побудування лінійної регресії. Зроблено обґрунтування вибору кількість класів в якості параметра для регресійного рівняння.

2 УДОСКОНАЛЕННЯ РІВНЯННЯ РЕГРЕСІЇ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМ НА МОВІ JAVA

2.1 Аналіз і перетворення вхідних даних за допомогою логарифму

Для побудови регресії потрібні незалежні та залежна зміна, в якості незалежної змінної виступають критерії які якимось чином пов’язані з залежної зміною, яку потрібно змоделювати.

Для даної кваліфікаційної роботи по оцінюванню розміру програми, залежної зміною буде виступати розмір програми в кілобайтах – це критерій який потрібно спрогнозувати. В якості незалежної змінної було обрано кількість класів у всьому проекті.

Сбір даних здійснювався з допомогою написання власного парсеру, який за допомогою RegExp, міг проходити потрібні файли з розширенням java та рахувати кількість класів в програмі, та додатково робити підсумок розміру у кілобайтах всіх пройдених файлів, в результаті записуючі усі дані в файл. Далі був використаний сервіс GitHub для пошуку інформаційні системи з відкритим кодом на java [29]. Після був написаний додатковий скрипт для завантаження проектів, та запуск парсеру для кожного з них окремо. На етапі сбіру даних було вирішено оцінювати лише розмір файлів мови програмування java, с погляду на те, що на java пишуться різні види проектів, від якихось простих консольних програм для розрахунку процесів до великих програм, як веб фреймворк, який надає велику кількість можливостей, і в багатьох них міститься різний шум, наприклад технічна документація, або приклади роботи програми, чи навіть набір музикальних файлів або відео. Ці всі атрибути є нежиттєвими для виконання та запуску програми.

Після збору інформації в цілому було зібрано 91 результатів оцінювання, в яких міститься кількість класів та розмір програми в такому випадку.

Вхідні дані наведені в таблиці 2.1.

Таблиця 2.1 – Дані які використовувалися

	x	y		x	y		x	y
1	584	1193,025	32	12153	43389,7	62	784	4000,071
2	664	3535,633	33	1160	3278,499	63	523	1236,763
3	9532	19257,98	34	556	458,0381	64	1261	5219,517
4	8126	25942,61	35	1932	3525,486	65	119	459,8135
5	247	1286,147	36	1348	1584,696	66	4294	20042,56
6	1152	3286,555	37	1493	1406,148	67	1253	4286,706
7	16583	42393,6	38	1801	3037,919	68	398	1271,903
8	548	554,582	39	243	426,3135	69	275	1387,195
9	323	946,5879	40	1190	4753,177	70	3638	10186,14
10	474	1053,607	41	15457	67276,01	71	1060	5282,948
12	575	2338,425	43	629	658,5977	73	603	3843,184
13	189	283,2617	44	3089	15763,24	74	18989	62995,21
14	469	2518,21	45	587	2669,89	75	1373	4498,627
15	138	502,9307	46	316	485,6807	76	296	209,7607
20	3269	14589,99	51	16959	65792,84	81	738	2291,439
21	872	3350,225	52	4973	19291,96	82	263	670,7236
22	5485	7137,62	53	1875	5967,104	83	1030	5317,396
23	141	345,7559	54	407	659,373	84	2926	9483,001
24	646	998,8926	55	1166	4490,313	85	277	244,4951
25	414	777,6641	56	396	1017,509	86	470	805,4063
26	472	2058,34	57	607	734,4023	87	199	728,7881
27	343	1084,804	58	4295	24529,99	88	7110	36805,3
28	3572	10765,68	59	232	444,2852	89	314	1365,45
29	1616	10991,96	60	429	1309,692	90	1059	5552,818
30	314	954,1299	61	289	1261,815	91	442	2255,766
31	1482	6283,175						

В першу чергу потрібно проаналізувати ці дані, для цього потрібно спочатку розрахувати розподіл даних. Для оцінки розподілу можливо використовувати різні методи оцінювання розподілу, для даної роботи використовувся нормальний розподіл (розподіл Ґауса).

Для оцінки розподілу, також були побудовані гістограми розподілу даних, на рисунках 2.1, 2.2 зображений розподіл вхідних даних.

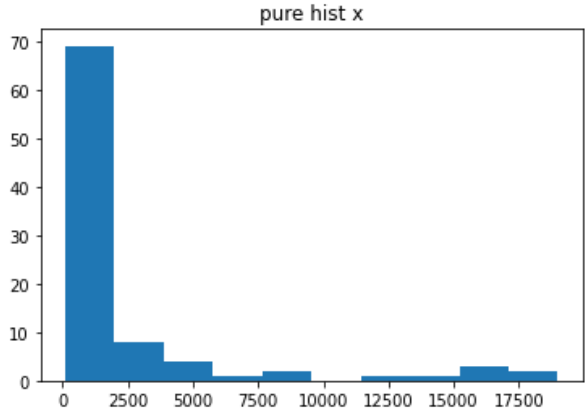


Рисунок 2.1 – Розподіл вхідних даних по кількості класів

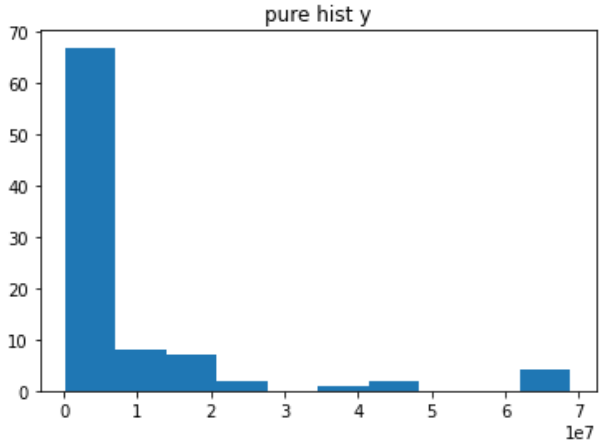


Рисунок 2.2 – Розподіл вхідних даних по розміру програм

На рисунку дуже добре видно що розподіл даних далекий від нормального, тому будування адекватної моделі не є можливим, це ґрунтується на тому, що багато статистичних методів оцінки регресії зроблені з врахуванням нормального розподілу, тобто є можливість, що навіть на таких даних віднайдене рівняння регресії буде мати дуже хороші метрики моделі, та це число не буде ні як пов’язано з реальністю та буде надавати хибне значення передбачення.

Для вирішення цієї проблеми потрібно робити перетворення даних, щоб у результаті перетворення отримати нормальний розподіл даних. Найбільш розповсюдженими методами перетворення є: перетворення Бокс-кокса, перетворення через десятковий логарифм та перетворення через натуральний

логарифм. Для вхідних даних були зроблені перетворення на основі логарифму десяткового, який дає добрий результат, та надалі на основі такого перетворення легко робити зворотне перетворення до первісних даних.

На рисунках 2.3, 2.4 зображений розподіл після застосування перетворення десяткового логарифму на вхідних даних.

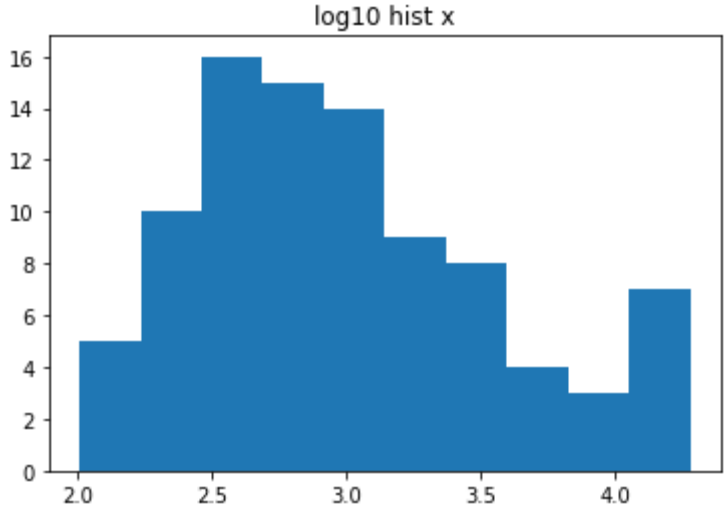


Рисунок 2.3 – Розподіл після перетворення через логарифм для кількості класів

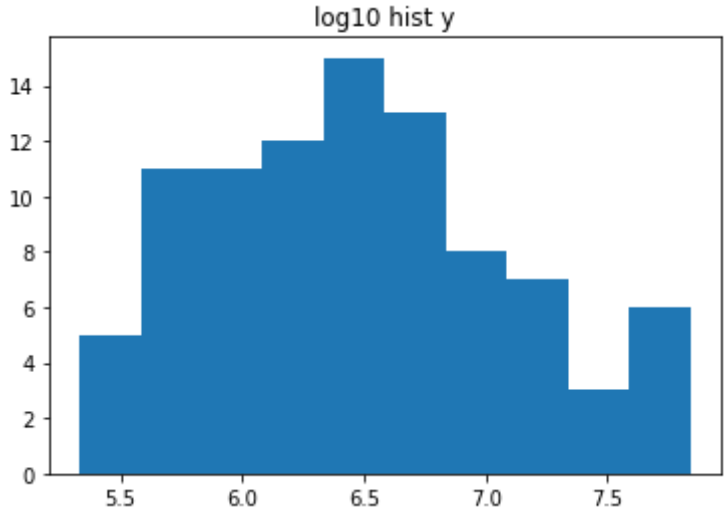


Рисунок 2.4 – Розподіл після перетворення через логарифм для розміру програми

Також результат перетворення представлений у таблиці 2.2.

Таблиця 2.2 – Результати перетворення даних

	x	y		x	y		x	y
1	2,766413	3,07665	32	4,084683	4,637387	62	2,894316	3,602068
2	2,822168	3,548467	33	3,064458	3,515675	63	2,718502	3,092286
3	3,979184	4,284611	34	2,745075	2,660902	64	3,100715	3,71763
4	3,909877	4,414014	35	3,286007	3,547219	65	2,075547	2,662582
5	2,392697	3,109291	36	3,12969	3,199946	66	3,632862	4,301953
6	3,061452	3,516741	37	3,17406	3,148031	67	3,097951	3,632124
7	4,219663	4,6273	38	3,255514	3,482576	68	2,599883	3,104454
8	2,738781	2,743966	39	2,385606	2,629729	69	2,439333	3,142138
9	2,509203	2,976161	40	3,075547	3,676984	70	3,560863	4,00801
10	2,675778	3,022679	41	4,189125	4,82786	71	3,025306	3,722876
11	3,430398	3,99827	42	3,188647	3,517444	72	3,20085	3,957371
12	2,759668	3,368923	43	2,798651	2,81862	73	2,780317	3,584691
13	2,276462	2,452188	44	3,489818	4,197645	74	4,278502	4,799308
14	2,671173	3,401092	45	2,768638	3,426493	75	3,137671	3,65308
15	2,139879	2,701508	46	2,499687	2,686351	76	2,471292	2,321724
16	2,0086	2,631949	47	2,338456	2,349644	77	3,056905	3,535669
17	3,207634	3,366735	48	2,853698	3,296275	78	2,227887	2,777413
18	3,09237	3,672226	49	4,129239	4,279945	79	3,477989	3,998332
19	3,513084	4,219629	50	2,903633	3,381706	80	4,252999	4,789566
20	3,514415	4,164055	51	4,2294	4,818179	81	2,868056	3,360108
21	2,940516	3,525074	52	3,696618	4,285376	82	2,419956	2,826544
22	3,739177	3,853553	53	3,273001	3,775764	83	3,012837	3,725699
23	2,149219	2,53877	54	2,609594	2,819131	84	3,466274	3,976946
24	2,810233	2,999519	55	3,066699	3,652277	85	2,44248	2,38827
25	2,617	2,890792	56	2,597695	3,007538	86	2,672098	2,906015
26	2,673942	3,313517	57	2,783189	2,865934	87	2,298853	2,862601
27	2,535294	3,035351	58	3,632963	4,389697	88	3,85187	4,56591
28	3,552911	4,032041	59	2,365488	2,647662	89	2,49693	3,135276
29	3,208441	4,041075	60	2,632457	3,117169	90	3,024896	3,744513
30	2,49693	2,979607	61	2,460898	3,100996	91	2,645422	3,353294
31	1482	6283,175						

Після перетворення розподіл даних виглядає значніше краще ніж було до цього. І на основі нормалізованих даних вже можливо використовувати різні методи статистичного аналізу, які допоможуть нам оцінити метрики регресійної моделі.

2.2 Побудова регресійного рівняння через зворотне перетворення

Після завершення підготовки даних, треба знайти рівняння регресії, оскільки використовувалось перетворення даних, та на основі їх робили вирахування регресійної моделі, слід зауважити, що потрібно робити зворотне перетворення для моделі, для того щоб вона враховувала перетворення змінних, в даній кваліфікаційній роботі використовувався логарифмічне перетворення, тобто якщо робиться перетворення X та Y через логарифм десятковий то рівняння представляється, як наведено в формулі (1.1). Зворотне перетворення рівняння буде виглядати, як наведено в формулі (1.2).

Для оцінки якості моделі будуть використовуватися метрики оцінки регресійної моделі такі як: $R^2, MMRE, PRED$.

Для розрахунку R^2 використовується наступна формула [3]:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \tag{2.1}$$

Де $SS_{tot} = \sum(y_i - \bar{y})^2$, а $SS_{res} = \sum(y_i - \hat{y}_i)^2$.

Для розрахунку $MMRE$ застосовується наступна формула:

$$MMRE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{2.2}$$

Та PRED має наступну форму знаходження:

$$PRED = \frac{k}{n}, \tag{2.3}$$

Де k – кількість значень з вирахування $\left| \frac{y_i - \hat{y}_i}{y_i} \right| \leq l$

l – розрахувати для $l \leq 25$.

Перейдемо до побудування моделі регресії у випадку для ненормалізованих та нормалізованих даних. Для пошуку потрібних коефіцієнтів та зсуву регресії використовувався метод найменших квадратів.

Перед побудовою даних, було зображено розкид даних на графіку (див. рис. 2.5).

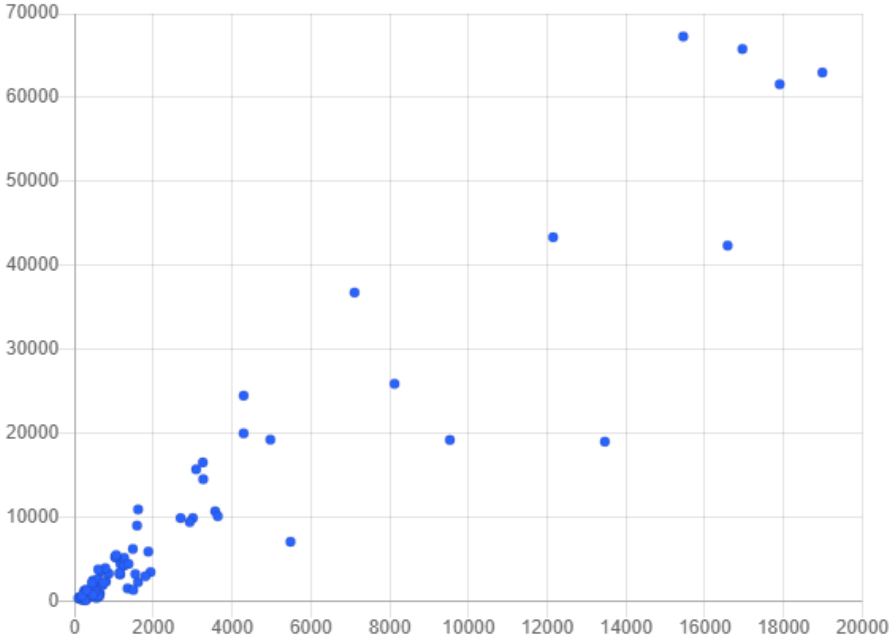


Рисунок 2.5 – Розкид вхідних даних на графіку

Для ненормалізованих даних рівняння регресії має наступну форму:

$$y = 3{,}27x + 193$$

$$R^2 = 0{,}89$$

$$MMRE = 0{,}66$$

$$PRED = 0{,}59$$

Для розрахунку R^2 використовувавсь формула (2.1), для $MMRE$ формула (2.2) та для PRED формула (2.3).

Було побудовано графік даного рівняння регресії, який зображений на рисунку 2.6.

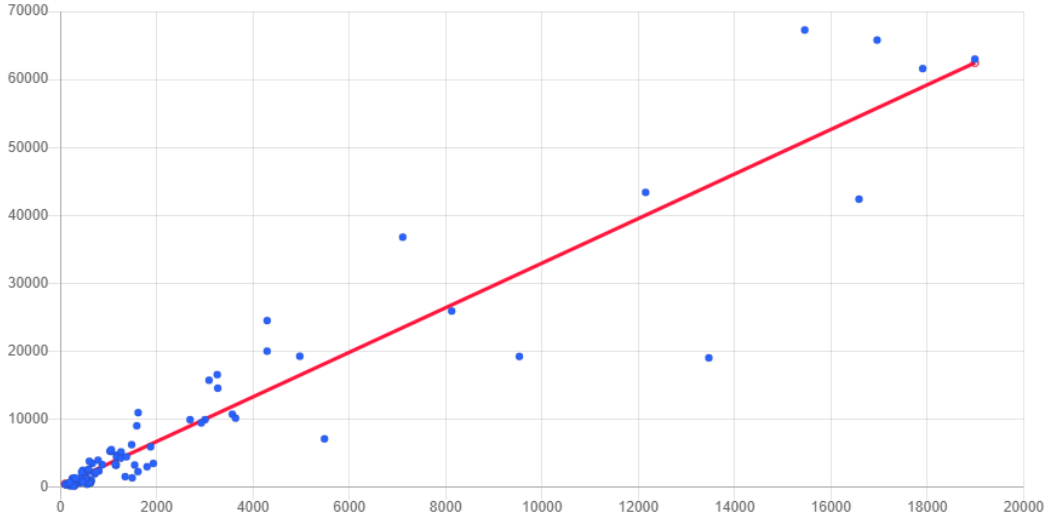


Рисунок 2.6 – Графік лінійного рівняння регресії

Далі, перед знаходженням рівняння регресії для нормалізованих даних будуть знайдені викиди за методом зовнішніх стьюдентизованих видалених залишків, формула для їх знаходження має форму рівняння, як наведено в формулі (1.4).

Після отримання значення t_i потрібно порівняти це значення з значенням 2, та якщо значення більше, то потрібно видаляти це спостереження з вибірки даних та ітеративно продовжувати цей процес. Після того, як в результаті знаходження викидів не буде нових значень, будується регресія на основі залишившихся даних.

В результаті був отриманий наступний результат (див. рисунок 2.7).

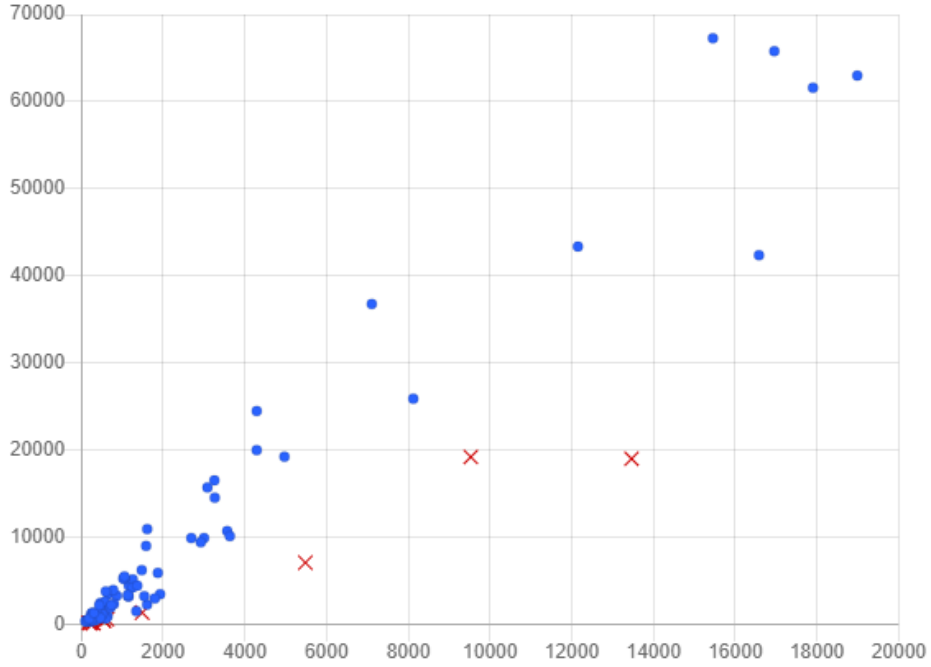


Рисунок 2.7 – Знайдені викиди для моделі регресії

В наслідок цього з вибірки вхідних даних було видалено 28 спостережень, які були позначені як викиди. Результати розрахунків стьюдентизованих видалених залишків, які були позначені як викиди приведені у таблиці 2.3.

Таблиця 2.3 – Значення стьюдентизованих видалених залишків для видалених спостережень

	x	y	t
1	2	3	4
1	6,320768	6,126952	2,382321
2	7,308543	7,24861	2,227814
3	5,690359	5,345968	2,622151
4	5,624018	5,499195	2,155247
5	6,306275	6,318215	2,384566
6	8,609772	8,873135	2,015878
7	7,206377	7,368148	2,115654
8	6,444131	6,490113	2,314773
9	5,384495	5,410254	2,287414
10	7,385851	7,752193	2,078396
11	9,507923	9,854938	2,363195
12	6,408529	6,599057	2,479066
13	5,241747	5,646371	2,030896
14	6,4708	6,906647	2,083568
15	5,755742	6,185551	2,014494

Продовження табл. 2.3

1	2	3	4
16	7,496097	8,018928	2,169739
17	6,008813	6,491289	2,187748
18	6,152733	6,691347	2,022637
19	7,387709	9,30492	2,059145
20	7,566311	8,167774	2,198028
21	5,493061	6,055175	2,211968
22	9,16241	9,865681	2,159289
23	6,025866	6,656295	2,248889
24	5,446737	6,096467	2,145931
25	6,401917	8,254056	2,01967
26	6,369901	7,084248	2,285978
27	7,342132	8,099213	2,156585
28	6,161207	6,959975	2,140132

Після знаходження викидів та розрахунку регресійного рівняння зі зворотним перетворенням десяткового логарифму, метрики оцінки регресійної моделі покращились відносно первісно знайденої моделі з ненормалізованими даними.

$$y = 10^{1,007*log(x)+0.55}$$

$$R^2 = 0,97$$

$$MMRE = 0,20$$

$$PRED = 0,82$$

Графік рівняння зображений на рисунку 2.8.

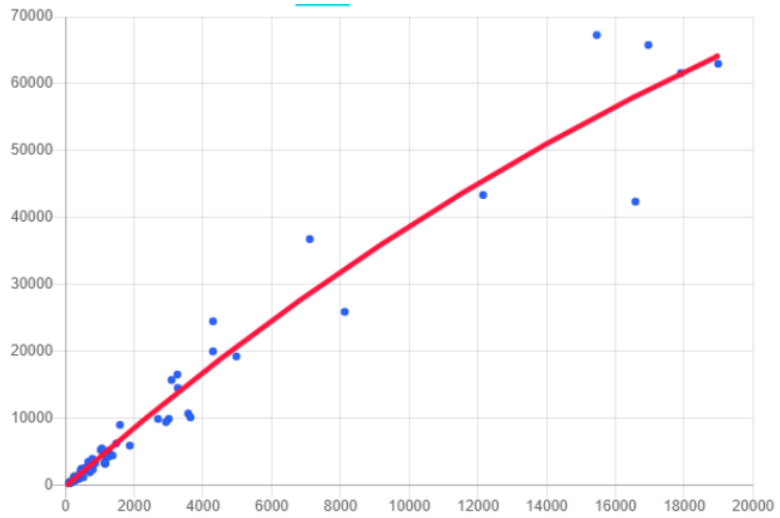


Рисунок 2.8 – Графік нелінійного рівняння регресії

Після знаходження рівняння регресії також треба побудувати довірчий інтервал, та інтервал передбачення.

Для знаходження довірчого інтервалу потрібно розрахувати стандартну похибку MSE та обрати потрібний t-стьюдент множник. Для знаходження MSE потрібні лише залишки та кількість даних до яких потрібно застосувати формулу (1.3).

Після знаходження MSE та визначення t-критерія потрібно застосувати наступну формулу для кожного значення наших вхідних даних [26]:

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

Для знаходження інтервалу передбачення нових даних не потрібно, для цього також використовується MSE, та t-критерій у формулі, для кожного пари вхідних значень [26]:

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

Формули дуже схожі за виключенням додавання одиниці у другій формулі. Далі віднайдені значення інтервалів були зображені на графіку. Зеленими точками позначений інтервал передбачення, а блакитними точками довірчий інтервал (див. рис. 2.9).

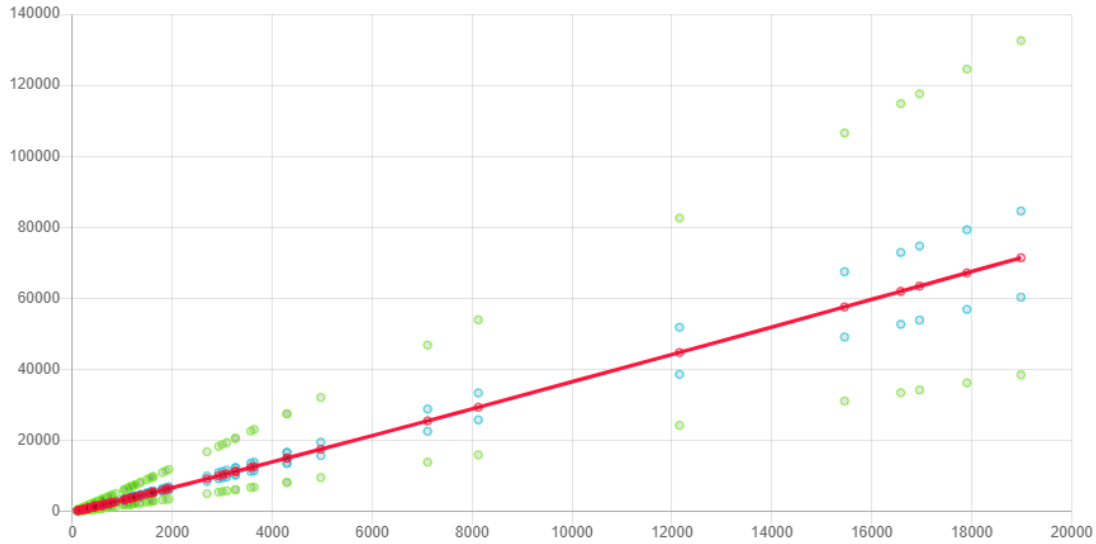


Рисунок 2.9 – Графік інтервалу передбачення та довірчого інтервалу

Також представлені результат розрахунків для довірчого інтервалу в таблиці 2.4, та для інтервалу передбачення в таблиці 2.5.

Таблиця 2.4 – Довірчий інтервал

	r-	r+		r-	r+		r-	r+
1	1738,257	2010,417	28	38656,69	51908,12	55	3910,915	4475,089
2	1993,146	2293,723	29	3585,816	4098,19	56	310,0282	403,7
3	25847,76	33433,13	30	6068,03	7044,21	57	13629,12	16686
4	687,0865	839,9019	31	4190,852	4801,951	58	3885,103	4445,058
5	3560,113	4068,517	32	5647,302	6534,599	59	1151,674	1359,974
6	52723,38	72938,49	33	675,0072	826,2122	60	772,0334	935,845
7	919,0853	1100,84	34	3682,407	4209,865	61	11532,8	13938,16
8	1390,011	1624,359	35	49149,53	67533,71	62	3264,047	3728,067
9	8511,729	10070,76	36	4821,225	5545,524	63	4962,694	5713,767
10	1709,647	1978,671	37	9776,481	11675,01	64	1798,694	2077,508
11	1374,263	1606,905	38	1747,801	2021,008	65	60354,79	84603,48
12	364,4565	468,2246	39	897,5366	1076,732	66	4271,272	4896,241
13	261,9649	346,0166	40	2152,983	2471,949	67	3521,488	4023,961
14	5043,075	5809,578	41	2431,681	2784,027	68	454,6179	573,6585
15	3833,621	4385,215	42	53916,44	74751,3	69	9510,778	11336,11
16	10320,55	12371,93	43	15797,24	19573,96	70	56921,29	79335,08
17	10352,48	12412,96	44	5884,991	6822,039	71	2229,772	2557,758
18	2659,588	3040,644	45	1179,769	1391,179	72	735,5394	894,693
19	373,1089	478,4146	46	3605,168	4120,543	73	3167,518	3617,603
20	1935,698	2229,782	47	1145,432	1353,04	74	9254,644	11010,33
21	1201,651	1415,473	48	1811,407	2091,626	75	1377,418	1610,402
22	1383,716	1617,383	49	13632,44	16690,38	76	543,1883	675,88
23	980,7982	1169,77	50	641,8564	788,5734	77	22611,89	28897,7
24	11321,77	13664,24	51	1248,624	1467,602	78	891,3781	1069,838
25	5052,783	5821,157	52	814,7544	983,9052	79	3260,821	3724,371
26	891,3781	1069,838	53	2377,162	2722,835	80	1289,404	1512,833
27	4621,908	5309,317	54	1544,741	1795,826			

Таблиця 2.5 – Значення для інтервалу передбачення

	r-	r+		r-	r+		r-	r+
1	1027,381	3401,484	28	24286,67	82621,29	55	2300,614	7607,401
2	1175,442	3889,366	29	2108,254	6970,391	56	192,4846	650,2256
3	16005,78	53991,21	30	3592,298	11898,92	57	8253,778	27552,89
4	415,8327	1387,782	31	2466,779	8158,115	58	2285,317	7556,724
5	2093,074	6920,148	32	3338,482	11053,78	59	686,8699	2280,266
6	33495,49	114808,4	33	408,7439	1364,422	60	465,6037	1551,756
7	551,4922	1834,597	34	2165,337	7159,366	61	6946,553	23140,39
8	825,2758	2735,904	35	31146,65	106568,4	62	1918,529	6342,665
9	5080,726	16871,52	36	2842,653	9405,376	63	2927,313	9686,587
10	1010,773	3346,774	37	5858,895	19481,59	64	1062,471	3517,086
11	816,1358	2705,812	38	1032,922	3419,736	65	38529,04	132529,3
12	225,0789	758,1677	39	538,9241	1793,215	66	2514,603	8316,692
13	163,5486	554,2341	40	1268,41	4195,855	67	2070,271	6844,676
14	2975,465	9846,575	41	1430,803	4731,513	68	278,748	935,5957
15	2254,819	7455,698	42	34280,85	117567,8	69	5695,036	18931,44
16	6195,008	20610,96	43	9614,41	32161,58	70	36261,46	124535,9
17	6214,763	20677,38	44	3481,774	11530,8	71	1313,114	4343,276
18	1563,92	5170,891	45	703,1969	2334,012	72	444,2378	1481,373
19	230,2454	775,2629	46	2119,686	7008,233	73	1861,743	6154,89
20	1142,053	3779,321	47	683,2424	2268,325	74	5537,263	18401,99
21	715,9103	2375,863	48	1069,853	3541,408	75	817,967	2711,841
22	821,6228	2723,877	49	8255,856	27559,92	76	331,1612	1108,615
23	587,4576	1953,006	50	389,2729	1300,247	77	13935,35	46890,22
24	6815,479	22698,84	51	743,1951	2465,683	78	535,331	1781,384
25	2981,282	9865,905	52	490,5874	1634,044	79	1916,63	6336,387
26	535,331	1781,384	53	1399,004	4626,591	80	766,8753	2543,638
27	2723,56	9009,965	54	915,0679	3031,564			

Потрібно нагадати для чого потрібні дані інтервали. Для того щоб відповісти на це, найпростіше представити два питання для аналогії.

Перший, якщо потрібно знайти середній розмір всіх програм, які містять від 1000-3000 класів, для цього доцільно розрахувати довірчий інтервал, в якому в 95 відсотків буде середнє значення цієї вибірки.

Другий, якщо потрібно знайти розмір конкретній програми яка містить 2000 класів, для визначення цього довилось би розрахувати інтервал

передбачення, в якому буде міститися значення розміру програми в 95 відсотках випадках.

Довірчий інтервал буде менший за інтервалу передбачення, тому що там використовується середнє значення, що трохи згладжує розкид даних, та дає більш точні результати.

Дані інтервали були розраховані тільки для нормалізованих даних, оскільки t-критерій покладається на нормальний розподіл даних, і при використані ненормальних даних результати будуть не актуальні.

Також після побудови рівняння регресії рекомендується розрахувати та знайти залишки та відобразити їх на діаграмі. Залишками – є значення різниці для кожної точки відносно передбаченим значенням та фактичним. З діаграми залишків буде зрозуміло, чи має наша модель лінійну залежність між значеннями, в такому випадку діаграма залишків буде розрізнена та хаотична.

На рисунку 2.10 зображений графік залишків для нормалізованих даних.

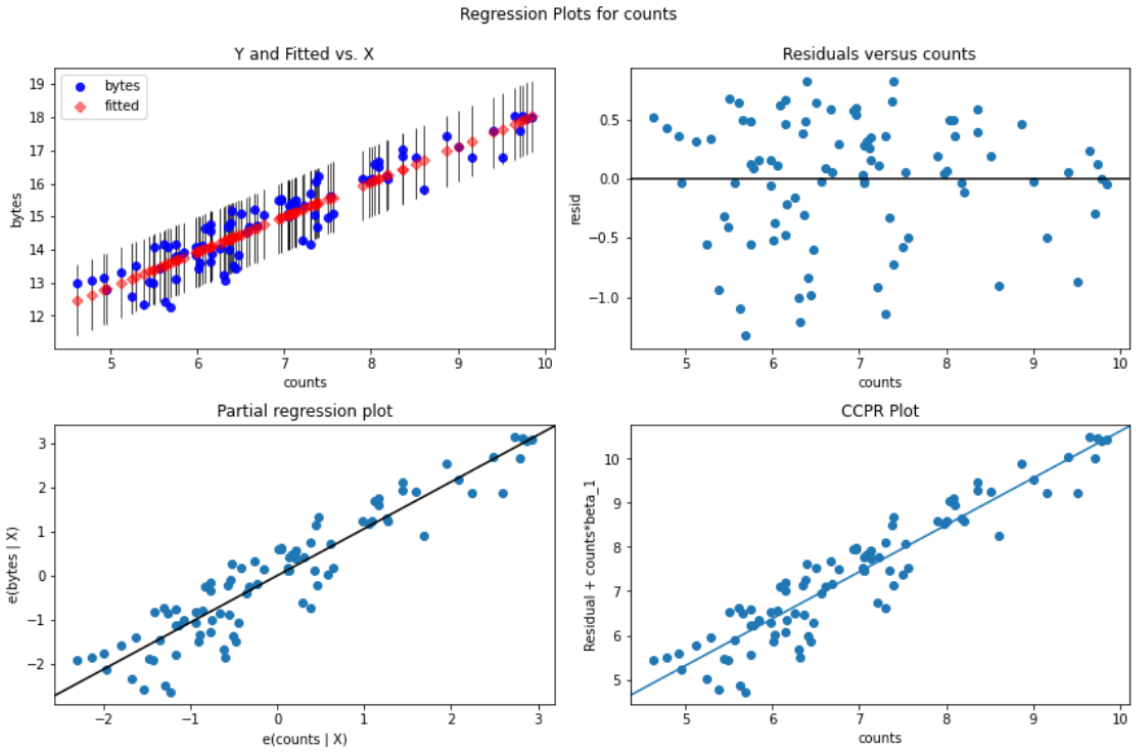


Рисунок 2.10 – Графік розкиду залишків

На рисунку зображений хаотичний розкид точок, що підвереджує лінійне взаємовідношення між даними.

2.3 Результати вдосконалення рівняння регресії для оцінювання розміру програм на мові java

Якщо порівнювати регресійну модель яка була побудована на ненормалізованих даних з регресійною моделлю з зворотним перетворенням та знайденими викидами, коефіцієнт детермінації системи виріс на 8%, та середня величина відносної похибки зменшилась на 46%, також зріс показник рівня прогнозування на 0,23. Таким чином були нормалізовані дані, зроблене зворотне перетворення та знайдені викиди для моделі та в результаті чого була покращена модель регресії по різним показникам, що доказують результуючі метрики регресійної моделі.

Результуючі дані приведені у таблиці 2.6.

Таблиця 2.6 – Результати метрик рівнянь регресії

	Лінійна ненормалізовані дані	Нелінійна з знайденими викидами
Рівняння	$y = 3{,}27x + 193$	$y = 10^{1{,}007*log(x)+0.55}$
R^2	0,89	0,97
$MMRE$	0,66	0,20
$PRED$	0,59	0,82

Була приведена таблиця результатів роботи, яка доказує ефективність перетворення та знаходження викидів.

2.4 Постановка задачі на розробку програмного забезпечення

Після розглянутих особливостей побудування регресійного рівняння, а також аналізу різних типів регресії та шляхів пошуку викидів, сформулюємо наступну постановку задачі: на основі побудованої регресійної моделі,

розробити програмний додаток для покращення оцінювання розміру програмного забезпечення інформаційних систем на java.

2.4.1 Вимоги до функціональних характеристик

До програмного продукту є ряд функціональних вимог, котрими може скористуватися користувач. Система повинна давати користувачу можливість завантажити дані різними шляхами та попереджати про помилки в процесі виконання.

При аналізу предметної галузі були виділені основні функціональні вимоги до програмного продукту:

- завантаження даних до програми;
- розрахунок розподілу даних;
- застосування перетворення даних до вхідних спостережень;
- розрахунок лінійної регресії;
- розрахунок довірчого інтервалу та інтервалу передбачення;
- обчислення нелінійної регресії за рахунок зворотного перетворення;
- розрахування основних метрик регресійної моделі;
- зображання графіків функцій регресійного рівняння на екрані;
- знаходження вибіркового середнього, інтервалу передбачення та довірчого інтервалу від побудованої моделі регресії для окремо введених даних;
- пошук викидів за допомогою видалених залишків.

2.4.2 Вимоги до організації вхідних та вихідних даних

Тип та структура даних, яка використовується у якості вхідних даних:

Для завантаження або введення даних до програми використовується JSON формат, який є найбільш розповсюджений для веб-застосунків, завдяки вбудованим інструментам для роботи з такими файлами в мові JavaScript, та простота формування такого формату було вирішальним при виборі даного формату даних.

Вихідних даних не передбачено.

2.4.3 Вимоги до складу та параметрів технічних засобів

Для завантаження та правильної роботи програмного продукту потрібен персональний комп’ютер з постійним інтернет з’єднанням, для того щоб завантажити веб-застосунок з серверу.

Вимоги до комп’ютерного технічного оснащення наведено нижче:

Для PC:

- Windows 7 і більш пізні;
- Процесор Intel core i3 2,33, AMD Ryzen 3(або аналогічний);
- 1 Гб оперативної пам'яті;
- 512 Мб відео пам'яті;
- Firefox 2.x-3.x, Opera 9.5 або більш пізньої версії, Google Chrome.

Для Macintosh:

- Mac OS X v10.4 і пізніші;
- Процесор IntelCore i3 з тактовою частотою не менше 1,8 ГГц;
- 512 Мб оперативної пам'яті;
- 512 Мб відео пам'яті;

- Браузери: Firefox 2.x, Firefox 3.x, Opera 9.5, Safari 3.x

Для Linux:

- Процесор Intel core i3 2,33, AMD Ryzen 3(або аналогічний);
- 1 Гб оперативної пам'яті;
- 512 Мб відео пам'яті;
- Браузери: Firefox 2.x, Firefox 3.x, SeaMonkey 1.11;

Для Android:

- 512 Мб оперативної пам'яті;
- Android 4.0 або більш пізньої версії.

Для iOS:

- 512 Мб оперативної пам'яті;
- iOS 7.0 або більш пізньої версії.

Були наведені основні технічні вимоги до застосування програмного продукту.

2.4.4 Вимоги до інформаційної та програмної сумісності

Програмний продукт потребую наявність інтернет браузера з доступом до інтернету. Вимого щодо операційної системи не висуваються.

2.5 Висновки до розділу 2

Було проведено сбір та аналіз вхідних даних, після чого приведена гістограма нормальності розподілу даних. Для усунення зсуву розподілу було застосовано десятковий логарифм для вхідних даних, що показало добру ефективність щодо приведення даних до нормального розподілу. Були знайдені викиди для нормалізованих даних за методом стьюдентизованих видалених залишків, завдяки чому було знайдено 28 викидів. Було побудовано нелінійну регресію через зворотне перетворення десяткового логарифму лінійного

регресійного рівняння, також для нелінійного рівняння було побудовані інтервал передбачення та довірчий інтервал, які допоможуть оцінити похибку при прогнозуванні даних. В результаті було покращено регресійне рівняння, про що свідчить остаточні результати порівняння.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ НА JAVA

3.1 Ескізний проект програмного забезпечення

Метою ескізного проекту є зазначення основних функції або ідей на верхньому рівні, щоб дати загальну ідею рішення та функцій які лягають в основі подальшої розробки по.

В подальшому можливо на основі ескізного проекту вирішити, чи є доцільним розробка робочої документації чи технічного проекту. Тобто на стадії розробки ескізного проекту ведеться розгляд основних частин виробу або його різних варіантів у цілому. А деталізація рішень має буду опрацьована в тій мірі, щоб мати можливість порівняти різні варіанти між собою.

3.1.1 Розробка діаграми варіантів використання

Для позначення основних функцій програми на верхньому рівні було спроектовано діаграму використання в UML вигляді. Основної метою діаграми використання зображення відображення системних функцій, його оточення та зв’язки між сутностями в цілому.

Діаграма варіантів використання дає уявлення які в нашій системі є сутності, хто є основними актори програми, так які вони мають основні повноваження відносно основних функцій системи.

Після аналізу функціональних вимогу до програмного забезпечення, було відібрані наступні можливі варіанти використання:

- Завантаження статистичних даних;
- Корегування даних близьких до нормального розподілу;

- Зображення графіку розкиду точок на графіку;
- Розрахування моделі лінійної регресії;
- Розрахування моделі нелінійної регресії;
- Зображення графіку моделей;
- Розрахування різних оцінювальних метрик для аналізу регресії;
- Знаходження викидів для регресії;
- Оцінка розміру пз та додатково інтервал передбачення та довірчий інтервал.

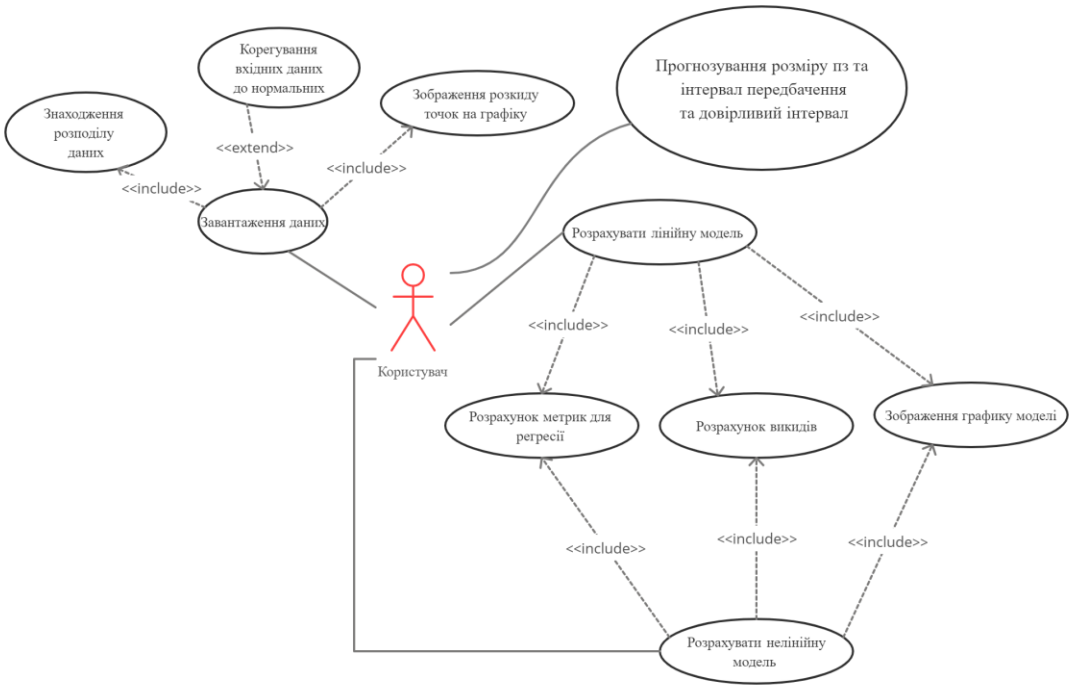


Рисунок 3.1 – Діаграма варіантів використання програмного додатку для розрахунку регресійної моделі

На основі побудованої діаграми варіантів використання ми маємо змогу опрацювати та деталізувати кожний пункт окремо.

3.1.2 Специфікація варіантів використання

Після розгляду діаграми варіантів використання, наведем специфікації для основних функцій в таблиці 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9.

Таблиця 3.1 – Специфікація варіанту використання «Завантаження даних»

Призначення	Варіант використання призначений для завантаження даних до системи
Учасник	Користувач, система
Передумова	Дані у правильному форматі
Короткий опис	Дозволяє завантажити дані до системи
Процедура	– користувач обирає потрібний файл в файловій системі; – система намагається прочитати дані; – система інформує користувача.
Альтернативна процедура	– користувач копіює дані з файлу та вставка їх до полю вводу даних.
Аварійні умови	Якщо в даних є помилка формату, система виводить інформацію о помилки
Пост умова	Після успішного завантаження даних, користувачу розблокується інші функції

Таблиця 3.2 – Специфікація варіанту використання «Знаходження розподілу даних»

Призначення	Варіант використання призначений для розрахунку розподілу даних
Учасник	Користувач, система
Передумова	Завантажені дані
Короткий опис	Дозволяє подивитися розподіл даних
Процедура	– користувач натискає кнопку для визначення розподілу; – система розраховує дані.
Альтернативна процедура	Не виділена.
Аварійні умови	Система інформує користувача о помилки.
Пост умова	Після успішного завантаження даних, користувачу надаються нові варіанти використання.

Таблиця 3.3 – Специфікація варіанту використання «Корегування вхідних даних»

Призначення	Варіант використання призначений для корегування вхідних даних.
Учасник	Користувач, система.
Передумова	Дані повинні бути не нормального розподілу.
Короткий опис	Даний варіант, дає змогу користувачу зробити деякі перетворення над даними, для корегування розподілу.
Процедура	– Користувач натискає на потрібній варіант корегування; – Система намагається застосувати обраний варіант перетворення.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	При успішному корегуванні, користувачу буде наведена інформація про результат.

Таблиця 3.4 – Специфікація варіанту використання «Зображення розкиду точок на графіку»

Призначення	Варіант використання призначений для зображення розкиду точок на графіку.
Учасник	Користувач, система.
Передумова	Дані повинні бути завантаженими.
Короткий опис	Даний варіант, дає змогу користувачу побачити як розкидані точки на графіку.
Процедура	– користувач перемикає фільтр що до розкиду; – система малює точки на графіку.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Зображення точок на графіку.

Таблиця 3.5 – Специфікація варіанту використання «Розрахунок лінійної регресії»

Призначення	Варіант використання призначений для розрахунку лінійної регресії.
Учасник	Користувач, система.
Передумова	Дані повинні бути завантаженими.
Короткий опис	Даний варіант, дає змогу користувачу розрахувати основні важелі лінійного регресійного рівняння.
Процедура	– користувач натискаю кнопку; – система вираховує коефіцієнти регресійного рівняння; – система робить розрахунок коефіцієнту детермінації моделі.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Вивід основної інформації щодо моделі.

Таблиця 3.6 – Специфікація варіанту використання «Розрахунок нелінійної регресії»

Призначення	Варіант використання призначений для розрахунку нелінійної регресії.
Учасник	Користувач, система.
Передумова	Дані повинні бути завантаженими.
Короткий опис	Даний варіант, дає змогу користувачу розрахувати основні важелі нелінійного регресійного рівняння.
Процедура	– користувач натискаю кнопку; – система вираховує коефіцієнти регресійного рівняння; – система робить розрахунок коефіцієнту детермінації моделі.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Вивід основної інформації щодо моделі.

Таблиця 3.7 – Специфікація варіанту використання «Зображення графіку регресійної моделі»

Призначення	Варіант використання призначений для зображення графіку регресійної моделі.
Учасник	Користувач, система.
Передумова	Прорахована лінійна/нелінійна регресійна модель.
Короткий опис	Даний варіант, дає змогу користувачу побачити як модель відображається на графіку.
Процедура	– користувач натискає на потрібний фільтр; – система малює графік регресійної моделі.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Зображення графіку регресійної моделі.

Таблиця 3.8 – Специфікація варіанту використання «Розрахунок викидів моделі»

Призначення	Варіант використання призначений для розрахунку викидів моделі.
Учасник	Користувач, система.
Передумова	Прорахована лінійна/нелінійна регресійна модель.
Короткий опис	Даний варіант, дає змогу користувачу розрахувати викиди системи.
Процедура	– користувач натискає на потрібний кнопку; – система розраховую викиди.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Вивід інформації щодо викидів.

Таблиця 3.9 – Специфікація варіанту використання «Прогнозування розміру пз, інтервал передбачення та довірчий інтервал»

Призначення	Варіант використання призначений для вивід інформації в таблицю.
Учасник	Користувач, система.
Передумова	Прорахована лінійна/нелінійна регресійна модель.
Короткий опис	Даний варіант, дає змогу користувачу зробити прогноз розміру програми.
Процедура	– користувач натискає на потрібний кнопку; – система запитує потрібні дані; – система на основі моделі виводить результат та рахує інтервал передбачення, довірчий інтервал.
Альтернативна процедура	Не виділена.
Аварійні умови	Якщо в процесі виникла помилка, система інформує про це користувача.
Пост умова	Вивід інформації щодо передбачення розміру.

Були наведені основні специфікації для варіантів використання програмного забезпечення.

3.1.3 Розробка діаграм діяльності прецедентів програмного забезпечення

Діаграма діяльності – в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Для представлення послідовності дій основних варіантів використання застосовується діаграма діяльності. Вони уявляють с себе граф діяльності, це також є різновидом графу станів скінченного автомату, вершинами якого є дії, а перехід відбувається по завершенню дії. Нижче було наведено головні варіанти використання.

Діаграма діяльності для прецеденту «Завантаження даних» представлена на рисунку 3.2.

Рисунок 3.2 – Діаграма діяльності прецеденту «Завантаження даних»

Діаграма діяльності для прецеденту «Розрахунок моделі регресії» представлена на рисунку 3.3.

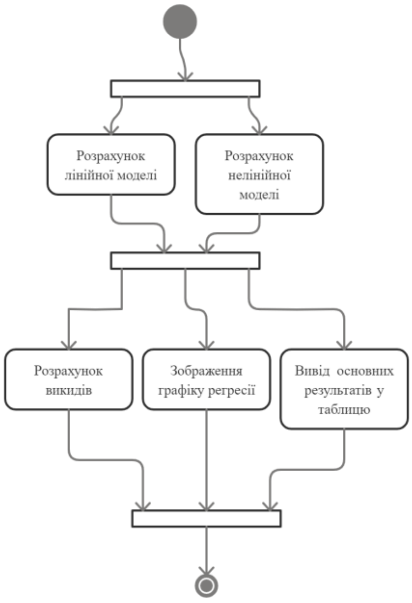


Рисунок 3.3 – Діаграма діяльності прецеденту «Розрахунок моделі регресії»

Була побудована діаграма діяльності, яка зможе більш детально зрозуміти основний спосіб використання програмного забезпечення.

3.1.4 Інформаційна модель

Інформаційна модель програмного забезпечення «Удосконалення рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на Java» розроблена діаграма класів, що утворена виходячи з діаграми варіантів використання і створеної діаграми діяльності для цього.

На рисунку 3.4 наведено структуру основних діючих сутностей за допомогою діаграми класів

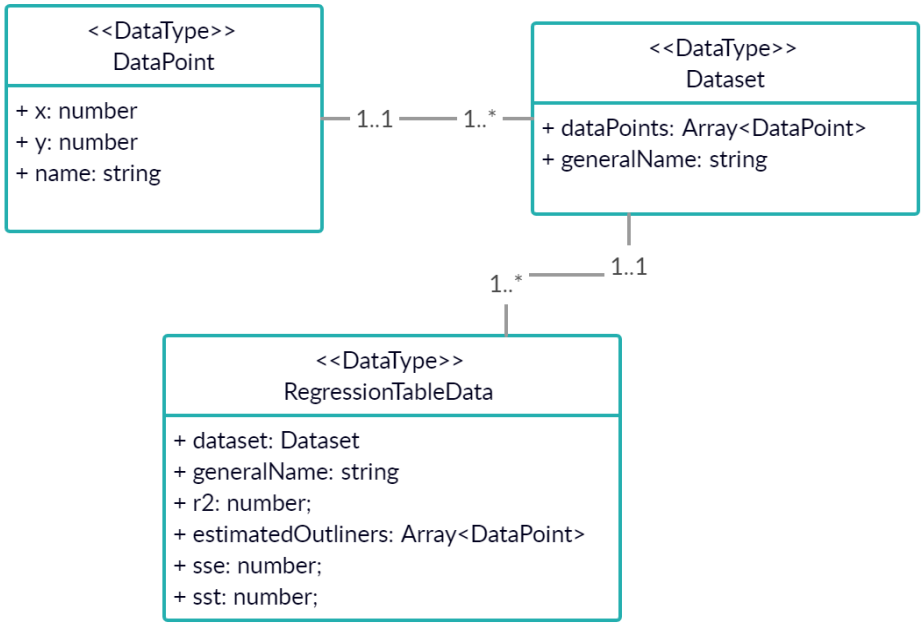


Рисунок 3.4 – Діаграма класів

Побудована діаграма класів допоможе спроектувати більш консистентні дані, там дасть нам загальне уявлення з якими даними доведеться працювати програмі.

3.1.5 Розробка інтерфейсу програмного забезпечення

Інтерфейс – це головна річь для взаємодії програмного коду та користувача, він має містити зрозумілий і легку в освоєні структуру, в ідеалі, щоб користувач інтуїтивно міг розуміти, за що відповідає різні компоненти інтерфейсу, і як досягти своєї мети, без доцільного читання інструкції. Але це все може не враховуватися для професійних програм, де є велика кількість різноманітних професійних вимог, і де дуже складно розробити такий інтерфейс, що одразу буде зрозумілий користувачу, так як потребує деякої системності, ніж зручність користування.

В розділі приведений головний користувацький інтерфейс для розрахуванні моделі регресії та її аналізу.

Спочатку користувач повинен завантажити вхідні дані до програми, для цього він має дві опції, перша – скопіювати json текст у поле для вводу, та натиснути кнопку «Зчитати з поля», або завантажити файл натиснувши кнопку «Завантажити файл» та система запропонує обрати файл с файлової системи. Інші функції програми, як: розрахунок лінійної, розрахунок нелінійної, пошук викидів для лінійної і нелінійної та обробка вхідних даних – не доступні доки користувач не надитиме вхідні дані.

На рисунку 3.5 зображений основний вигляд інтерфейсу програми, при не введених вхідних даних.

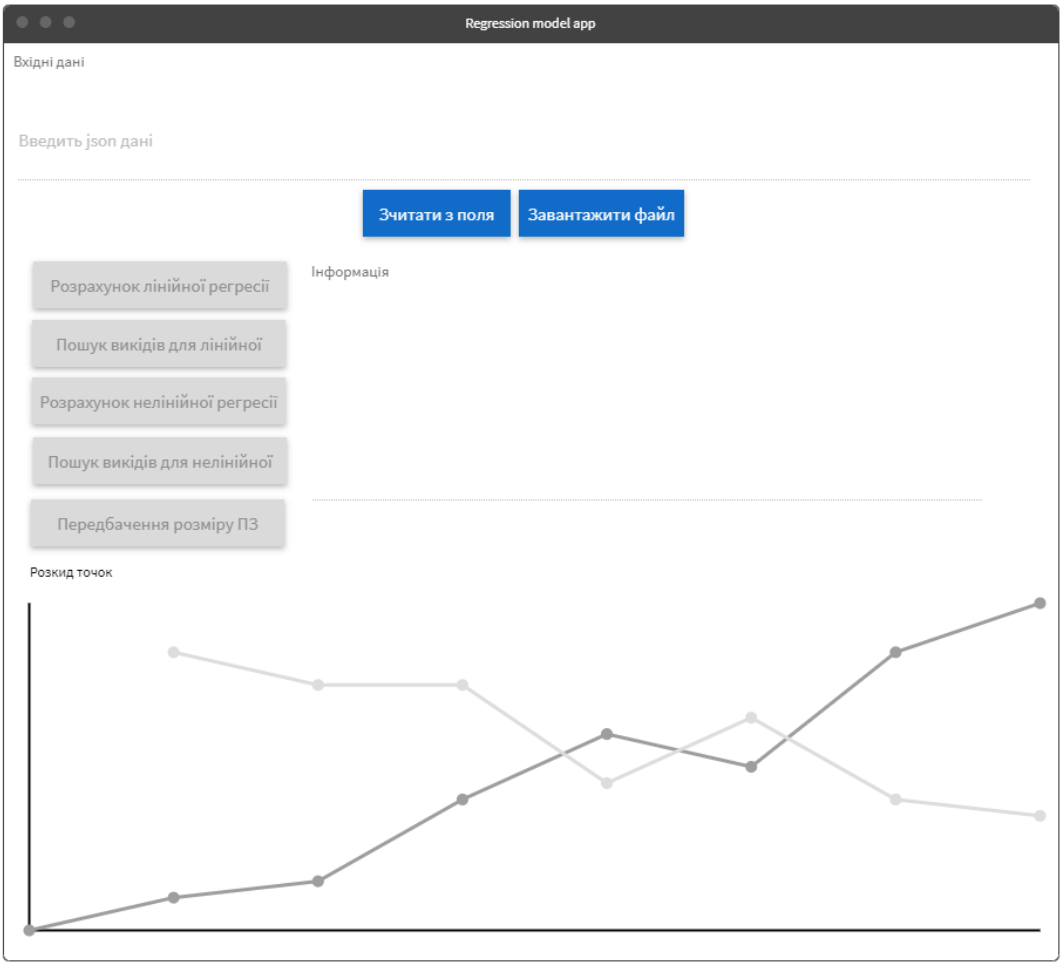


Рисунок 3.5 – Головний екран додатку без завантажених даних

Після того, як користувач надав необхідні дані, програма малює вхідні точки на графіку, та виводить текстову інформацію у поле, що дані було зчитано, і після цього розблокуються кнопки розрахунку моделі лінійної та нелінійної регресії, слід зауважити, що пошук викидів для моделі доступний тільки після розрахунку якоюсь с двох моделей.

Ці зміни зображені на рисунок 3.6.

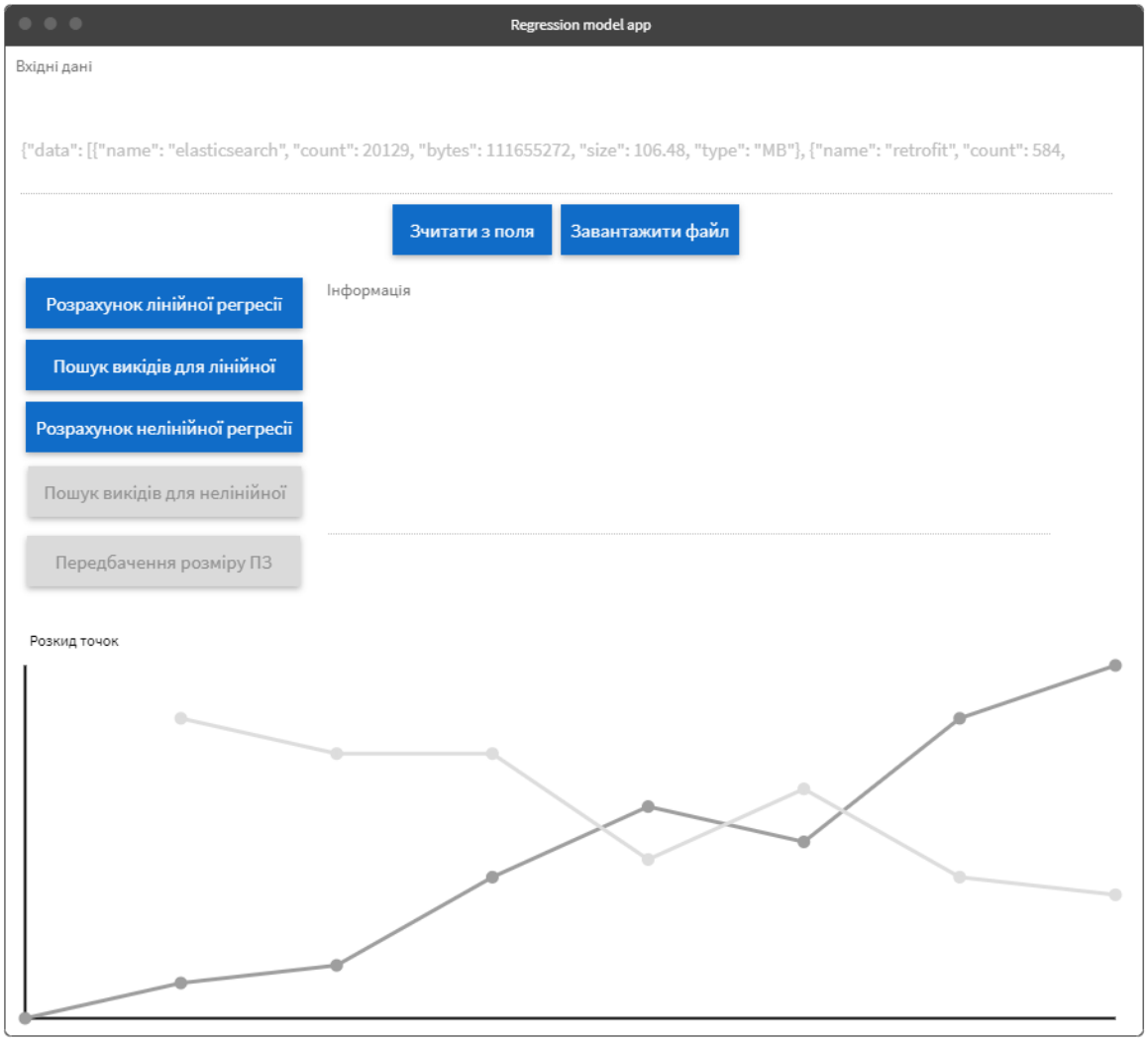


Рисунок 3.6 – Головний екран додатку з завантаженими даних

Був побудований основний макет інтерфейсу за яким надалі буде будуватися користувацький інтерфейс основної програми.

3.2 Технічний проект програмного забезпечення

Після аналізу ескізного проекту, був спроектований та розроблений технічний проект програмного продукту, с допомогою побудови статичних та динамічних моделей. Динамічна модель була представлена у вигляді діаграми взаємодії, а статична за допомогою діаграми класів.

3.2.1 Структура класів

С висновку моделі варіантів використання, проекту програмного інтерфейсу, інформаційної моделі, була розроблена статична модель, яка має форму діаграми класів, і наведена на рисунку 3.7:

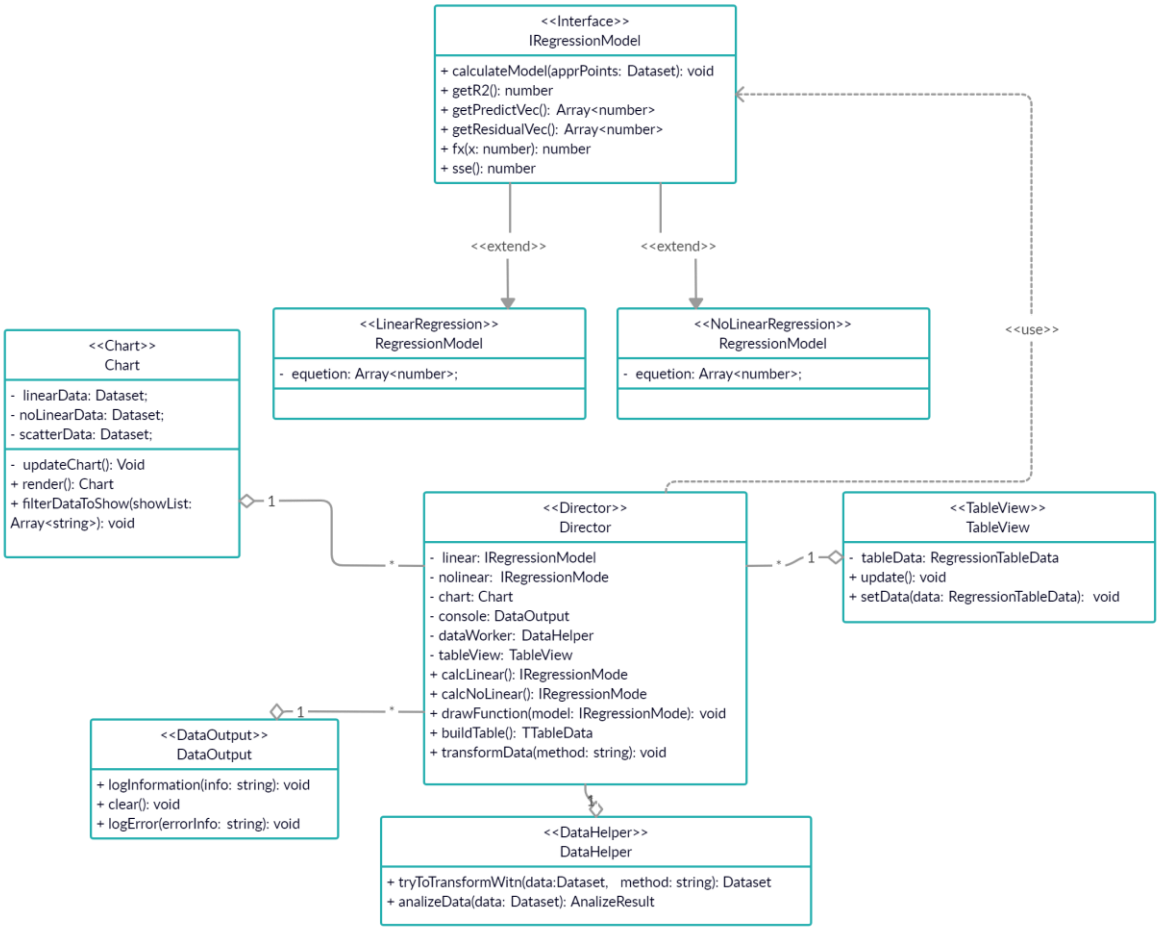


Рисунок 3.7 – Статична модель у вигляді діаграми класів

Даний етап дає нам основне уявлення о бізнес логікі програмного забезпечення, добре спроектований система має бути легко підтримувана та подальше розширюватися, для цього перед проектуванням потрібно виділити основні точки зміни програми, щоб побудувати на цьому місці достатньої кількості абстракції, які потім можна використовувати для зміни або додавання нових модулів програми без зміни великої кількості програмного коду.

Для побудування таких систем можна виділити різні підходи, які можуть спростити нам основну роботу над програмою та поліпшують подальшу підтримку програми. До цих методів можна виділити як і основні об’єктно орієнтовані концепції та різні патерни проектування.

3.2.2 Специфікація класів

Інтерфейс: «IRegressionModal»

Відповідальність: Даний інтерфейс відповідаю за основні функції регресійної моделі, будь то лелійна модель або нелінійна, і інкапсулює користувача від деталей та методів конкретної моделі.

Операції:

calculateModel() – функція розрахунку регресійної моделі, на основі переданих аргументів с даними точками;

getR2() – функція яка вертає коефіцієнт детермінації для моделі;

getPredictVec()– функція яка вертає вектор значень залежною зміною, які розрахувала модель;

getResidualVec() – функція що повертає вектор значень, що відповідають за різницю фактичних та передбачених значень;

fx() – функція розрахунку залежною змінної за переданими незалежними змінними;

getSse() – розрахувати і повернути SSE для даної моделі.

Клас: «LinearRegression»

Відповідальність: Клас відповідає за розрахунки для лінійної регресійної моделі.

Атрибути: equetion

Операції є реалізацією інтерфейсу «IRegressionModal» всі операції виконуються в контексті розрахунку лінійної моделі.

Клас: «NoLinearRegression»

Відповідальність: Клас відповідає за розрахунки для нелінійної регресійної моделі.

Операції є реалізацією інтерфейсу «IRegressionModal» всі операції виконуються в контексті розрахунку нелінійної моделі.

Клас: «Chart»

Відповідальність: Клас відповідає за зображення даних на графіку, та фільтрування потрібних даних.

Атрибути: linearData, noLinearData, scatterData

Операції:

updateChart() – функція оновлення внутрішніх даних класу, для зображення нових даних на графіку;

render() – функція малює графік за даними які встановленні в атрибутах, та фільтрами які активні;

filterDataToShow()– функція яка застосовує передані фільтри до графіку, тобто вимкнення або показу якоїсь моделі або точок.

Клас: «TableView»

Відповідальність: Клас відповідає за відображення даних у таблиці.

Атрибути: RegressionDataTable

Операції:

update() – функція оновлення внутрішніх даних класу, для відображення нових даних в таблиці;

setData() – функція встановлю нові дані для класу.

Клас: «DataOutput»

Відповідальність: Клас який виводить основну інформацію в текстовому вигляді, і вед список зроблених операцій.

Операції:

logInformation() – функція виводу переданого рядку в поле виводу даних;

clear() – функція очистки поточного поля виводу даних;

logError() – функція яка викликається у разі помилки, та друкує рядок червоним кольором.

Клас: «Director»

Відповідальність: Головний клас який концертує в собі всі інші класи для їх комунікації та св’язку.

Атрибути: linear, nolinear, chart, console, tableView, inputData

Операції:

calcLinear() – функція розрахунку лінійної регресії, яка делігує роботу до класу лінійної регресії;

calcNoLinear() – функція розрахунку нелінійної регресії, яка делігує роботу до класу нелінійної регресії;

drawFunction() – функція для зображення моделі на графіку;

buildTable() – функція яка формує потрібні дані для класу таблиця;

transformData() – функція редагування даних для нормального розподілу.

Клас: «DataHelper»

Відповідальність: Клас який може аналізувати вхідні дані.

Операції:

analizeData() – функція аналізу даних;

tryToTransformData() – функція яка застосовує якийсь метод трансформації до даних.

3.2.3 Динамічна модель програмного забезпечення

Динамічна модель відображає зміну програми, які відбуваються у деякій час, або специфічний фунціонал, який потрібно деталізувати. Динамічні моделі також можуть називати функціональними.

Динамічна модель була зображена за допомогою діаграми послідовності, це різновид UML діаграми, яка відображає взаємодію між об’єктами системи з дією часу. На діаграмі у вигляді вертикальних ліній та прямокутників зображають процеси та об’єкти, що існують в одну одиницю часу.

Завантаження первісних даних зображено на рисунку 3.8

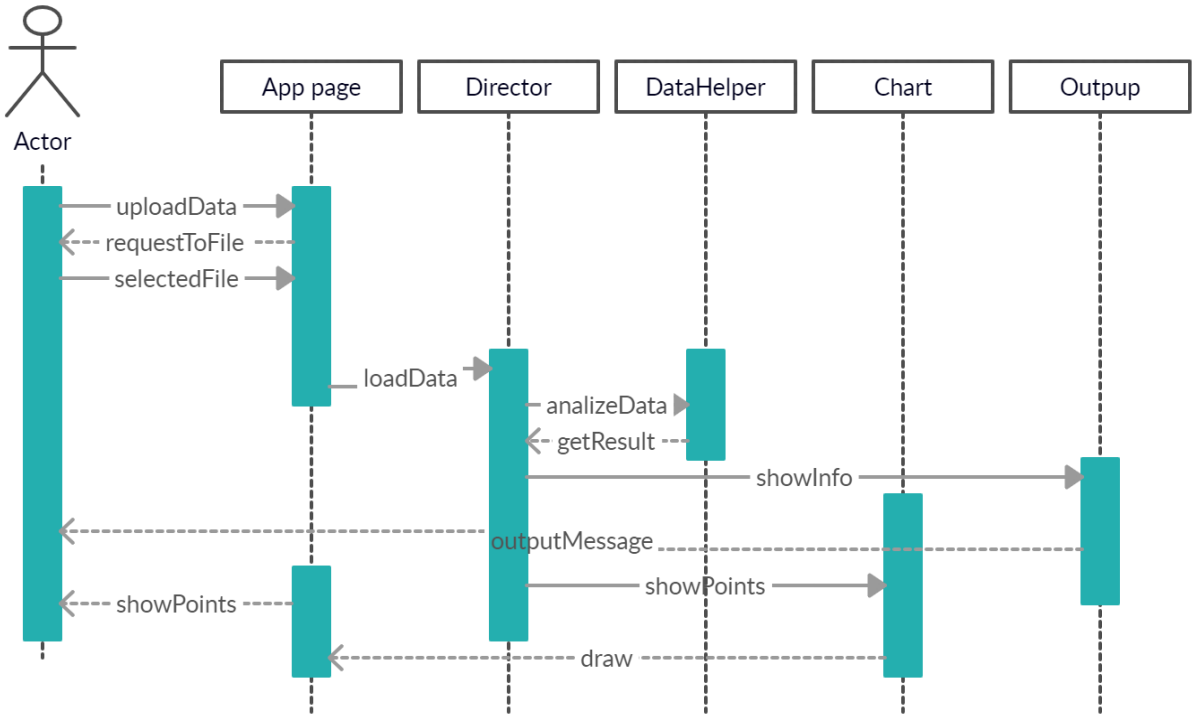


Рисунок 3.8 – Діаграма послідовності завантаження первісних даних

На рисунку 3.9 показано послідовність розрахунку регресійної моделі, та зображення її на екрані, для випадку як і лінійної так і нелінійної.

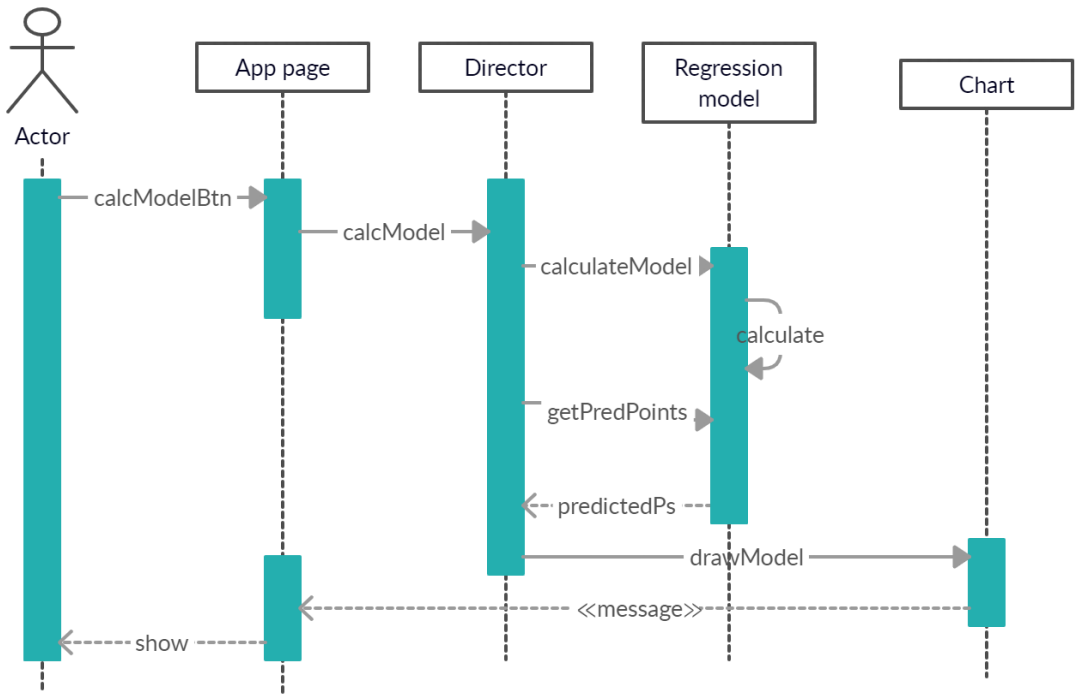


Рисунок 3.9 – Діаграма послідовності розрахунку та виводу графіка регресійної моделі

Були побудовані діаграми послідовності, які дають розуміння як взаємодіють різні сутності програми.

3.3 Робочий проект програмного забезпечення

3.3.1 Обґрунтування вибору мови та системи програмування

Для розробки програмного забезпечення для даного дипломного проекту була обрана мова Javascript, тому як це є найпопулярніша мова програмування, с допомогою якої легко створювати сучасні користувацькі інтерфейси, для формування інтерфейсу також використовувалась найбільш поширена

бібліотека React, за допомогою неї розробник не буде зациклюватися на з’єднані моделі даних та його відображенню (користувацького інтерфейсу), тому що бібліотека пропонує зручний інтерфейс для маніпулюванням даних та їх динамічному оновлені на екрані. Також для більш комфортній розробці використовувався TypeScript який розширюю звичайний Javascript та додає йому типізацію, яка спрощує подальшу підтримку програми.

Середовищем для розробки було обрана проект с відкритим вихідним кодом Microsoft Visual Code. Підставами для обрання цього середовища є його безкоштовне використання, та середу де все можливо налаштувати під себе. Також в ній міститься безліч додаткових плагінів, які спрощують розробку. З коробки середовище має підтримку Javascript та Typescript, та може бути налаштований аналізатор та форматор коду, який форматує код згідно правил, які були встановлені, також середа розробки може відслідковувати невикористані зміні та блоки коду які можливо спростити.

React забезпечує дуже легку та практичну до застосування архітектуру MVVM яка забезпечує одно направлену прив’язку даних, завдяки чому розробнику не потрібно замислюватися об оновленнях зображенні на екрані та різних значень на ньому. Це досягається завдяки інкапсульованому доступу до відображення та абстракцій, які нам дає react, одними з них є jsx, який є так званими синтаксичним цукром для розробники, він допомагає описувати інтерфейси програми більше декларативно, та відокремлює нас від html, хоча він і схожий на звичайний html, в процесі обробки коду це перетворюється на звичайні об’єкти якими оперує вже React, таким чином ми не прив’язані до html, та можемо таким чином описувати одні і ті самі елементи інтерфейсу для різних платформ, наприклад android чи ios. Тобто наш jsx код може бути інтерпретований на компоненти мобільної платформи, та працювати як «нативні» компоненти. Така реалізація вже міститься в React Native, який дозволяє розробляти додатки одразу на три платформи, як звичайний веб додаток, додаток для android і ios.

Для оновлення даних на екрану використовується так званий алгоритм звірки, тобто в нашому компоненті є так звані вхідні параметри, як у звичайній функції і так само у компонента може бути свій окремий стан, в якому він може зберігати різні специфічні дані потрібні йому для відображення контенту, це так звана модель-відображення (View-model), при кожному зміні внутрішнього стану, чи при зміні вхідним параметрів компонент знову робить рендер відображення, тобто повертає jsx код, далі він перетворюється на звичайні об’єкти, та завдяки алгоритму звірки react може зрозуміти чи потрібно оновлювати відображення на екрані чи ні.

Завдяки цьому всьому react є найбільш популярним інструментом для побудування користувацьких інтерфейсів.

3.3.2 Фізична модель даних

Оскільки додаток розробляється без бази даних, тому що розрахування проводяться в той же час як користувач заповнює дані та одразу отримує результат. Також додаток не агрегує дані в собі, та в такому вигляді має клієнтську архітектуру. Витікаючи з цього програма не потребує поділ повноважень, і в такому додатку не потрібна система реєстрації чи авторизації.

Для відображення фізичної моделі даних була обрана діаграма компонентів, яка може зобразити головні компоненти системи, та їх зв’язки між собою, в ролі компонентів системи можуть виступати різні сутності: вихідний, бінарний чи код який виконується. Але найчастіше компонентом відповідає файл. Основними елементами діаграми є: компоненти, інтерфейс, залежності між ними та класи.

На рисунку 3.10 зображена діаграма компонентів для додатку «Удосконалення лінійної регресії».

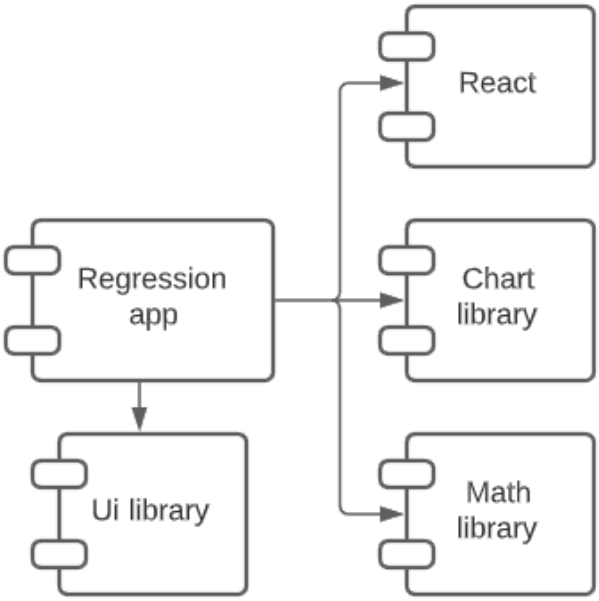


Рисунок 3.10 – Діаграма компонентів

Головним компонентом програми є сам додаток, який використовує інші компоненти та бібліотеки, після цього всі ці компоненти зберуться в один виконавчий файл, який користувач зможе завантажити собі. В додатку використовуються допоміжні засоби: для малювання графіків «ChartJs», для розрахування деяких математичних дії, для збереження часу на проектування дизайну, для елементів інтерфейсу використовується бібліотека «Material UI», і ядром формуванням користувацького інтерфейсу виступає бібліотека «React», всі ці бібліотеки є безкоштовними та є проектами з відкритим вихідним кодом(OpenSource).

Також важливим пунктом є розгортання системи, та розуміння як користувач може отримати доступ до цього, та побачити як в цілому виглядає запуск програми, і які є потреби для цього.

Для таких випадків існує діаграма розгортання, вона відображає компоненти та елементи додатку, які приймають участь лише на етапі

виконання програми. Діаграма містить в собі графічні образи процесів та зв’язок між ними (див. рис. 3.11).

На рисунку 3.11 зображена діаграма розгортання для додатку.

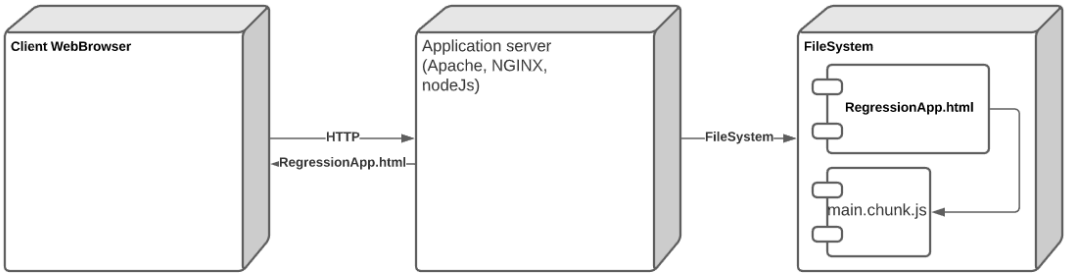


Рисунок 3.11 – Діаграма розгортання

Оскільки розроблена програма є веб-додатком, для неї потрібен веб сервер з виділеним, який зможе віддавати статичні файли, з будь якого місця. Оскільки веб додаток буде цілком збудованим через Webpack, всі файли проекту будуть міститися в одному файлі, тобто буде основний файл програми який буде завантажувати клієнт. Тому завантажив додаток один раз, нам не потрібно більше звертатися до додаткових сервісів, та збільшувати ризики не працюючої програми. Також був реалізована технологія «Progressive web application», завдяки якої користувачу достатньо один раз завантажити додаток у свій браузер, та після цього додаток буде збережений в кешу браузера, та після повторного звернення до веб серверу, браузер зможе без опитування веб-серверу одразу повернути дані, які було до цього збережені.

3.3.3 Тестування варіантів використання

Для додатку доцільне проведення тестування головних варіантів використання. Тестування робилося с варіантами використання «Завантаження даних».

Щоб протестувати варіанти використання застосовувався метод еквівалентної розбивки, в якому потрібно розділити інтерактивну область

програми на чітке число класів еквівалентності, для того щоб мати можливість пропустити кожен тест, що уявляє деякій клас, еквівалентний до будь якого іншого тесту для цього класу. Якщо якийсь тест виявляє помилку, тоді всі інші тести для цього класу еквівалентності повинні знаходити цю саму помилку.

Тестування варіанту «Завантаження даних»

Вхідні дані: текст, файл

Текст – текст формату json

Файл – файл який містить в собі json дані.

При помилці структури або повноті даних, відображається повідомлення про це:

- в поле не введені дані – кнопка зчитати дані блокується, також як і кнопки розрахунків
- файл не обраний – розрахунків заблокована

Тестові вимоги:

- перевірити, що при пустому поле вводу, кнопка читання даних заблокована;
- перевірка того, що при обраному файлі кнопка розрахунків розблокується;
- перевірка того, що якщо дані які були завантаженні з файлу є коректними;
- перевірка того, що дані введені з текстового поля є коректними.

Розробка тестових варіантів для тестових потреб

Перевірити, що при пустому поле вводу, кнопка читання даних заблокована

- Клас еквівалентності: поле вводу має бути пустим

Вхідні дані: «»

Результат: кнопка заблокована

- Клас еквівалентності: введена деяка кількість символів

Вхідні дані: «{“data”: []}»

Результат: кнопка розблокована

Перевірити, що при обраному файлі кнопка розрахунків розблокується

- Клас еквівалентності: файл не обраний

Вхідні дані: «»

Результат: кнопки заблоковані

- Клас еквівалентності: файл обраний

Вхідні дані: «{“data”: [{ “count”: 3, “bytes”: 344, “name”: “someproj” }]}»

Результат: кнопки розблоковані

Перевірити, що якщо дані які були завантаженні з файлу є коректними

- Клас еквівалентності: файл обраний і має json структуру

Вхідні дані: «{“data”: [{ “count”: 3, “bytes”: 344, “name”: “someproj” }]}»

Результат: дані завантажені, користувач був повідомлений про успішність завантаження.

- Клас еквівалентності: файл обраний

Вхідні дані: «{“data: [{ “count”:, “bytes”: 344, “name”: }]}»

Результат: кнопки заблоковані, виведено повідомлення про помилку структури даних.

3.3.4 Випробування програми

Методику випробування написаного програмного продукту міститься в Додатку В. За допомогою опису та інструкцій, що представлені в додатку, для розробленого програмного забезпечення було проведено випробування.

Як тільки користувач завантажує веб-додаток він бачити головну та єдину сторінку додатка, приклад наведений на рисунку 3.12.

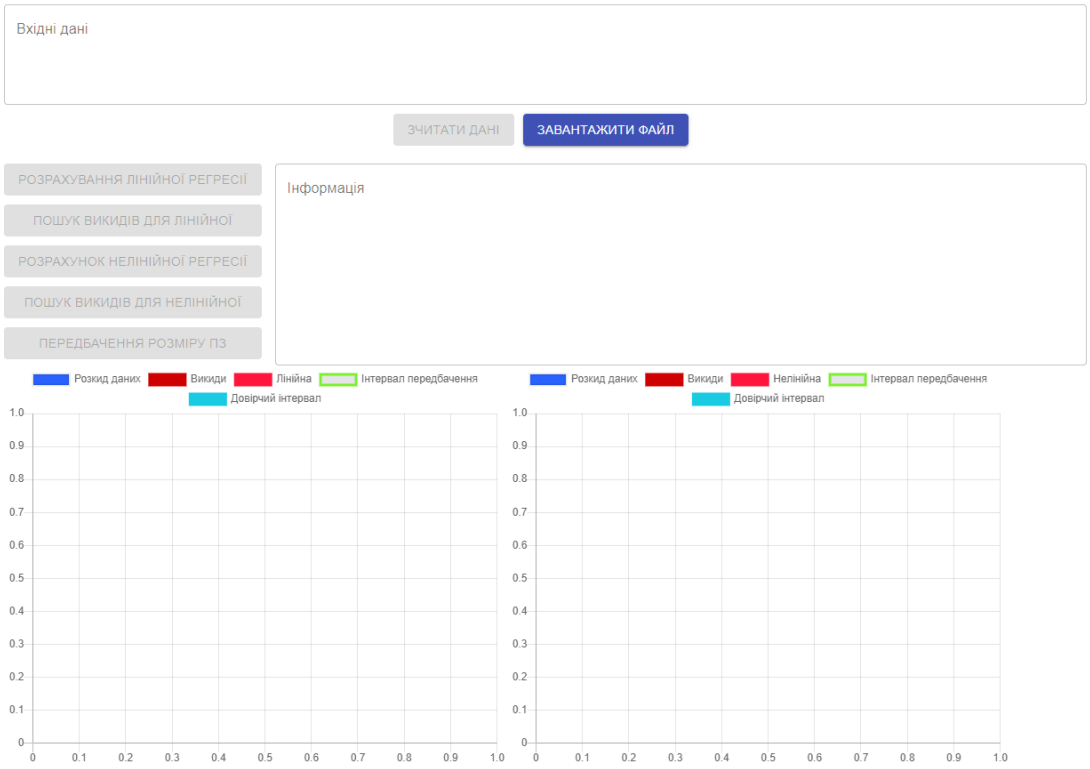


Рисунок 3.12 – Головна сторінка додатку

Далі користувачу потрібно завантажити дані, для цього є два шляхи, перший завантажити файл та другий ввести дані у поле вводу.

Після успішного завантаження даних користувач бачить розкид точок на графіку та відкриваються до цього недоступні можливості програми, це зображено на рисунку 3.13.

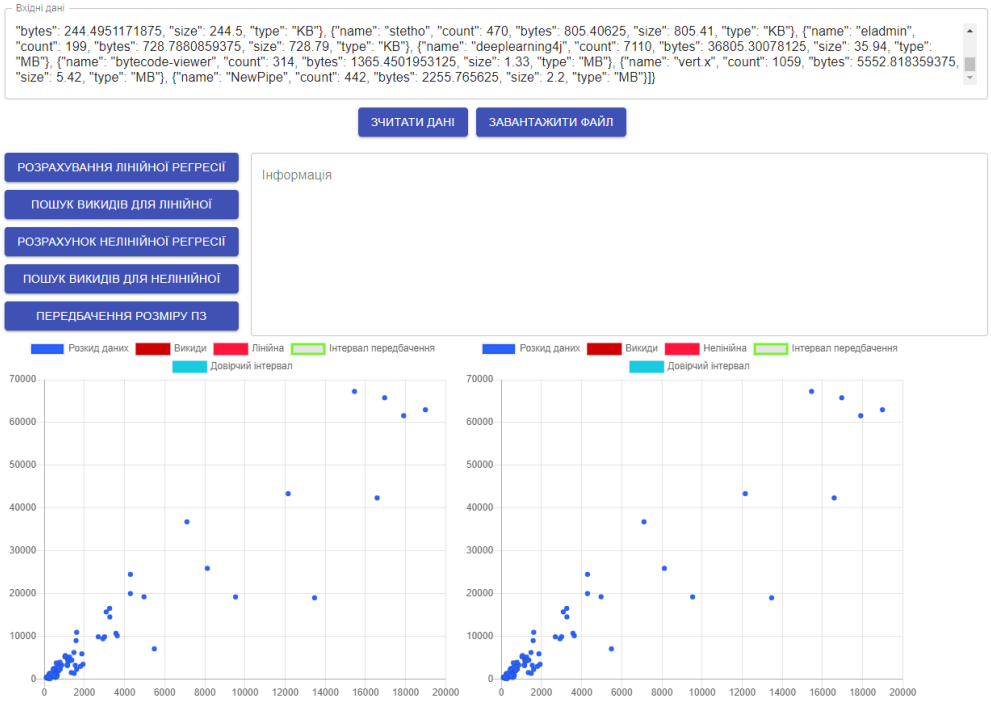


Рисунок 3.13 – Вигляд додатку після завантаження даних

Також якщо користувач введе некоректні дані, в інформаційному полі виведеться інформація, що сталась помилка при зчитані даних, найчастіше ця помилка може трапитися, якщо користувач помилився з полями або неправильно сформована json структура.

Це зображено на рисунку 3.14, у даному випадку ми навмисно зробили помилку, та стерли одну з лапок.

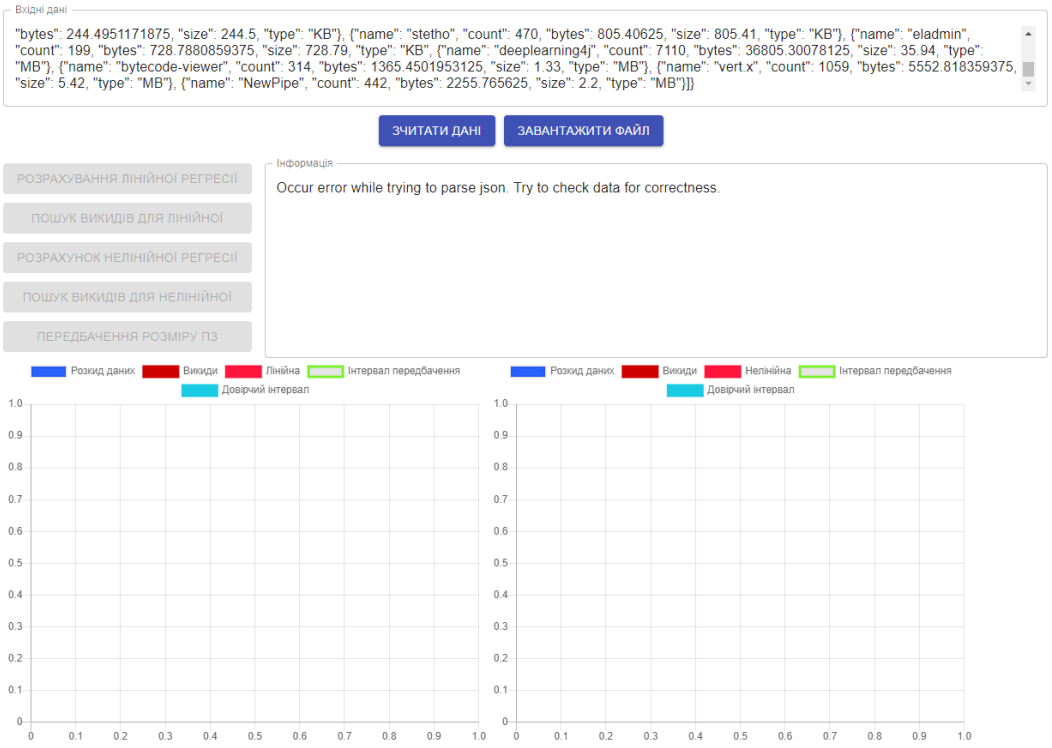


Рисунок 3.14 – Вивід помилки при неправильно наданих даних

Далі ми намагаємося побудувати регресію та знайти викиди для неї, результат роботи програми показаний 3.15.

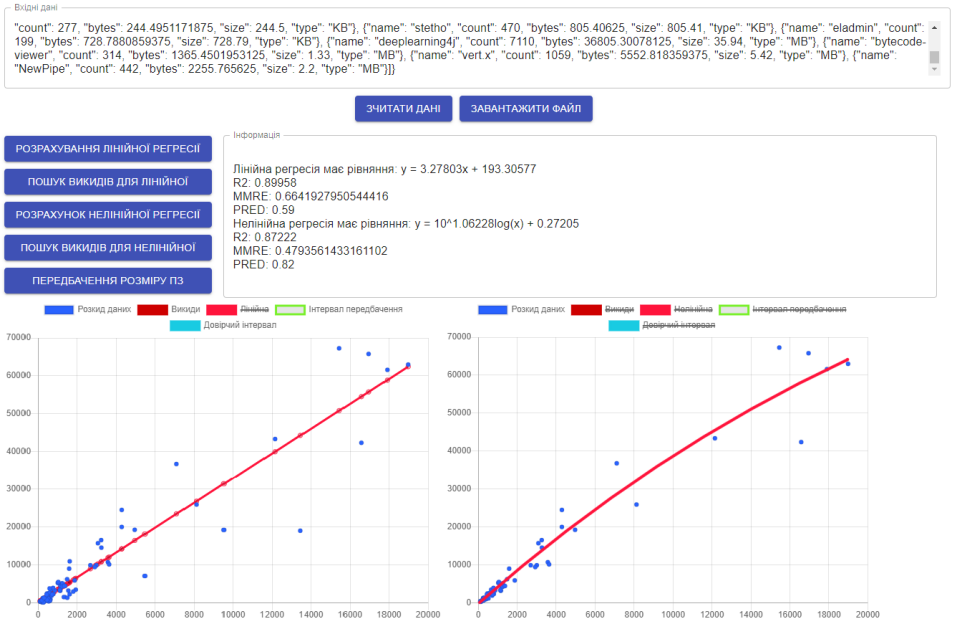
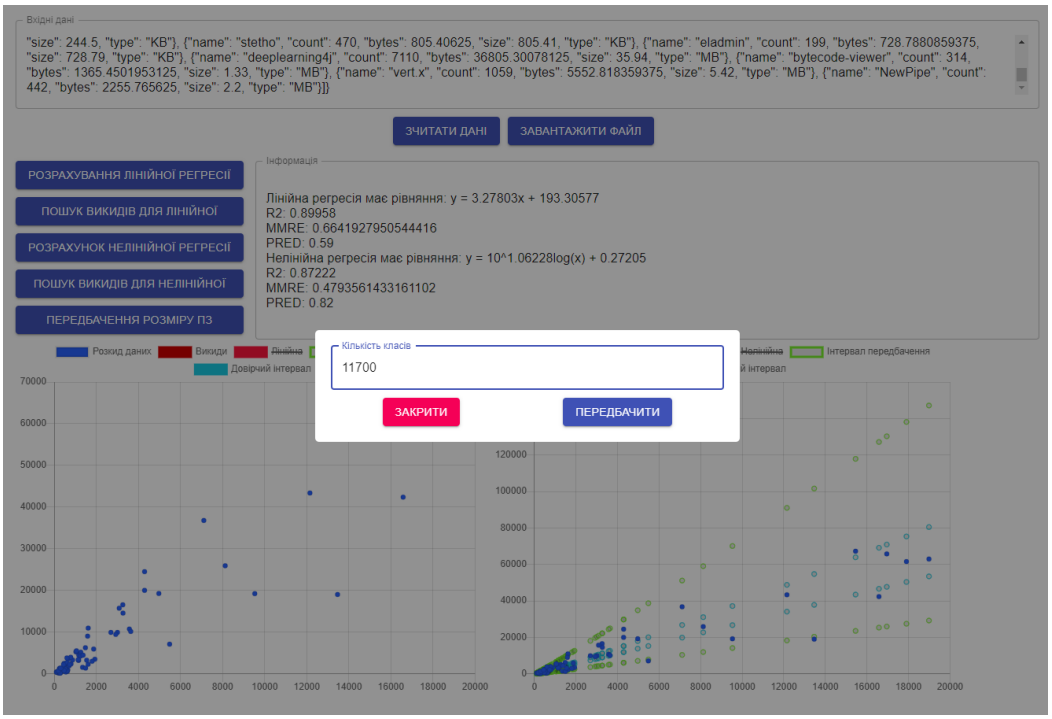


Рисунок 3.15 – Результати роботи програми

Також є можливість передбачити значення довільного значення класів, для цього потрібно натиснути кнопку «Передбачення розміру ПЗ» та ввести кількість класів, після чого в полі «Інформація» з’явиться інформація щодо передбаченому значенню (див. рис. 3.16, 3.17).



3.16 – Введення кількості даних для прогнозування розміру пз

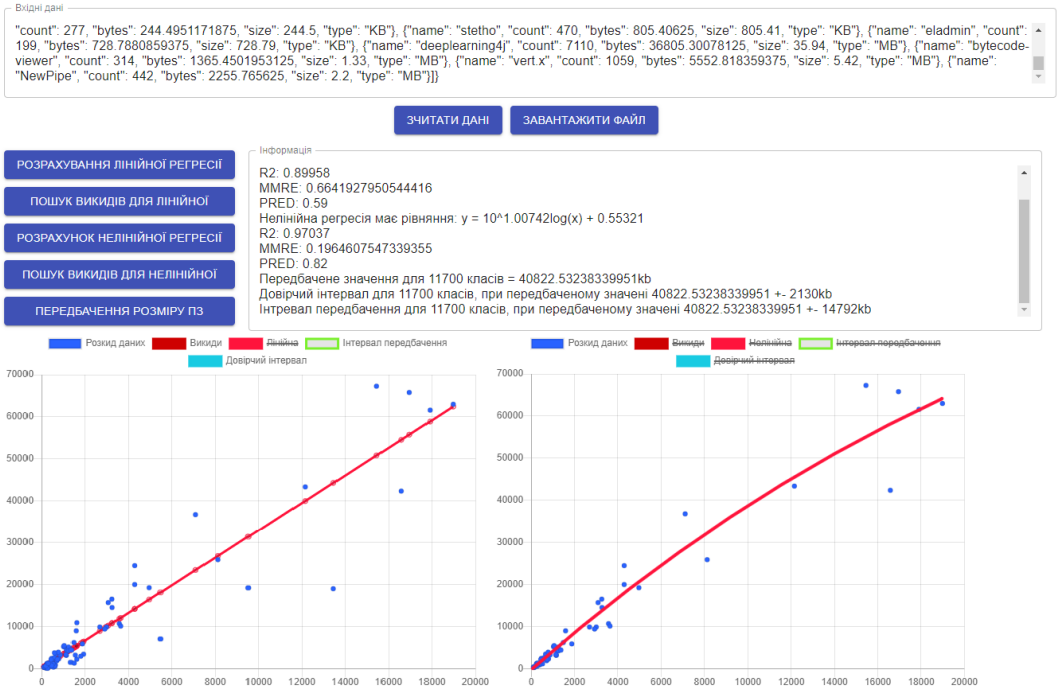


Рисунок 3.17 – Результат передбачення розміру ПЗ на основі нелінійної моделі

Результат роботи програми наводиться у вікні «Інформація», де виводиться вся основна інформація щодо побудування регресійного рівняння (див. рис. 3.18).

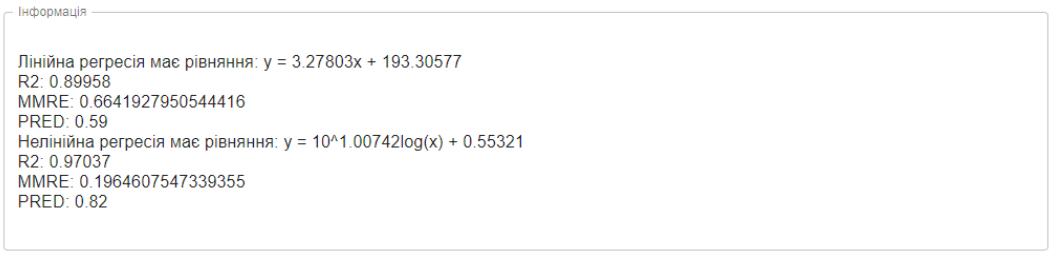


Рисунок 3.18 – Інформація щодо метрик та рівняння регресійної моделі

В результаті був спроектований та реалізований простий інтерфейс, завдяки використанню добре спроектованої архітектурою, що поліпшить подальшу підтримку. Оскільки це веб застосунок, користувач не прив’язаний до якогось конкретної системи чи пристрою, та може завантажити додаток маючі лише інтернет та веб браузер.

3.4 Висновки до розділу 3

Було розроблено проект програмного забезпечення. Проаналізовані основні варіанти використання програми, після приведено деталізацію окремих варіантів. Після визначення головних функцій спроектовано технічний проект, були розроблені ескізний варіант інтерфейсу програми. На основі ескізного і технічних проектів побудований робочий проект програмного забезпечення.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «УДОСКОНАЛЕННЯ РІВНЯННЯ РЕГРЕСІЇ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ НА JAVA»

Розроблений веб-додаток призначений для розрахування та вдосконалення рівняння регресії за рахунок зворотного перетворення та пошук викидів.

Розроблений застосунок має доступ до всіх основних функцій, які потребується для розрахунку моделі регресії, має сучасний та зручний інтерфейс. Основні дані виводяться користувачу на екран у поле виводу та графіку.

Веб-сайт додаток надає наступні можливості користувачу:

- завантаження даних з допомогою файлу;
- завантаження даних через поле вводу;
- розрахунок нормальності розподілу даних;
- розрахунок лінійної регресії;
- розрахунок нелінійної регресії;
- вивід результатів на графік;
- передбачення довільного значення на основі кількості классів;
- пошук викидів.

Технічне оснащення пристрою, на якому запускається веб-додаток є наступне:

Для PC:

- Windows XP і більш пізні;
- Процесор Intel Core i3 3,33, AMD Ryzen 3 (або аналогічний);
- 1 Gb оперативної пам'яті;
- 512 Мб відео пам'яті;

– Firefox 2.x-3.x, Opera 9.5 або більш пізньої версії, Google Chrome.

Для MacOs:

– Mac OS X v10.4 і пізніші;
– Процесор Intel Core i3 з тактовою частотою не менше 1,9 ГГц;
– 512 Мб оперативної пам'яті;
– 512 Мб відео пам'яті;
– Браузери: Firefox 3.x, Firefox 5.x, Safari 3.x.

Для Linux:

– Процесор Процесор Intel Core i3 3,33, AMD Ryzen 3 (або аналогічний);
– 1024 Мб оперативної пам'яті;
– 512 Мб відео пам'яті;
– Браузери: Firefox 2.x-3.x, Opera 9.5, Google Chrome;

Для Android:

– 512 Мб оперативної пам'яті;

Android 5.0 або більш пізньої версії.

– Для iOS:
– 512 Мб оперативної пам'яті;
– iOS 7.0 або більш пізньої версії.

При розробці програмного забезпечення було використано мову JavaScript, та допоміжну бібліотеку React для побудування інтерфейсу. Для побудування графіків використовувалась бібліотека ChartJS.

Була розроблена інструкція користувача, яка наведена в додатку Д. Програма та методика тестування наведена в додатку Г. Код програми представлений в Додатку Б, та опис програми в додатку В.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УДОСКОНАЛЕННЮ ФОРМУЛИ РЕГРЕСІЇ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПО З ВІДКРИТИМ КОДОМ НА JAVA

Дана кваліфікаційна робота присвячена розробці програмного забезпечення для удосконаленню формулі регресії для оцінювання розміру ПО з відкритим вихідним кодом на java, яка може бути використана в цілях аналізу проекту, та оцінити його розмір відносно інших програм.

Доцільність розробки програмного забезпечення можливо обґрунтувати з економічної точки зору. Якщо ми розрахуємо собівартість програмного забезпечення та прибуток. Основним критерієм економічною доцільністю витрат на розробку програмного продукту, буде річний економічний ефект.

Обґрунтувати доцільність розробки з економічної точки зору можливо, розрахувавши собівартість програмного забезпечення та прибуток, що буде отриманий в результаті продажу. Основним показником, що визначає економічну доцільність витрат на створення продукту – є річний економічний ефект.

Для початку звернемося до терміну «ефективний». Ефективний результат означає оптимальний результат з мінімальними витратами: на вході вкладаючи мінімум, на виході виходить максимум можливого. Інформацію, наскільки виправдані вкладення, яка результативність роботи компанії, дає розрахунок ефективності - показник, що характеризує оптимальність використання ресурсів.

Хорошим показником економічної ефективності програмного продукту, що розраховується на стадії створення і впровадження – є ефективність витрат, яку характеризує строк окупності програмного забезпечення.

В цьому розділі розраховується собівартість розробки програмного продукту та виконується оцінка окупності цього проекту, за рахунок скорочення штату працівників.

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 10000 грн. Додаткова заробітна плата складає 10% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 11000 грн./міс , а вартість сучасної ПЗВМ складає 7000 грн. (середня вартість машини на базі AMD Ryzen 3). Вартість кіловат-години електроенергії дорівнює 1.10 грн. Витрати на допоміжні матеріали наведені в табл. 5.1

Таблиця 5.1 - Витрати на допоміжні матеріали

Пункти витрат	Сума, грн.
Програмне забезпечення	500
Папір	120
Непередбачені витрати	200
Разом матеріали і комплектуючі вироби.	250
Разом матеріали і комплектуючі вироби.	250

Вартість розробки системи розраховується за формулою:

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

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 1,1 грн. до 100кВт включно та 1,72 після 100кВт отримаємо:

$$З_е = (100 * 1{,}1 + 68 * 1{,}72) * 0{,}3 = 68{,}08 \text{ грн.}$$

Представимо всі поточні витрати на розробку програмного забезпечення в таблиці 5.2

Таблиця 5.2 - Витрати на розробку програмного забезпечення

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	міс.	2
Основна і додаткова заробітні плати	грн.	10000
Відрахування на соціальні заходи	грн.	4000
Загальногосподарські витрати	грн.	1750
Витрати на допоміжні матеріали	грн.	1320
Витрати на електроенергію	грн.	68,08

Відповідно до формули (5.1) вартість розробки ПЗ складає:

$$С_{пр} = (10000 + 4000 + 1750 + 68{,}08) * 3 + 1320 = 48774{,}24 \text{ грн.}$$

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

$$А_{об} = 7000 * 0{,}6 = 4200 \text{ грн.}$$

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

$$В_м = 10000 * 0{,}05 = 350 \text{ грн.}$$

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:

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

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

$$З_е = 1773,1 * 0,3 * 1,1 = 585,13 \text{ грн.}$$

Експлуатаційні витрати для ПЕОМ за рік складуть:

$$З_{зр} = 4200 + 350 + 585,13 = 5135,13 \text{ грн.}$$

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

$$З_{се} = 48774,24 + 5135,13 = 53909,37 \text{ грн.}$$

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:

$$10000 * 12 * 1 = 120000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

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

де ΔC_n – вивільнені кошти після впровадження системи (120000 грн.) мінус експлуатаційні витрати (5135,13 грн.);

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

k – одноразові витрати на впровадження продукту (53909,37).

$$Э_{рік} = 120000 - 5135,13 \ - 53909,37 * 0,6 = 82519,25 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{48774{,}24}{120000 - 5135{,}13} \approx 0.43 \text{ року}$$

Отже строк окупності програмного забезпечення складає приблизно 5-6 місяців.

Висновки

В розділі були зроблені розрахунки на створення і впровадження програмного забезпечення, після чого була розрахована економічна ефективність та строк окупності програмного продукту. Як висновок з розрахунків, після впровадження програмного забезпечення строк окупності буде складати 5-6 місяців протягом першого року. Тому є доцільним важити, що програмний продукт є економічно обґрунтованим та вигідним.

6 ОХОРОНА ПРАЦІ В ОФІСІ

Під терміном «охорона праці» мають на увазі систему законодавчих, соціально-економічних, технічних, санітарно-гігієнічних і організаційних заходів і засобів, що забезпечують безпеку, збереження здоров’я і працездатності людини в процесі праці.

Науково-технічний прогрес вплинув на серйозні зміни в умови виробничої діяльності працівників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці і відпочинку.

В даний час комп'ютерна техніка широко застосовується у всіх областях діяльності людини. При роботі з комп'ютером людина піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в попереку, в області шиї і руках.

6.1 Аналіз шкідливих та небезпечних факторів у офісі Upland inc

Праця фахівців, весь робочий день проводять в офісі, завжди пов'язаний з великою інтелектуальної, сенсорної (спостереження за монітором) і емоційним навантаженням. Звідси виникає необхідність суворого дотримання режимів роботи. Монотонний, важкий і напружений трудовий процес офісного персоналу полегшується встановленням регламентованих перерв. Їх частота і тривалість визначаються видом трудової діяльності. Так, при зчитуванні інформації з монітора і наборі протягом восьмигодинного робочого дня до 40 тисяч друкованих знаків сумарний час перерв повинно досягати 70 хвилин.

Хоча шкідливих і небезпечних факторів в офісних приміщеннях менше, ніж на виробництві в класичному його розумінні, необхідно пам'ятати, що і тут присутні чи можуть бути присутніми шкідливі фактори (хімічні, біологічні, психофізіологічні і фізичні) відповідно до «Небезпечні і шкідливі виробничі фактори ».

За наслідками впливу на організм шкідливі фактори підрозділяють на:

- загальнотоксичні - призводять до загального отруєння організму;
- сенсибілізуючі - провокують алергічні реакції;
- канцерогенні - провокують розвиток онкологічних захворювань;
- мутагенні - провокують виникнення мутацій, впливають на репродуктивну функцію - призводять до безпліддя.

Часто дані критерії відносять здебільшого до шкідливих хімічних речовин. Однак, наприклад, алергічні реакції може викликати і проста пил (фізичний фактор), і пилок або грибок (біологічний фактор). А мутагенез і канцерогенез можуть викликати випромінювання (фізичний фактор).

Хімічні шкідливі фактори

Наявність хімічних шкідливих і небезпечних факторів в офісах пов'язано, здебільшого, з використанням недорогих синтетичних матеріалів в «чистому» вигляді або у вигляді «добавок», а також чорнила для принтерів, факсів,

копірів. Дешева меблі, пластикові стінові панелі, килимові покриття, робочі елементи обладнання, віконні профілі виділяють складний «коктейль» токсичних речовин. Феноли, бензол, формальдегід - лише деякі з довгого списку.

Хоча синтетичні матеріали дозволені до застосування в офісах, багато що залежить від їх якості. Дешевизна багатьох пластиків обумовлена використанням при виробництві не найдосконалішою технології, а це означає, що процес полімеризації відбувається не повністю. В результаті з готової продукції тривалий час виділяються вільні радикали. Ці «уривки» полімерних ланцюгів мають підвищену хімічну активність, потрапляючи в організм, атакують всі клітини, провокуючи появу нових вільних радикалів, «розмноження» яких відбувається практично в геометричній прогресії.

Відомо, що вік синтетики, в цілому, недовгий. І після приблизно 3-х років вона починає руйнуватися, знову ж таки, виділяючи цілий список шкідливих речовин. Причому прискорення деструкції відбувається при світловому, іонізуюче опромінення, вплив озону, кисню, хімічних реагентів, струмів високої частоти і високої напруги і навіть води.

Варто згадати також озон, який, в основному, виділяється при роботі копіювального обладнання. Озон - потужний антисептик, його застосовують при очищенні води, газів, антисептування, проте в високих концентраціях він отруйний.

Біологічні шкідливі фактори

Джерелами біологічних шкідливих чинників в офісі є кондиціонери та вентиляційні системи. Основна причина їх виникнення - невірний розрахунок кратності повітрообміну і / або несвоєчасна чистка систем і заміна фільтруючих елементів. З потоком повітря по приміщенню розносяться бактерії, віруси, спори грибка, що може спровокувати розвиток у співробітників інфекцій різного роду, включаючи важко піддаються лікуванню мікози.

Крім того, при неграмотно організованому повітря об’ємні і недотриманні правил прибирання приміщень створюються умови для розвитку патогенної мікрофлори і плісняви безпосередньо в приміщенні. Основні «місця базування» - підвіконня самі вікна, що виходять назовні стіни, важкодоступні (загороджені меблями) місця, ковролін. В останньому також можуть накопичуватися і розмножуватися пилові кліщі.

Фізичні шкідливі фактори

Більшість шкідливих фізичних факторів офісних приміщень пов'язані з:

- підвищені рівні шуму, запиленості, ЕМІ і МП, статичної електрики, іонізації повітря;
- підвищені або знижені температура, вологість і рухливість повітря; недолік освітленості, фактори;
- недоліки при розрахунку кратності повітрообміну.

Багато хто з перерахованих вище видів шкідливого впливу також можуть бути причиною надмірної негативної психоемоційного навантаження. Наприклад, шум, невірно організоване освітлення робочого місця, протяги, затхле повітря і т.д.

Психофізичні шкідливі фактори

В цілому ж до психофізичних шкідливих факторів відносять психоемоційні перевантаження, викликані, в тому числі, монотонністю роботи, перенапруженням аналізаторів органів слуху, зору, дотику, повторювані операції, здатні привести до виникнення професійних захворювань.

Інші можливі шкідливі фактори.

Джерела світла, такі як світильники і вікна, які дають віддзеркалення від поверхні екрану, значно погіршують точність знаків і тягнуть за собою перешкоди фізіологічного характеру, які можуть виразитися в значній напрузі, особливо при тривалій роботі. Відображення, включаючи відображення від

вторинних джерел світла, повинне бути зведено до мінімуму. Для захисту від надмірної яскравості вікон можуть бути застосовані штори і екрани.

Залежно від орієнтації вікон рекомендується наступне забарвлення стін і підлоги:

- вікна орієнтовані на південь: стіни зеленувато-блакитного або світло-блакитного кольору; підлога - зелений;
- вікна орієнтовані на північ: стіни світло-оранжевого або оранжево-жовтого кольору; підлога - червонувато-помаранчевий;
- вікна орієнтовані на схід: стіни жовто-зеленого кольору, підлога зелена або червонувато-помаранчевого;
- вікна орієнтовані на захід: стіни жовто-зеленого або голубувато-зеленого кольору; підлога зелена або червонувато-помаранчевий.

У приміщеннях, де знаходиться комп'ютер, необхідно забезпечити наступні величини коефіцієнта віддзеркалення: для стелі: 60 ... 70%, для стін: 40 ... 50%, для підлоги: близько 30%. Для інших поверхонь і робочих меблів: 30 ... 40%.

Існує три види освітлення – природне, штучне і поєднане (природне і штучне разом).

Природне освітлення – освітлення приміщень денним світлом, що потрапляє через світлові прорізи в зовнішніх огороджувальних конструкціях приміщень. Природне освітлення характеризується тим, що змінюється в широких межах залежно від часу дня, пори року, характеру області і ряду інших чинників.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішаним освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, у свою чергу, може бути загальним або комбінованим. Загальне – освітлення, при якому світильники розміщуються у верхній зоні приміщення рівномірно або стосовно до розташування обладнання. Комбіноване – освітлення, при якому до загального додається місцеве освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкта розрізнення 0,3 ... 0,5 мм) величина коефіцієнта природного освітлення (КПО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкта розрізнення 0,5 ... 1,0 мм) КЕО повинен бути не нижче 1,0%. Як джерела штучного освітлення звичайно використовуються люмінесцентні лампи типу ЛБ, або ДРЛ, які попарно об'єднуються в світильники, які повинні розташовуватися рівномірно над робочими поверхнями.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300лк, а комбінована – 750лк; аналогічні вимоги при виконанні робіт середньої точності – 200 і 300лк відповідно.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення.

Обсяг приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути менше 19,5м3 / людина з урахуванням максимального числа одночасно працюючих в зміну. Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в таблицях 6.1, 6.2.

Таблиця 6.1 Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря	22…24°С
	Відносна вологість	40…60%
	Швидкість руху повітря	до 0,1м/с
Теплий	Температура повітря	23…25°С
	Відносна вологість	40…60%
	Швидкість руху повітря	0,1…0,2м/с

Таблиця 6.2 Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери

Характеристика приміщення	Об'ємна витрата подаваного в приміщення свіжого повітря, $м^3$/на одну людину в годину
Обсяг до 20$м^3$ на людину	Не менш 30
20..40$м^3$ на людину	Не менш 20
Більш 40$м^3$ на людину	Природна вентиляція

Було приведено таблиці для правильної організації робочого місця.

6.2 Розрахунок рівня звукового тиску в офісі Upland Inc

Одним з несприятливих факторів виробничого середовища в офісі є високий рівень шуму, створюваний друкованими пристроями, устаткуванням для кондиціонування повітря, вентиляторами систем охолодження в самих ЕОМ.

Для вирішення питань про необхідність і доцільність зниження шуму необхідно знати рівні шуму на робочому місці оператора.

Рівень шуму, що виникає від декількох некогерентних джерел, що працюють одночасно, підраховується на підставі принципу енергетичного підсумовування випромінювань окремих джерел за формолою:

$$L = 10lg \sum_{i=1}^{i=n} 10^{0,1L_i}, \quad (6.1)$$

Де L_i – рівень звукового тиску і-го джерела шуму;

n – кількість джерел шуму.

Отримані результати розрахунку порівнюється з допустимим значенням рівня шуму для даного робочого місця. Якщо результати розрахунку вище допустимого значення рівня шуму, то необхідні спеціальні заходи щодо зниження шуму. До них відносяться: облицювання стін і стелі залу звукопоглинальними матеріалами, зниження шуму в джерелі, правильне планування устаткування і раціональна організація робочого місця оператора.

Рівні звукового тиску джерел шуму, що діють на оператора на його робочому місці представлені в табл. 6.3.

Таблиця 6.3 – показники шуму різних приладів

Джерело шуму	Рівень шуму, дБ
Жорсткий диск	20
Вентилятор	33
Монітор	10
Клавіатура	20
Принтер	40
Сканер	31

Зазвичай робоче місце оператора володіє наступним устаткуванням: накопичувач в системному блоці, вентилятор (и) систем охолодження ПК, монітор, клавіатура, принтер і сканер.

$L = 10\lg(10^2 + 10^{3,3} + 10^1 + 10^2 + 10^4 + 10^{3,1})$ = 41,29 дБ.

Отримане значення не перевищує допустимий рівень шуму для робочого місця оператора, рівний 65 дБ.

6.3 Заходи щодо зменшення впливу шкідливих та небезпечних факторів

Розглянемо небезпечні і шкідливі виробничі фактори, які діють на користувачів ПК і заходи щодо зниження їх впливу.

Невідповідність розташування робочого місця санітарним нормам. Приміщення офісу, їх розміри (площа, обсяг) повинні в першу чергу відповідати кількості працюючих і розміщеному в них комплекту технічних засобів. Для забезпечення нормальних умов праці санітарні норми СН 245-71 встановлюють на одного працюючого обсяг виробничого приміщення не менше 15 м3, площа приміщення, обгороджених стінами або глухими перегородками, не менше 4,5 м3. Робоче місце являє собою сучасний обчислювальний комплекс, що складається з: ПЕОМ, яка комплектується в загальному випадку: системним блоком, монітором, периферійною технікою; робочий стіл; крісло). Рекомендовані розміри столу: висота - 725 мм, ширина - 600 - 1400 мм, глибина -800 - 1000мм. Робочий стілець повинен мати такі основні елементи: сидіння, спинку та стаціонарні або знімні підлокітники, він повинен бути рухомим і зручним. Монітор і клавіатура повинні розташовуватися на оптимальній відстані від очей користувача (але не ближче 60 см) з урахуванням розміру алфавітно-цифрових знаків і символів.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній регульованій висоті робочої поверхні, окремо від столу на відстані 100 ... 300 мм від краю, ближчого до користувача.

Конструкція монітора повинна забезпечувати фронтальне спостереження екрана шляхом повороту корпуса в горизонтальній площині навколо вертикальної осі в межах ± 300 і у вертикальній площині навколо горизонтальної осі в межах ± 300 з фіксацією в заданому положенні.

Підвищена або знижена вологість повітря.

Метрологічні умови в приміщенні визначаються температурою, вологістю і швидкістю руху повітря. Норми метеорологічних умов на виробництві регламентуються ГОСТ 12.1.005-76 "Повітря робочої зони".

Норми температури, відносної вологості та швидкості руху повітря в робочій зоні обчислювальних центрів наведені в таблиці 6.4.

Заходи, що вживаються для підтримки мікрокліматичних умов: установка кондиціонерів, обігрівачів, вентиляції.

Таблиця 6.4 – Норми температури, відносної вологості, швидкості

Період року	Температура, ^{0}C	Відносна вологість, %	Швидкість руху, м/с
Холодні і перехідні періоди (температура зовнішнього повітря нижче + 100С)	19	4060	0,1
Теплий період (температура зовнішнього повітря вище + 100С)	20	4060	0,2

Для нормалізації вологості повітря в приміщення з робочими комп’ютерами слід застосовувати зволожувачі повітря, заправлені щодня дистильованої або кип'яченої водою. Встановити кондиціонери або загально обмінною припливно - витяжну вентиляцію.

Підвищена або знижена іонізація повітря.

Необхідно концентрувати позитивні і негативні іони в повітрі робочої зони. Для цього необхідно встановити: генератори негативних іонів, установки

штучного зволоження, кондиціонери, загальнообмінну припливну - витяжну вентиляцію, захисні екрани, які повинні бути заземлені.

Підвищений рівень шуму на робочому місці.

Вимоги до рівня шуму і вібрації. У приміщенні ВЦ можливі наступні види шумів:

1) аеродинамічний, створюваний роботою кондиціонерів;

2) механічний, створюваний пристроями друку;

3) електромагнітний, створюваний перетворювачами напруги.

Допустимі рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку в дБ на робочих місцях наведені в таблиці 6.5.

Таблиця 6.5 – Допустимі рівні звукового тиску

Рабочие места	Уровни звукового давления со среднегеометрическими частотами, Гц	Уровни звука и эквивалентные уровни, дБ
Помещение программистов	40	45

Для захисту від шуму стіни і стелі приміщень повинні бути обкладені звукопоглинальним матеріалом. Як звуковбирний матеріал необхідно використовувати спеціальні перфоровані плити, панелі, мінеральні плити та інші матеріали аналогічного призначення. Крім цього необхідно використовувати підвісні акустичні стелі.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього середовища – комплекс заходів (організаційних, правових, економічних, природничо-наукових, виробничо-технічних) щодо обмеження негативного впливу господарської та іншої діяльності на навколишнє середовище, під якою розуміється як природне середовище, так і штучні об'єкти, створені людиною для забезпечення його соціальних потреб (будівлі, дороги, інженерні споруди та ін.), а також об'єкти природно-антропогенного характеру (сади, лісосмуги та ін.).

Охорона навколишнього середовища є природоохоронній та соціальній функцій держави і органів місцевого самоврядування.

Охорона навколишнього середовища здійснюється державними і муніципальними органами, громадськими та політичними партіями, учасниками підприємницької діяльності та ін. З метою забезпечення сприятливого навколишнього середовища, раціонального використання і відновлення природних ресурсів. Діяльність з Охорони навколишнього середовища спрямована на підтримку такої якості навколишнього середовища, яке забезпечувало б стійке функціонування природних екологічних систем, природних і природно-антропогенних об'єктів.

Економічні інструменти, використовувані для охорони навколишнього середовища: податкові пільги підприємствам, що застосовують передові природоохоронні технології, економічні та фінансові стимули створення і впровадження сучасних способів і методів очищення викидів і скидів, переробки відходів, зниження рівня шумів, різних шкідливих для людини випромінювань (дивись Забруднення навколишнього середовища); введення економічних санкцій за негативний вплив на навколишнє середовище; встановлення в ряді країн спеціальних екологічних податків; фінансування державних програм з Охорони навколишнього середовища і т. д.

Охорона навколишнього середовища врегульована законодавством. Природоохоронні норми містяться в конституціях, міжнародних договорах, законодавчих і підзаконних актах, які є джерелами екологічного, земельного, водного, лісового, містобудівного, адміністративного права та інших його галузей. Законом встановлюється кримінальна і адміністративна відповідальність за порушення законодавства про охорону навколишнього середовища.

Екологічна ситуація виявилася однією з головних причин погіршення основних показників здоров'я населення, зниження середньої тривалості життя і зростання смертності.

Проблема захисту навколишнього середовища і природних ресурсів також важлива задача, майже немає в світі держави, яке б в тій чи іншій мірі не намагалося її вирішити. Однак для цього необхідна відповідна статистична інформація. Існує безліч концепцій і методів аналізу впливу економічної діяльності на природне середовище і зворотного впливу природного середовища на економічну діяльність, а також оцінки збитку від забруднення навколишнього середовища та ефективності природоохоронних заходів.

В даний час статистикою навколишнього середовища охоплені всі компоненти природного середовища, і в першу чергу такі, як повітря, вода, земля, рослинний і тваринний світ, надра.

Найважливіша задача охорони навколишнього середовища – розкрити причинно-наслідкові зв'язки у взаємодії людського суспільства і природи. Ще більш складна задача - знайти заходи до усунення причин або несприятливих наслідків людської діяльності. Проблеми охорони навколишнього середовища і використання природних ресурсів складаються з комплексу державних, міжнародних і громадських заходів, реалізація яких знаходиться в прямій залежності від соціально-економічного ладу різних держав і їх технічних можливостей. Основною стратегічною лінією наукової та господарської

діяльності людей повинна стати формула: зрозуміти, щоб передбачити, передбачити, щоб раціонально використовувати.

Раціональний підхід природокористування повинен спиратися на фундаментальні принципи: по-перше, можливе повне використання природного ресурсу; по-друге, доведення невикористаних відходів виробництва до такого стану, при якому вони можуть бути асимільовані екологічними системами.

7.1 Забруднення навколишнього середовища промисловими підприємствами

Промислові підприємства приносять користь економіці багатьох країн, а ось екології завдають шкоди. На сьогоднішній день негативний вплив на навколишнє середовище наносять виробництва наступних сфер:

- металургійні;
- нафтохімічні;
- машинобудівні;
- хімічні.

В результаті роботи цих об'єктів в атмосферу виділяється вуглекислий і сірчані гази, зола і отруйні гази. Ці елементи, перш за все, забруднюють атмосферу, а також грунт і воду, впливають на флору і фауну.

Взаємодія промислового виробництва і природи має розглядатися в єдності, як процес природокористування державними інститутами. Він носить соціальний характер, так як відбувається людьми в рамках трудових відносин. Оскільки виробництво є складовою частиною, суспільним інститутом будь-якої держави, то для нього характерні практично всі проблеми суспільства. Взаємна вплив промисловості та навколишнього середовища виступає як би складовим елементом екологічної системи «людина - природа».

Екологічні проблеми є надзвичайно актуальними як для окремого підприємства і всього промислового комплексу країни, так і для Землі в цілому. Розвиток промисловості, з одного боку, - результат науково-технічного прогресу і виробничої діяльності людей. А з іншого, промисловість - основний споживач природних ресурсів і потужне джерело забруднення. Незважаючи на те що екологічна безпека окремо взятих промислових об'єктів безперервно підвищується, в цілому по країні питання захисту навколишнього середовища встають все гостріше, що викликано рядом багатьох об'єктивних і суб'єктивних причин. Кількісне і якісне вдосконалення промислових підприємств як одного з елементів екосистеми «підприємство - природне середовище» незмінно призводить до кількісно-якісної зміни іншого елемента даної екосистеми - природи, а розвиток підприємств переводить ці зміни на якісно новий рівень. Так, збільшення виробничих потужностей на підприємстві і зростання випуску продукції призводять до підвищення кількості споживаних ресурсів - а значить, до збільшення шкідливих викидів у довкілля. Відносини між двома паралельними процесами - процесом розвитку підприємств і промисловості в цілому і процесом погіршення екологічної обстановки відображають діалектичне заперечення, яке показує три основних напрямки вирішення питання захисту навколишнього природного середовища.

Перший напрямок. Повне припинення промислового виробництва.

За це виступає партія Зелених і організація «Greenpeace», які, пропагуючи цнотливість навколишньої природи, забувають, що захист природи і прогрес людства – абсолютно протилежні або обернено пропорційні процеси. Розвиток людської цивілізації неминуче веде до порушення природного середовища, і, навпаки, боротьба за чистоту природи вимагає повернення до виробничі суспільству.

Другий напрямок. Розвиток і функціонування промислових підприємств при ігноруванні стану природного середовища, тобто заперечення екологічних проблем. Однак це неминуче призводить до екологічної кризи.

Ці напрямки - вирішення проблеми шляхом знищення одного з елементів екосистеми «підприємство - природне середовище», а саме - підприємства та промисловості (в першому випадку) і природного середовища (у другому випадку).

Третій напрям – оптимальне поєднання функціонування промислових підприємств з підтриманням максимально можливої їх екологічної безпеки. Скорочення виробництва до розумної достатності і його оптимізація з одночасним захистом навколишнього природного середовища.

Вирішення екологічних проблем вимагає наукового підходу, як би не відрізнялася сучасна екологічна обстановка в світі від ситуації з природою сто - сто п'ятдесят років тому.

Працюючи, промислові підприємства виробляють масу корисних виробів, починаючи від посуду і домашнього начиння, закінчуючи автомобілями, кораблями і літаками. Для виготовлення тих чи інших предметів потрібна велика кількість ресурсів. В ході їх обробці з'являється велика кількість відходів. Їх потрібно добре переглянути, оскільки частина можна пустити на вторинну переробку. Використовуючи раціональний підхід природокористування, можна значно знизити забруднення навколишнього середовища промисловими підприємствами.

Перша 10-ка найбільших забруднювачів навколишнього середовища в Україні по скидах забруднених стічних вод наступна:

1. ПАТ АК "Київводоканал", Київ
2. ЗАТ "Меткомбінат" Азовсталь ", Маріуполь, Донецька обл.
3. ЛМКП "Львівводоканал", Львів
4. ВАТ "Дніпровський Меткомбінат", Кам'янське, Дніпропетровська обл.
5. КП "Дніпроводоканал", Дніпропетровськ
6. ВАТ "Запоріжсталь", Запоріжжя
7. МКП "Миколаївводоканал", Миколаїв

8. Філія "Павлоградське регіональне управління по водопостачанню та очищення каналізаційних стоків" ПАТ "ДТЕК Павлоградвугілля", Павлоград, Дніпропетровська обл.

9. КП "Чернігівводоканал", Чернігів

10. КП "Міськводоканал", Суми

7.2 Розробка заходів щодо зменшення забруднення

До планувальних заходів, які приймаються на етапі проектувальних робіт або розробки документації для нового виробничого напрямку або будівель відносять такі, як:

облік основних напрямків вітру і розташування будівель на основі «рози вітрів» на віддалі від житлових кварталів;

проектування і організація СЗЗ (санітарно-захисної зони);

розташування «заслону» між підприємством та житловими масивами у вигляді «зеленої зони», гірської гряди і так далі.

До технологічних заходів відносяться:

- зменшення кількості екологічно «брудних» процесів на підприємстві;
- використання більш технологічних і екологічних процесів;
- збільшення потужності виробничих агрегатів, а не їх кількості;
- вибір більш екологічно чистого палива і сировини;
- застосування рециркуляції та очищення димових газів і викидів.

При бажанні і високого ступеня відповідальності з боку керівництва підприємства, заходи щодо зменшення викидів в атмосферу обов'язково принесуть запланований ефект, що позитивно позначиться як на економіці і репутації підприємства, так і на стан навколишнього середовища.

Виявленні, оцінці, постійному контролі й обмеження шкідливих викидів в навколишнє середовище, створення природоохоронних та ресурсозберігаючих технологій і техніки.

Розробці юридичних законів, правових актів з охорони навколишнього природного середовища, а також матеріальне стимулювання виконання вимог цих законів і природоохоронних заходів.

Безвідходна технологія є найбільш активною формою захисту навколишнього середовища від шкідливого впливу викидів промислових підприємств. Під поняттям «безвідходна технологія» слід розуміти комплекс заходів в технологічних процесах від обробки сировини до використання готової продукції, в результаті чого скорочується до мінімуму кількість шкідливих викидів і зменшується вплив відходів на навколишнє середовище до прийнятного рівня. У цей комплекс заходів входять:

- створення і впровадження нових процесів отримання продукції з утворенням найменшої кількості відходів;
- розробка різних типів безстічних технологічних систем і водооборотних циклів на базі способів очищення стічних вод;
- розробка систем переробки відходів виробництва у вторинні матеріальні ресурси;
- створення територіально-промислових комплексів, що мають замкнуту структуру матеріальних потоків сировини та відходів всередині комплексу.

В даний час досягнуті успіхи в області створення і впровадження безвідходної технології в ряді галузей промисловості, однак повний переклад народного господарства на безвідходну технологію зажадає рішення великого комплексу вельми складних технологічних, конструкторських та організаційних завдань, заснованих на використанні новітніх науково-технічних досягнень. Тому до всебічного впровадження безвідходної технології важливими напрямками екологізації промислового виробництва слід вважати:

- вдосконалення технологічних процесів і розробку нового обладнання з меншим рівнем викидів домішок і відходів у навколишнє середовище;
- заміна токсичних відходів на нетоксичні;
- заміна неутилізованих відходів на утилізованих;
- застосування пасивних методів захисту навколишнього середовища.

Пасивні методи захисту навколишнього середовища включають комплекс заходів щодо обмеження викидів промислового виробництва з подальшою утилізацією або похованням відходів. До їх числа відносяться:

- очистка стічних вод від домішок;
- очищення газових викидів від шкідливих домішок;
- розсіювання шкідливих викидів в атмосфері;
- глушіння шуму на шляхах його поширення;
- заходи по зниженню рівнів інфразвуку, ультразвуку та вібрацій на шляхах їх розповсюдження;
- екранування джерел енергетичного забруднення навколишнього середовища;
- поховання токсичних і радіоактивних відходів.

Зелені насадження

Для поліпшення охорони зелених зон і лісопаркових територій необхідно визначити їх чіткі межі. Повинні бути встановлені та впорядковані в них місця тривалого і короткочасного відпочинку населення. Організовано охорону і своєчасне очищення даних територій. Значну роль відіграє проведення робіт з розширення в горад і приміських зонах площі зелених насаджень, створення нових парків, садів, скверів. Також строго обмежувати відведення земельних ділянок в лісах зелених зон міст, лісових захисних смугах та інших лісах першої групи, для цілей, не пов'язаних з розвитком лісового господарства. Треба сказати, що в даний час в цій області дуже багато порушень, що пов'язано з погано розвиненою законодавчою системою.

Корисні копалини

З метою зменшення втрат корисних копалин при їх видобутку та переробці, а також попередження забруднення навколишнього середовища відходами виробництва. Впровадження більш ефективних способів і систем розробки родовищ корисних копалин і технологічних схем переробки мінеральної сировини, що забезпечують найбільш повне, комплексне та економічно доцільне вилучення з надр запасів основних і спільно з ними залягаючих корисних копалин, а також використання містяться в них компонентів, що мають промислове значення.

Будівництво або реконструкція цехів, фабрик, установок з комплексної переробки сировини, відвалів і шлаків, виділення капітальних вкладень на ці цілі, визначення термінів завершення переходу вказаних підприємств на роботу, що забезпечує комплексне використання корисних копалин.

Треба сказати, що крім цього вживаються такі заходи, як:

- забезпечення організації виробництва нового, більш досконалого обладнання та апаратури для очищення промислових викидів в атмосферу від шкідливих газів, пилу, сажі та інших речовин;
- проведення відповідних наукових досліджень і дослідно-конструкторської робіт зі створення більш досконалої апаратури і обладнання для захисту атмосферного повітря від забруднення промисловими викидами;
- здійснення на підприємствах, в організаціях та установах шефмонтажу і налагодження газоочисного і пиловловлюючого обладнання та апаратури;
- здійснення державного контролю за роботою газоочисних і пиловловлюючих установок на промислових підприємствах.

Охорона земель

Землекористувачі зобов'язані проводити ефективні заходи по підвищенню родючості грунтів, здійснювати комплекс організаційно-господарських,

агротехнічних, лісомеліоративних і гідротехнічних заходів щодо запобігання вітрової та водної ерозії грунтів, не допускати засолення, заболочування, забруднення земель, заростання їх бур'янами, а також інших процесів, що погіршують стан грунтів.

Промислові та будівельні підприємства, організації, установи зобов'язані не допускати забруднення сільськогосподарських та інших земель виробничими та іншими відходами, а також стічними водами.

Найбільш важлива проблема з усіх розглянутих раніше - проблема охорон вод. Однією з головних завдань є регулювання водних відносин з метою забезпечення раціонального використання вод для потреб населення і народного господарства. Крім того існують і інші завдання:

- охорона вод від забруднення, засмічення і виснаження;
- попередження і ліквідації шкідливої дії вод;
- поліпшення стану водних об'єктів;
- охорона прав підприємств, організацій, установ та громадян, зміцнення законності в галузі водних відносин.
- Розміщення, проектування, будівництво і введення в експлуатацію підприємств, споруд та інших об'єктів, що впливають на стан вод.
- Забороняється введення в експлуатацію:
- нових і реконструйованих підприємств, цехів і агрегатів, комунальних та інших об'єктів, не забезпечених пристроями, що запобігають забрудненню і засміченню вод або їх шкідливий вплив;
- зрошувальних і обводнювальних систем, водосховищ і каналів до проведення передбачених проектами заходів, що запобігають затопленню, підтопленню, заболочування, засолення земель і ерозії грунтів;
- осушувальних систем до готовності водоприймачів та інших споруд відповідно до затверджених проектів;

- водозабірних споруд без рибозахисних пристроїв відповідно до затверджених проектів;
- гідротехнічних споруд до готовності пристроїв для пропуску паводкових вод і риби відповідно до затверджених проектів;
- бурових свердловин на воду без обладнання їх водорегулюючими пристроями і встановлення у відповідних випадках зон санітарної охорони;
- забороняється наповнення водосховищ до здійснення передбачених проектами заходів щодо підготовки ложа.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є удосконалене регресійне рівняння та програма розрахунку цього рівняння. В процесі роботи були вирішені основні завдання. Були досягнути наступні задачі:

- проаналізовані основні речі при побудові регресійної моделі;
- проведений аналіз різних регресійних рівнянь;
- були проаналізовані та порівнянні різни методи пошуку викидів;
- обґрунтований вибір критерія для оцінки розміру програмного забезпечення на мові java;
- зроблено вдосконалення рівняння регресії за допомогою зворотного перетворення та пошуку викидів;
- наведено високорівневу модель програмного забезпечення за допомогою ескізного та технічного проекту програмного забезпечення;
- розроблено програмний додаток;
- розглянуті питання з охорони праці та охорони навколишнього середовища;
- проведено тестування програмного продукту.

Основним досягненням було вдосконалення рівняння регресії, в результаті вдалось підвищити основні критерії оцінки регресійної моделі: коефіцієнт детермінації моделі на 8% , показник прогнозування на 23% та середня величина відносної похибки зменшилась на 0,46 – відносно лінійної моделі.

На основі вдосконалення рівняння регресії було розроблено програмне забезпечення у вигляді веб-застосунку, яке зможе автоматизувати даний процес. Програмне забезпечення задовольняє основним вимогам, які були сформовані в технічному завдані.

СПИСОК ВИКОРАСТАННИХ ДЖЕРЕЛ

[1] Reduce your app size [Електронний ресурс]. Режим доступу: https://developer.android.com/topic/performance/reduce-apk-size

[2] Вартість виправлення помилки на різних етапах розробки ПЗ [Електронний ресурс]. Режим доступу: http://okiseleva.blogspot.com/2018/03/blog-post_59.html

[3] Software Project Cost Estimates Using COCOMO II Model [Електронний ресурс]. Режим доступу: https://www.codeproject.com/Articles/9266/Software-Project-Cost-Estimates-Using-COCOMO-II-Mo

[4] Regression Analysis [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/topics/medicine-and-dentistry/regression-analysis

[5] Основи регресійного аналізу [Електронний ресурс]. Режим доступу: https://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-statistics-toolbox/regression-analysis-basics

[6] Regression: An Explanation of Regression Metrics And What Can Go Wrong [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914#:~:text=Mean%20Squared%20Error%3A%20MSE%20or,preferred%20metrics%20for%20regression%20tasks.&text=Root%20Mean%20Squared%20Error%3A%20RMSE,value%20predicted%20by%20the%20model.

[7] Ершов, Э.Б. Поширення коефіцієнта детермінації на загальний випадок лінійної регресії, що оцінюється за допомогою різних версій методу найменших квадратів // ЦЕМІ РАН Економіка і математичні методи. — ЦЭМИ РАН, 2002. — Т. 38, вып. 3. — С. 107-120.

[8] Inference: Confidence Intervals [Електронний ресурс]. Режим доступу: https://sites.nicholas.duke.edu/statsreview/ci/

[9] What is P-Value? – Understanding the meaning, math and methods [Електронний ресурс]. Режим доступу: https://www.machinelearningplus.com/statistics/what-is-p-value/

[10] Identifying Outliers [Електронний ресурс]. Режим доступу: https://online.stat.psu.edu/stat462/node/172/

[11] Гаусовий розподіл [Електронний ресурс]. Режим доступу: http://nuclphys.sinp.msu.ru/enc/e040.htm

[12] Формула лінійної регресії [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F

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

[14] Logarithmic Transformation in Linear Regression Models: Why & When [Електронний ресурс]. Режим доступу: https://dev.to/rokaandy/logarithmic-transformation-in-linear-regression-models-why-when-3a7c#:~:text=A%20regression%20model%20will%20have,change%20to%20a%20percent%20change.&text=A%20logarithm%20is%20the%20base%20of%20a%20positive%20number.

[15] Polynomial regression [Електронний ресурс]. Режим доступу: https://ru.qaz.wiki/wiki/Polynomial_regression

[16] Вариации регрессии [Електронний ресурс]. Режим доступу: https://neerc.ifmo.ru/wiki/index.php?title=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D1%86%D0%B8%D0%B8_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%B8

[17] Artificial Neural Networks: Some Misconceptions [Електронний ресурс]. Режим доступу: https://dzone.com/articles/artificial-neural-networks-some-misconceptions-par-2

[18] Знаходження викидів і впливових спостережень в регресійному аналізі [Електронний ресурс]. Режим доступу: http://forum.disser.ru/index.php?act=attach&id=284&type=post

[19] Стандартизовані залишки та їх використання [Електронний ресурс] Режим доступу: https://online.stat.psu.edu/stat462/node/172/

[20] 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/

[21] Studentized Residuals [Електронний ресурс] Режим доступу: https://online.stat.psu.edu/stat462/node/247/

[22] Mahalanobis distance [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Mahalanobis_distance

[23] Cook's Distance [Електронний ресурс]. Режим доступу: https://www.mathworks.com/help/stats/cooks-distance.html

[24] Influence Diagnostics [Електронний ресурс]. Режим доступу: https://www.sfu.ca/sasdoc/sashtml/stat/chap55/sect38.htm

[25] Тест Голдфелда – Куандта [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Goldfeld%E2%80%93Quandt_test

[26] Тест Бройша – Пагана [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Breusch%E2%80%93Pagan_test

[27] Improve Linear Regression Using Statistics [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/statistics-supporting-linear-models-bfc24fb9781f

[28] Java Architecture & Components Explained [Електронний ресурс]. Режим доступу: https://www.upgrad.com/blog/java-architecture-components-explained-2020/#:~:text=Final%20Thoughts-,Java%20Architecture%20Explained,which%20the%20machine%20executes%20directly.

[29] Top 100 Stars in Java [Електронний ресурс]. Режим доступу: https://github.com/EvanLi/Github-Ranking/blob/master/Top100/Java.md

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Програмне забезпечення називається «Удосконалення рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на Java».

Створена система має передбачати простий доступ до неї через мережу інтернет та веб-інтерфейс, через персональний комп’ютер або телефон чи планшет, система запитує у користувача вхідні дані для моделі після вираховує регресійну модель та виводь інформацію на екран, в графічному представлені та текстовому.

1. Підстави для розробки

Підставами для розробки є те що в даний час не існує аналогів для оцінювання розміру програмного коду на java, з використанням кількість класів як критерії на основі якої можливо передбачити це.

2. Призначення розробки

2.1 Функціональне призначення

Програмний продукт призначений для побудова моделі, яка зможе передбачувати приблизний розмір програмного забезпечення в кілобайтах на основі передбачуваної кількості класів програми.

2.2. Експлуатаційне призначення

Програмний продукт може використовуватися будь яка людина, котра бажає знайти рівняння регресії для своїх даних, або на основі існуючих вирахувати приблизний розмір програмного продукту на основі кількості класів.

3. Вимоги до програмного продукту

3.1 Вимоги до складу виконуваних функцій

Програмний продукт повинен мати змогу забезпечувати виконувати наступні функції:

- на основі вхідних даних малювати розкид даних на графіку;
- проводити аналіз вхідних даних на нормальність розподілу;
- робити перетворення для вхідних даних до приведення їх до нормальності;
- на основі вхідних наборів даних будувати лінійну та нелінійну регресію;
- зображувати графіки моделей полотні;
- знаходити основні метрики оцінки моделі;
- робити розрахунки по пошуку викидів для моделі;
- передбачення розміру пз на основі кількості класів та рахувати інтервалу передбачення та довірчого інтервалу.

3.2 Вимоги до організації вхідних та вихідних даних

Завантаження даних може відбуватися як за допомогою полю вводу даних json так і з допомогою завантажити файл у форматі json. Після цього програмний продукт виконує розрахунки та приводить результати підрахунків на екран у вигляді графіку, та додатку текстову інформацію о моделі з зазначенням її степені детермінації, коефіцієнтів незалежної змінної, знайдених викидів, та спосіб перетворення вхідних даних.

3.3 Вимоги до надійності

3.3.1 Вимоги до забезпечення надійного функціонування програми

Для сталого (стійкого) функціонування програмного забезпечення, має бути забезпечене виконання організаційно-технічних заходів, які переведені нижче:

- забезпечення постійного місця на накопичувачу, та доступ до нього веб серверу;
- безперебійного живлення фізичного серверу який зберігає дані;
- постійне інтернет сполучення;
- копію даних на іншому фізичному накопичувачу.

3.3.2 Час відновлення після відмови

У разі виникнення проблем з напучувачем який зберігав програмний продукт, відновлення даних, та формування нової копії не повинно перевищувати 1 годину.

У разі виникнення перебою у інтернет сполученні, продовження роботи не повинно перевищувати 3 години.

У ситуації яка виникає в разі несправності технічних засобів, помилці системі серверу, час відновлення не повинен бути більш ніж, повне переналаштування серверу або програмних засобів та заміна потрібних технічних компонентів.

3.3.3 Відмови через некоректність дій користувача

При некоректних діях користувача, такі як завантаженням користувачем неправильних за форматом та складом даних до інтерфейсу програми, можливі відмови програмного забезпечення.

Для цих випадків програма має надати користувачу відповідь у вигляді помилки, та надати можливість відредагувати дані, та завантажити їх спочатку.

3.4 Умови експлуатації

3.4.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких програмний продукт повинен надавити задані функції, повинні задовольняти умовам, які представленні у частині умов експлуатації до технічних засобів.

3.4.2 Вимоги до видів обслуговування

Програмний продукт не потребує додаткове або подальше обслуговування теперішніх функції.

3.4.3 Вимоги до чисельності та кваліфікації персоналу

Необхідна мінімальна кількість персоналу, що є необхідної для роботи програмного продукту складає 1 штатну одиницю – це є користувач програмного продукту.

Користувач повинен мати навички користуванням інтернет браузером та знанням формату даних json.

3.5 Вимоги до складу та параметрів технічних засобів

До складу технічних засобів має входити як найменш один сервер, або персональний комп’ютер який задовольняє наступним мінімальним систем умовам:

- 1gb RAM;
- 20mb вільного простору на SSD/HDD;
- процесор з тактовою частотою не меншою за 1 GHz;

3.6 Вимоги до інформаційної і програмної сумісності

3.6.1 Вимоги до інформаційних структур і методів розв’язку

Програмне забезпечення потребую вхідні дані у форматі JSON які мають наступну структуру:

{"data": [{"name": "elasticsearch", "count": 20129, "bytes": 111655272, "size": 106.48, "type": "MB"}]}.

3.6.2 Вимоги до вихідних кодів та мов програмування

Програмне забезпечення необхідно реалізувати з використанням наступних технологій:

- javascript, typescript – клієнтський код;
- html5, css3 – розмітка сторінок;
- react – користувацький інтерфейс.

3.6.3 Вимоги до програмних засобів, що використовуються програмою

Сервером використовуються наступні засоби:

- ОС Linux Ubuntu 18.04;
- сервер nginx/nodejs.

Для використання продукту необхідний інтернет браузери, наприклад: Google Chrome 50.1+, Safari 8, Edge 2.0, Firefox 30+ та інші з підтримкою веб стандартів на рівні ES5.

3.6.4 Вимоги до захисту інформації та програм

Вимог щодо захисту інформації немає.

3.7 Вимоги до маркування

Програмний продукт розповсюджується як SaaS, доступ до додатку відбувається через мережу інтернет.

3.8 Вимоги до транспортування та збереження

Вимоги до транспортування та зберігання не пред’являються.

4. Вимоги до програмної документації

Склад програмною документації включає в себе:

- технічне завдання;
- текст програмних модулів;
- опис програми;
- інструкція користувача;
- програма та методика випробування.

5. Техніко-економічні показники

Орієнтована економічна ефективність не розраховується.

6. Стадії та етапи розробки

Стадії та етапи розробки наведено в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

№	Стадії розробки	Етапи розробки	Зміст робіт по етапах	Строки виконання
1	2	3	4	5
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: 05.03.20 Закінчення: 26.03.20
2	Ескізний проект	Розробка та затвердження ескізного проекту	Виявлення вимог, концепція майбутньої системи, складання ескізної частини програмного продукту	Початок: 10.04.20 Закінчення: 19.04.20

Продовження табл. А.1

1	2	3	4	5
3	Технічний проект	Розробка та затвердження технічного проекту	Загальні системні вирішення, алгоритми задач, оцінка економічної ефективності автоматизованої системи та перелік заходів підготовки об’єкта для провадження	Початок: 20.04.20 Закінчення: 30.04.20
4	Робочий проект	Розробка та затвердження робочого проекту	Деталізовані загальносистемні проекті рішення, програми та інструкції для розв’язку завдань, розробка програмного забезпечення.	Початок: 01.05.20 Закінчення: 03.06.20

7. Порядок контролю та приймання

Порядок прийому та контроль програми мають бути проводитися відповідно до узгоджених заздалегідь з замовником опис програми, а також методика тестування.

Кожна стадія розробки повинна бути представлена в зазначені терміни і узгоджена з замовником.

Хід проведення приймально-здавальних випробувань проводиться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань.

На підставі протоколу проведення випробувань виконавець спільно з замовником підписують акт приймання-здачі програмного забезпечення в експлуатацію.

У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписується представниками

замовника і розробника і затверджується керівниками організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 3 тижнів виправити зазначені помилки, і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

Лістинг 1 – Файл Controls.tsx

```
import { Box, Button } from '@material-ui/core';
import React, { FC } from 'react';
import styles from './Controls.module.scss';

type Props = {
  fixDistribution: boolean;
  onTransformData: () => void;
  onCalcLinear: () => void;
  onFindOutliers: () => void;
};

export const Controls: FC<Props> = ({
  fixDistribution,
  onTransformData,
  onCalcLinear,
  onFindOutliers,
}) => {
  return (
    <Box className={styles.Controls}>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={!fixDistribution}
        color="primary"
        onClick={onCalcLinear}
      >
        Розрахувати розподіл даних
      </Button>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={!fixDistribution}
        color="primary"
        onClick={onTransformData}
      >
        Перетворення даних
      </Button>
    </Box>
  );
};
```

Лістинг 2 – Файл RegressionControls.tsx

```
import React, { FC, useState } from 'react';
import { Box, Button, MenuItem, Select } from '@material-ui/core';
import styles from './RegressionControls.module.scss';
import cn from 'classnames';

export const RegressionControls: FC<{
  className?: string;
  disabled: boolean;
  calcLinear: () => void;
  findOutliersLinear: () => void;
```

```
  findOutliersPower: () => void;
  calcPower: () => void;
}> = ({
  className,
  disabled,
  calcLinear,
  findOutliersLinear,
  calcPower,
  findOutliersPower,
}) => {
  return (
    <Box className={cn(styles.RegressionControls, className)}>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={disabled}
        color="primary"
        onClick={calcLinear}
      >
        Розрахування лінійної регресії
      </Button>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={disabled}
        color="primary"
        onClick={findOutliersLinear}
      >
        Пошук викидів для лінійної
      </Button>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={disabled}
        color="primary"
        onClick={calcPower}
      >
        Розрахунок нелінійної регресії
      </Button>
      <Button
        className={styles.Button}
        variant="contained"
        disabled={disabled}
        color="primary"
        onClick={findOutliersPower}
      >
        Пошук викидів для нелінійної
      </Button>
      <Select
        labelId="demo-simple-select-label"
        id="demo-simple-select"
        value={1}
        onChange={() => {}}
      >
        <MenuItem value={1}>Графік</MenuItem>
        <MenuItem value={2}>Таблиця</MenuItem>
      </Select>
    </Box>
  );
};
```

Лістинг 3 – Файл InputRegresssionData.tsx

```
import { Box, Button, TextField } from '@material-ui/core';
import React, {
  FC,
  useCallback,
  useState,
  SyntheticEvent,
  ChangeEvent,
} from 'react';
import styles from './InputRegressionData.module.scss';
import cn from 'classnames';

console.log('styles', styles);

type Props = {
  onLoadData: (value: string) => void;
  className?: string;
};

export const InputRegressionData: FC<Props> = ({
  onLoadData: onCalculate,
  className,
}) => {
  const [value, setValue] = useState('');

  const onChange = useCallback(
    (e: ChangeEvent<HTMLInputElement>) => {
      console.log('update value');
      setValue(e.currentTarget.value);
    },
    [setValue]
  );

  const onSubmit = useCallback(() => {
    onCalculate(value);
  }, [onCalculate, value]);

  return (
    <Box className={cn(styles.InputRegressionData, className)}>
      <TextField
        className={styles.textarea}
        fullWidth
        label="Вхідні дані"
        multiline
        rows={4}
        placeholder={'Enter input data'}
        variant="outlined"
        onChange={onChange}
      />
      <Box>
        <Button
          onClick={onSubmit}
          variant="contained"
          disabled={!value}
          color="primary"
          className={styles.Button}
        >
          Зчитати дані
        </Button>
        <Button
```

```
        onClick={onSubmit}
        variant="contained"
        disabled={false}
        color="primary"
        className={styles.Button}
       >
        Завантажити файл
       </Button>
     </Box>
   </Box>
  );
};
```

Лістинг 4 – Файл OutputData.tsx

```
import { Box, TextField } from '@material-ui/core';
import React, { FC, useCallback, KeyboardEvent } from 'react';

type Props = {
  value: string;
  className?: string;
};

export const OutputData: FC<Props> = ({ value, className }) => {
  const preventInput = useCallback((e: KeyboardEvent) => {
    e.preventDefault();
  }, []);

  return (
    <Box className={className}>
      <TextField
        fullWidth
        label="Інформація"
        aria-readonly
        multiline
        rows={10}
        variant="outlined"
        onKeyPress={preventInput}
        value={value}
      />
    </Box>
  );
};
```

Лістинг 5 – Файл RegressionApp.tsx

```
import { Box } from '@material-ui/core';
import React, {
  ComponentProps,
  FC,
  useCallback,
  useEffect,
  useState,
} from 'react';
import { InputRegressionData } from '../DataProvider/InputRegressionData';
import { OutputData } from '../OutputData/OutputData';
import styles from './RegressionApp.module.scss';
import {
```

```
  linear,
  polynomial,
  exponential,
  logarithmic,
  power,
} from 'regression';
import { DataSet, IPureDataItem, LineFunc } from '../../commonTypes';
import {
  calcDeletedResidual,
  createLineDataset,
  createLineFunc,
  createPolynomialDataset,
  deletedFunc,
  dispersionS2,
  gaussinDist,
  getMedian,
  hFunc,
  normalizeData,
  pValue,
  seError,
  thFunc,
  transformDataToArrayView,
  boxCox,
  OutliersItem,
  findOutliersDeletedResidual,
  createLineDataset2,
  findMD,
  findOutliersCookDistance,
  findOutliersDFFIT,
} from '../../Services/helpers';
import { RegressionChart } from '../RegressionChart/RegressionChart';
import { mean, sum, variance } from 'mathjs';
import { LinearRegressionModel } from '../../Services/LinearRegressionModel';
import { RegressionChartAdapter } from '../RegressionChart/RegressionChartAdapter';
import { Controls } from '../Controls/Controls';
import { RegressionControls } from '../Controls/RegressionControls';
import { PowerRegressionModel } from '../../Services/PowerRegressionModel';
import {
  IRegressionModel,
  IRegressionModelConstructor,
} from '../../Services/interfaces';
const { jStat } = require('jstat');

const transformForChart = (
  data: Array<IPureDataItem>,
  vanilaCountsMap: Map<string, number>
): Array<DataSet> =>
  data.map(item => ({
    x: item.count,
    y: item.bytes,
    label: item.name,
    classCount: vanilaCountsMap.get(item.name) || item.count,
  }));

export const RegressionApp: FC = () => {
  const [output, setOutput] = useState('');

  const [scatter, setScatter] = useState<Array<DataSet>>([]);
  const [scatterPower, setScatterPower] = useState<Array<DataSet>>([]);
  const [outliers, setOutliers] = useState<Array<DataSet>>([]);
  const [outliersPower, setOutliersPower] = useState<Array<DataSet>>([]);
```

```
const [lineDataset, setLineDataset] = useState<Array<DataSet>>([]);
const [powerDataset, setPowerDataset] = useState<Array<DataSet>>([]);

const [loadedData, setLoadedData] = useState<Array<IPureDataItem>>();
const [workingData, setWorkingData] = useState<Array<IPureDataItem>>();
const [workingDataPower, setWorkingDataPower] = useState<
  Array<IPureDataItem>
>();

const [predInterval, setPredInterval] = useState<Array<[number, number]>>();
const [confInterval, setConfInterval] = useState<Array<[number, number]>>();

const [dataMap, setDataMap] = useState<Map<string, number>>(new Map());

const clearOutput = useCallback(() => {
  setOutput('');
  setOutliers([]);
  setLineDataset([]);
}, [setOutput, setWorkingData, setOutliers, setLineDataset]);

const addOutput = useCallback(
  (output: string) => {
    setOutput(prev => `${prev}\n${output}`);
  },
  [setOutput]
);

useEffect(() => {
  if (workingData) {
    setScatter(transformForChart(workingData, dataMap));
  }
}, [workingData, setScatter, dataMap]);

useEffect(() => {
  if (workingDataPower) {
    setScatterPower(transformForChart(workingDataPower, dataMap));
  }
}, [workingDataPower, setScatter, dataMap]);

const onLoadData = useCallback(
  (json: string) => {
    try {
      const data: {
        data: Array<IPureDataItem>;
      } = JSON.parse(json);

      console.log('data length', data.data.length);
      const transformedData = data.data
        .filter(i => i.count)
        .filter(i => [''].includes(i.name) === false)
        // .filter(i => i.count > 1000)
        .map(i => ({
          ...i,
          bytes: i.bytes,
        }));

      console.log(
        'md:',
        findMD(transformedData.map(item => Math.log10(item.count)))
      );
```

```
      setLoadedData(transformedData);
      setDataMap(
        new Map(transformedData.map(item => [item.name, item.count]))
      );
      setWorkingData(transformedData);
      setWorkingDataPower(transformedData);
      console.log('transformedData', transformedData);
      clearOutput();
    } catch (e) {
      clearOutput();
      setOutput(
        'Occur error while trying to parse json. Try to check data for correctness.'
      );
    }
  },
  [clearOutput, setLoadedData, setDataMap, setWorkingDataPower]
);

const onTransformData = useCallback(() => {
  if (workingData) {
    setWorkingData(
      workingData.map(item => ({
        ...item,
        count: boxCox([item.count])[0],
        bytes: boxCox([item.bytes])[0],
      }))
    );
  }
}, [workingData, setWorkingData]);

const onCalcLinear = useCallback(() => {
  if (workingData) {
    addOutput('Try to find linear regression.');

    const vecData = workingData.map(
      item => [item.count, item.bytes] as [number, number]
    );

    const model = new LinearRegressionModel(vecData);

    addOutput(`Linear regression have a form: ${model.getStringForm()}`);
    addOutput(`R2: ${model.getR2()}`);
    addOutput(`MMRE: ${model.getMMRE()}`);

    setLineDataset(createLineDataset2(model, vecData));
  }
}, [addOutput, workingData]);

const onFindOutliers = useCallback(
  (
    ModelClass: IRegressionModelConstructor,
    setOutliersLocal: (data: Array<DataSet>) => void,
    setWorkingDataLocal: (data: Array<any>) => void,
    currentWorkingData: Array<IPureDataItem>
  ) => {
    console.log('call find out!');

    const recalculateData = (data: Array<IPureDataItem>) => {
      const transformedData = data.map(
        item => [item.count, item.bytes] as [number, number]
      );
```

```
      const model = new ModelClass(transformedData);
      return model;
    };
    let outliersStep: Array<number> = [];
    const generalOutliers: Array<IPureDataItem> = [];
    let currentModel: IRegressionModel;
    let currentParsedData = currentWorkingData;
    do {
      currentModel = recalculateData(currentParsedData);
      outliersStep = findOutliersDFFIT(
        currentModel,
        currentModel.getRtiThreshold()
      );
      if (outliersStep.length > 0) {
        currentParsedData = currentParsedData.filter((item, index) => {
          if (outliersStep.includes(index)) {
            generalOutliers.push(item);
            return false;
          } else {
            return true;
          }
        });
      }
      console.log('step outliers', outliersStep);
    } while (outliersStep.length !== 0);

    console.log('Found outliers', generalOutliers);
    setOutliersLocal(transformForChart(generalOutliers, dataMap));
    setWorkingDataLocal(
      currentWorkingData.filter(item => !generalOutliers.includes(item))
    );
  },
  [setWorkingData, dataMap]
);
const onFindOutliersLinear = useCallback(
  () =>
    workingData &&
    onFindOutliers(
      LinearRegressionModel,
      setOutliers,
      setWorkingData,
      workingData
    ),
  [onFindOutliers, setOutliers, workingData, setWorkingData]
);
const onFindOutliersPower = useCallback(
  () =>
    workingDataPower &&
    onFindOutliers(
      PowerRegressionModel,
      setOutliersPower,
      setWorkingDataPower,
      workingDataPower
    ),
  [onFindOutliers, setWorkingDataPower, workingDataPower]
);
const onCalcPower = useCallback(() => {
  if (workingDataPower) {
    addOutput('Try to find power regression.');
    const vecData = workingDataPower.map(
      item => [item.count, item.bytes] as [number, number]
```

```
      );
      const model = new PowerRegressionModel(vecData);
      addOutput(`Power regression have a form: ${model.getStringForm()}`);
      addOutput(`R2: ${model.getR2()}`);
      addOutput(`MMRE: ${model.getMMRE()}`);

      setPowerDataset(createLineDataset2(model, vecData));
      setPredInterval(model.getPredictionInterval());
      setConfInterval(model.getConfidenceInterval());
    }
  }, [addOutput, workingDataPower, setPowerDataset, setPredInterval]);
  return (
    <Box padding="20px" className={styles.RegressionApp}>
      <InputRegressionData className={styles.Input} onLoadData={onLoadData} />
      <Controls
        onTransformData={onTransformData}
        fixDistribution={!!loadedData}
        onCalcLinear={onCalcLinear}
        onFindOutliers={onFindOutliersLinear}
      />
      <Box className={styles.OutputAndRegressionCBox}>
        <RegressionControls
          calcLinear={onCalcLinear}
          findOutliersLinear={onFindOutliersLinear}
          calcPower={onCalcPower}
          disabled={!loadedData}
          className={styles.RegressionControls}
          findOutliersPower={onFindOutliersPower}
        />
        <OutputData className={styles.Output} value={output} />
      </Box>
      <Box
        position="relative"
        className={styles.ChartContainer}
        height="47vh"
        width="90vw"
      >
        <Box width="45vw">
          <RegressionChartAdapter
            line={lineDataset}
            scatter={scatter}
            outliers={outliers}
            lineName={'Лінійна'}
          />
        </Box>
        <Box width="45vw">
          <RegressionChartAdapter
            line={powerDataset}
            scatter={scatterPower}
            outliers={outliersPower}
            lineName={'Нелінійна'}
            predictionInterval={predInterval}
            confidenceInterval={confInterval}
          />
        </Box>
      </Box>
    </Box>
  );
};
```

Лістинг 6 – Файл RegressionChart.tsx

```
import React, { FC, useEffect, useState } from 'react';
import { DataSet } from '../../commonTypes';
import { Chart } from 'chart.js';

type Props = {
  scatterPoints: Array<DataSet>;
  outliersPoints: Array<DataSet>;
  linePoints: Array<DataSet>;
  lineName: string;
  predictionInterval?: Array<DataSet>;
  confidenceInterval?: Array<DataSet>;
};

enum EDatasetTypes {
  SCATTER = 'Розкид даних',
  LINE = 'Лінійна',
  OUTLIERS = 'Викиди',
  PREDICTED_INTERVAL = 'Інтервал передбачення',
  CONFIDENCE_INTERVAL = 'Довірчий інтервал',
}

(window as any).naxer = [];

export const RegressionChart: FC<Props> = ({
  linePoints,
  scatterPoints,
  outliersPoints,
  lineName,
  predictionInterval,
  confidenceInterval,
}) => {
  const [canvas, setCanvas] = useState<HTMLCanvasElement | null>(null);
  const [chart, setChart] = useState<Chart>();

  useEffect(() => {
    const ctx = canvas?.getContext('2d');

    if (ctx) {
      const chart = new Chart(ctx, {
        type: 'scatter',
        data: {
          datasets: [
            {
              pointBackgroundColor: '#2962ff',
              label: EDatasetTypes.SCATTER,
              backgroundColor: '#2962ff',
              data: [],
            },
            {
              pointBackgroundColor: '#c50b0b',
              label: EDatasetTypes.OUTLIERS,
              backgroundColor: '#ce0000',
              data: [],
              pointStyle: 'crossRot',
              borderColor: '#ce0000',
              pointRadius: 6,
            },
```

```
    {
     type: 'line',
     label: lineName,
     data: [],
     fill: false,
     borderColor: '#ff143c',
    },
    {
     // pointStyle: 'circle',
     label: EDatasetTypes.PREDICTED_INTERVAL,
     // fill: false,
     data: [],
     borderColor: '#7af02c',
    },
    {
     // pointStyle: 'dash',
     label: EDatasetTypes.CONFIDENCE_INTERVAL,
     fill: false,
     data: [],
     borderColor: '#19cbe2',
    },
  ],
},
options: {
  responsive: true,
  maintainAspectRatio: false,
  tooltips: {
    callbacks: {
      label: function (tooltipItem, data) {
        console.log('tooltipItem', tooltipItem);
        console.log('data', data);
        // @ts-ignore
        let label = data.datasets[tooltipItem.datasetIndex].label || '';

        if (label) {
          label += ': ';
        }

        if (
          data.datasets &&
          tooltipItem.datasetIndex !== undefined &&
          tooltipItem.index !== undefined
        ) {
          const dataList = data.datasets[tooltipItem.datasetIndex].data;

          if (Array.isArray(dataList) && dataList[tooltipItem.index]) {
            const item = dataList[tooltipItem.index];
            console.log('item', item);
            if ((item as any).label && (item as any).classCount) {
              (window as any).naxer.push((item as any).label);

              return `${(item as any).label}: [${tooltipItem.x}, ${
                (item as any).classCount
              }]`;
            }
          }
        }
        return label;
      },
    },
  },
```

```
      },
    });

    setChart(chart);
  }
 }, [setChart, canvas, lineName]);

 useEffect(() => {
  if (chart && chart.data.datasets) {
   chart.data.datasets.forEach(dataset => {
    switch (dataset.label) {
     case EDatasetTypes.SCATTER:
      dataset.data = scatterPoints;
      break;
     case EDatasetTypes.OUTLIERS:
      dataset.data = outliersPoints;
      break;
     case lineName:
      dataset.data = linePoints;
      break;
     default:
      break;
    }
   });
   chart.update();
  }
 }, [scatterPoints, linePoints, outliersPoints, chart]);

 useEffect(() => {
  if (chart && chart.data.datasets) {
   if (predictionInterval) {
    chart.data.datasets.forEach(dataset => {
     switch (dataset.label) {
      case EDatasetTypes.PREDICTED_INTERVAL:
       dataset.data = predictionInterval;
       break;
      default:
       break;
     }
    });
   }

   if (confidenceInterval) {
    chart.data.datasets.forEach(dataset => {
     switch (dataset.label) {
      case EDatasetTypes.CONFIDENCE_INTERVAL:
       dataset.data = confidenceInterval;
       break;

      default:
       break;
     }
    });
   }

   chart.update();
  }
 }, [chart, predictionInterval, confidenceInterval, linePoints]);

 return <canvas ref={setCanvas}></canvas>;
};
```

Лістинг 7 – Файл RegressionChartAdapter.tsx

```
import React, { FC, useEffect, useMemo, useState } from 'react';
import { DataSet } from '../../commonTypes';
import { RegressionChart } from './RegressionChart';

type Props = {
  scatter: Array<DataSet>;
  outliers: Array<DataSet>;
  line: Array<DataSet>;
  lineName: string;
  predictionInterval?: Array<[number, number]>;
  confidenceInterval?: Array<[number, number]>;
};

const toDataSet = (data: [number, number]) => ({
  x: data[0],
  y: data[1],
});

export const RegressionChartAdapter: FC<Props> = ({
  scatter,
  line,
  outliers,
  lineName,
  predictionInterval,
  confidenceInterval,
}) => {
  const [scatterDataPoints, setScatterDataPoints] = useState<Array<DataSet>>(
    []
  );

  const [lineDataPoints, setLineDataPoints] = useState<Array<DataSet>>([]);

  const [outlierDataPoints, setOutlierDataPoints] = useState<Array<DataSet>>(
    []
  );

  const predTransformInterval = useMemo(() => {
    return predictionInterval?.map(toDataSet);
  }, [predictionInterval]);

  const confTransformInterval = useMemo(() => {
    return confidenceInterval?.map(toDataSet);
  }, [confidenceInterval]);

  useEffect(() => {
    setScatterDataPoints(scatter);
  }, [scatter, setScatterDataPoints]);

  useEffect(() => {
    setLineDataPoints(line);
  }, [line, setLineDataPoints]);

  useEffect(() => {
    setOutlierDataPoints(outliers);
  }, [outliers, setOutlierDataPoints]);

  return (
    <RegressionChart
```

```
      outliersPoints={outlierDataPoints}
      linePoints={lineDataPoints}
      scatterPoints={scatterDataPoints}
      lineName={lineName}
      predictionInterval={predTransformInterval}
      confidenceInterval={confTransformInterval}
    />
  );
};
```

Лістинг 8 – Файл LineRegression.ts

```
import { mean, sum } from 'mathjs';
import { linear, Result } from 'regression';
import { IRegressionModel } from './interfaces';

export class LinearRegressionModel implements IRegressionModel {
  protected data: Array<[number, number]>;
  protected linear: Result;

  constructor(inputData: Array<[number, number]>) {
    this.data = [...inputData];

    this.linear = linear(this.data, {
      precision: 5,
    });
  }

  getRtiThreshold() {
    return 3;
  }

  getInputData = () => [...this.data];

  fx(x: number) {
    return this.linear.predict(x)[1];
  }

  getR2 = () => this.linear.r2;

  getResidual() {
    return this.data.map(item => item[1] - this.linear.predict(item[0])[1]);
  }

  getStringForm() {
    return this.linear.string;
  }

  getMSE() {
    return sum(...this.getResidual().map(r => r * r)) / this.data.length;
  }

  getT() {
    return 1.5;
  }

  getConfidenceInterval() {
    const n = this.data.length;

    const mse = this.getMSE();
```

```
    const t = this.getT();

    const meanX = mean(this.data.map(([x]) => x));

    const rsx = sum(...this.data.map(([x]) => Math.pow(x - meanX, 2)));

    const getInterval = (x: number) =>
      Math.sqrt(mse * (1 / n + Math.pow(x - meanX, 2) / rsx));

    return this.data
      .map(([x, y]) => {
        const r95 = t * getInterval(x);
        return [
          [x, this.linear.predict(x)[1] + r95],
          [x, this.linear.predict(x)[1] - r95],
        ];
      })
      .flat() as Array<[number, number]>;
  }

  getPredictionInterval() {
    const n = this.data.length;

    const mse = this.getMSE();

    const t = this.getT();

    const meanX = mean(this.data.map(([x]) => x));

    const rsx = sum(...this.data.map(([x]) => Math.pow(x - meanX, 2)));

    const getInterval = (x: number) =>
      Math.sqrt(mse * (1 + 1 / n + Math.pow(x - meanX, 2) / rsx));

    return this.data
      .map(([x, y]) => {
        const r95 = t * getInterval(x);

        return [
          [x, this.linear.predict(x)[1] + r95],
          [x, this.linear.predict(x)[1] - r95],
        ];
      })
      .flat() as Array<[number, number]>;
  }

  getMMRE() {
    console.log('this.getResidual()', this.getResidual());
    return (
      sum(
        ...this.getResidual().map((r, index) =>
          Math.abs(r / this.data[index][1])
        )
      ) / this.data.length
    );
  }
}
```

Лістинг 9 – Файл PowerRegression.ts

```
import { sum } from 'mathjs';
```

```
import { LinearRegressionModel } from './LinearRegressionModel';

export class PowerRegressionModel extends LinearRegressionModel {
  constructor(private inputData: Array<[number, number]>) {
    super(inputData.map(([x, y]) => [Math.log10(x), Math.log10(y)]));
  }
  fx(x: number) {
    return Math.pow(
      10,
      LinearRegressionModel.prototype.fx.call(this, Math.log10(x))
    );
  }
  getRtiThreshold() {
    return 2;
  }
  getStringForm() {
    return `y = 10^${LinearRegressionModel.prototype.getStringForm
      .call(this)
      .replace('x', 'log(x)')
      .replace('y = ', '')}`;
  }
  getConfidenceInterval() {
    const res = LinearRegressionModel.prototype.getConfidenceInterval
      .call(this)
      .map(([x, y]) => [Math.pow(10, x), Math.pow(10, y)]) as Array<
      [number, number]
    >;
    const output = res.reduce((acc, [, y], index) => {
      const nI = Math.floor(index / 2);
      if (acc[nI]) {
        acc[nI]['r-'] = y;
      } else {
        acc[nI] = {
          'r+': y,
          'r-': 0,
        };
      }
      return acc;
    }, [] as Array<{ 'r+': number; 'r-': number }>);

    console.log('getConfidenceInterval', JSON.stringify({ 'data': output }));

    return LinearRegressionModel.prototype.getConfidenceInterval
      .call(this)
      .map(([x, y]) => [Math.pow(10, x), Math.pow(10, y)]) as Array<
      [number, number]
    >;
  }
  getPredictionInterval() {
    const res = LinearRegressionModel.prototype.getPredictionInterval
      .call(this)
      .map(([x, y]) => [Math.pow(10, x), Math.pow(10, y)]) as Array<
      [number, number]
    >;

    const output = res.reduce((acc, [, y], index) => {
      const nI = Math.floor(index / 2);
      if (acc[nI]) {
        acc[nI]['r-'] = y;
      } else {
        acc[nI] = {
```

```
          'r+': y,
          'r-': 0,
        };
      }
      return acc;
    }, [] as Array<{ 'r+': number; 'r-': number }>);
    console.log('getPredictionInterval', JSON.stringify(output));
    return LinearRegressionModel.prototype.getPredictionInterval
      .call(this)
      .map(([x, y]) => [Math.pow(10, x), Math.pow(10, y)]) as Array<
      [number, number]
    >;
  }
  getMMRE() {
    const residual = this.inputData.map(([x, y]) => y - this.fx(x));
    console.log('residual', residual);

    return (
      sum(
        ...residual.map((r, index) => Math.abs(r / this.inputData[index][1]))
      ) / this.data.length
    );
  }
}
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1 Загальні відомості

Розроблене ПЗ у вигляді веб-додатку для розрахунку регресійного рівняння та його удосконалення через зворотне перетворення на знаходження викидів.

Дружній користувацький інтерфейс дозволить користувачеві використовувати програмне забезпечення без додаткового навчання.

Для розробки програмного забезпечення обрано наступні засоби:

- в якості мови програмування обрано JavaScript;
- в якості середовища розробки обрано VSCode.

Для використання розробленого ПЗ необхідна наявність встановленого браузера та з’єднання з мережею Інтернет.

2 Функціональне призначення

Програмне забезпечення «Удосконалення рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на java» призначене для автоматизації роботи по знаходженню та вдосконалені рівняння регресії. Програма дозволяє на основі вхідних даних робити аналіз цих даних на нормальний розподіл після цього, робити їх перетворення для усунення недоліків, будувати лінійну регресію та виводити її графік, вираховувати коефіцієнт детермінації моделі, шукати викиди та роботи перетворення лінійної моделі до нелінійної, та робити передбачення розміру пз на основі побудованої моделі.

4 Технічні та програмні засоби

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

Для PC:

- Windows 7 і більш пізні;
- Процесор Intel core i3 2,33, AMD Ryzen 3(або аналогічний);
- 1 Гб оперативної пам'яті;
- 512 Мб відео пам'яті;
- Firefox 2.x-3.x, Opera 9.5 або більш пізньої версії, Google Chrome.

Для Macintosh:

- Mac OS X v10.4 і пізніші;
- Процесор IntelCore i3 з тактовою частотою не менше 1,8 ГГц;
- 512 Мб оперативної пам'яті;
- 512 Мб відео пам'яті;
- Браузери: Firefox 2.x, Firefox 3.x, Opera 9.5, Safari 3.x

Для Linux:

- Процесор Intel core i3 2,33, AMD Ryzen 3(або аналогічний);
- 1 Гб оперативної пам'яті;
- 512 Мб відео пам'яті;
- Браузери: Firefox 2.x, Firefox 3.x, SeaMonkey 1.11;

Для Android:

- 512 Мб оперативної пам'яті;
- Android 4.0 або більш пізньої версії.

Для iOS:

- 512 Мб оперативної пам'яті;
- iOS 7.0 або більш пізньої версії.

5 Виклик та завантаження

Програмне забезпечення встановлюється на віддалений сервер та доступний з будь-якої точки світу через мережу Інтернет за певним посиланням у браузері.

6 Вхідні та вихідні дані

Вхідні дані:

Вхідними даними є дані початкові вхідні дані про проекти, з зазначеною структурою.

Вихідні дані:

Вихідними даними є результати розрахунків регресії та виведення графіків на екран.

ДОДАТОК Г – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення «Удосконалення рівняння регресії для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на java».

Областю застосування програмного забезпечення є автоматичний аналіз вхідних даних, пошук викидів, розрахування моделі регресії та її удосконалення рівняння регресії.

2 Ціль випробувань

Ціллю випробувань, що проводяться за даними програмою і методикою, є визначення функціональної працездатності ПЗ на етапі проведення випробувань.

Програма випробувань повинна засвідчити працездатність в відповідності до функціонального призначення та вимог, визначених в програмному документі «Технічне завдання».

3 Вимоги до програми

Програмний інтерфейс має надавати зручний доступ до всіх основних функцій, що потребуються в процесі, та надавати розробнику можливість створити програмне забезпечення для проведення опитувань, створивши тільки свій користувацький інтерфейс.

Створене програмне забезпечення має виконувати такі функції:

– завантаження даних;

– перетворення даних;
– розрахунок лінійної регресії;
– розрахунок нелінійної регресії;
– вивід результатів;
– зображення графіку.

При проведенні випробувань функціональні характеристики програмного забезпечення підлягають перевірці на відповідність вимогам, викладеним в документі «Технічне завдання».

4 Вимоги до програмної документації

4.1 Перелік програмної документації, що пред’являється на випробування

Програмна документація повинна містити:
– технічне завдання;
– інструкцію користувача;
– тексти програмних модулів;
– програму та методику випробувань.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не пред’являються.

5 Засоби та порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Випробування програмного забезпечення повинні проводитися на цільовому обладнанні сторони замовника. До складу технічних засобів має входити не менше двох серверів, що задовольняють таким умовам:

- не менш ніж 4 Gb RAM;
- не менш ніж 10 Gb SSD\HDD;
- процесор з тактовою частотою не нижчою за 3,5 GHz.

Вимоги до клієнтських технічних засобів:

- процесор з тактовою частотою не меншою за 1 GHz;
- не менше ніж 1 Gb RAM;
- не менш ніж 1 Gb SSD\HDD.

5.2 Програмні засоби, що використовуються під час випробувань

Сервером мають використовуватися такі програмні засоби:

- операційна система Linux Ubuntu 11.0+ / Debian 7.0+ / Windows Server 2008+;
- сервер NGINX/Apache/NodeStaticServer.

Для веб-клієнту продукту необхідні такі програмні засоби:

- браузер Google Chrome 50.1+, Safari 8, Edge 2.0, Firefox 30+.

5.3 Порядок проведення випробувань

Випробування проводяться у два етапи.

I етап – ознайомлення з програмним забезпеченням.

II етап – випробування програмного забезпечення.

5.3.1 Перелік перевірок, що виконуються на першому етапі випробувань

Перелік перевірок, що виконуються на першому етапі випробувань повинен включати в себе:

- перевірку комплектності програмної документації;
- перевірку комплектності і складу технічних та програмних засобів.

Методики проведення перевірок, що входять в перелік першого етапу випробувань, викладені в даному програмному документі в розділі «Методи випробувань».

5.3.2 Перелік перевірок, що проводяться на другому етапі випробувань

Перелік перевірок, що проводяться на другому етапі випробувань повинен включати в себе:

- перевірку відповідності технічних характеристик програми;
- перевірку відповідності вимогам функціонального призначення програми.

Методики проведення перевірок, що входять в перелік другого етапу випробувань, викладені в даному програмному документі в розділі «Методи випробувань».

5.4 Кількісні і якісні характеристики, що підлягають оцінці

5.4.1 Кількісні характеристики, що підлягають оцінці

При проведенні приймально-здавальних випробувань оцінці підлягають такі кількісні характеристики:

- комплектність програмної документації;

– комплектність складу технічних і програмних засобів.

5.4.2 Якісні характеристики, що підлягають оцінці

При проведенні приймально-здавальних випробувань оцінці підлягають якісні (функціональні) характеристики програми. Перевірці підлягає здатність виконання програмою функцій, викладених в документі «Технічне завдання».

5.5 Умови проведення випробувань

5.5.1 Кліматичні умови

Вимоги до кліматичних умов не висуваються.

5.5.2 Умови початку і завершення окремих етапів випробувань

Необхідною і достатньою умовою завершення першого етапу випробувань і початку другого етапу є успішне закінчення перевірок, що проводяться на першому етапі.

Умовою завершення другого етапу випробувань є успішне закінчення перевірок, що проводяться на другому етапі.

5.5.3 Обмеження в умовах випробувань

Кліматичні умови експлуатації, при яких повинні забезпечуватись задані характеристики, повинні відповідати вимогам, що пред’являються технічним засобам в розділі умов їх експлуатації.

5.5.4 Заходи, що забезпечують безпеку і безаварійність випробувань

При проведенні випробувань повинно бути забезпечене дотримання вимог безпеки, встановлених ГОСТ 12.2.007.0-75, «Правилами техніки безпеки при експлуатації електроустановок споживачів», і « Правилами технічної експлуатації електроустановок споживачів».

5.6 Перелік робіт, що проводяться після завершення випробувань

Хід проведення приймально-здавальних випробувань документують за допомогою протоколу проведення випробувань.

У разі успішного проведення випробувань у повному обсязі, на підставі протоколу проведення випробувань, виконавець спільно з замовником підписують акт приймання-здачі програмного забезпечення в експлуатацію.

У разі знаходження помилок під час прийому програмного виробу, складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації-замовника та організації-розробника. Розробник повинен протягом двох тижнів виправити зазначені помилки та оповістити замовника про повторне проведення перевірки.

6 Методи випробувань

6.1 Методика проведення перевірки комплектності програмної документації

Перевірка комплектності програмної документації на програмний виріб проводиться візуально представником служби, відповідальної за експлуатацію.

В ході перевірки зіставляється склад і комплектність поданої програмної документації з переліком програмної документації, наведеними в пункті «Склад програмної документації, що пред'являється на випробування» цього документа.

Перевірка вважається завершеною в разі відповідності складу та комплектності представленої програмної документації з переліком програмної документації, наведеним в зазначеному вище пункті.

За результатами проведення перевірки, представник служби, відповідальної за експлуатацію вносить запис до протоколу випробувань – ”Комплектність програмної документації відповідає (не відповідає) вимогам пункту «Склад програмної документації, що пред'являється на випробування » даного документа”.

6.2 Методика проведення перевірки комплексності і складу технічних та програмних засобів

Перевірка комплектності та складу технічних і програмних засобів проводиться візуально представником служби, відповідальної за експлуатацію. В ході перевірки зіставляється склад і комплектність представлених технічних і програмних засобів з переліком технічних і програмних засобів, наведеним у пунктах «Технічні засоби, що використовуються під час випробувань» та «Програмні засоби, що використовуються під час випробувань» даного документа. Комплектність програмних засобів проводиться також візуально.

Перевірка вважається завершеною в разі відповідності складу та комплектності представлених технічних і програмних засобів з переліком технічних і програмних засобів, наведених в пунктах «Технічні засоби, що використовуються під час випробувань» та «Програмні засоби, що використовуються під час випробувань» даного документа.

За результатами проведення перевірки представник служби, відповідальної за експлуатацію, вносить запис до протоколу випробувань – “Комплектність технічних і програмних засобів відповідає (не відповідає) вимогам пунктів «Технічні засоби, що використовується під час випробувань» та «Програмні засоби, що використовуються під час випробувань» даного документа”.

6.3 Методика проведення перевірки функціональних характеристик програмного забезпечення

Під час випробувань проводиться повне функціональне тестування, відповідно до вимог вказаних в пункті 3 та технічному завданні.

При проведенні приймальних випробувань доступ до програмного продукта надається досліднику-соціологу, який працює із програмним забезпеченням, виконуючи свої службові обов’язки, чим забезпечує повнофункціональне тестування.

Перевірка вважається пройденою, якщо функціональні характеристики відповідають вимогам, вказаним в документі «Технічне завдання».

За результатами проведення перевірки представник служби, відповідальної за експлуатацію, вносить запис до протоколу випробувань - ”Функціональні вимоги відповідають (не відповідають) вимогам пункту «Вимоги до програми» цього документа”.

ДОДАТОК Д – ІНСТРУКЦІЯ КОРИСТУВАЧА

Завантаживши веб-додаток, відвідувачу користувач бачить інтерфейс, на рисунку Д.1 представлена сторінка:

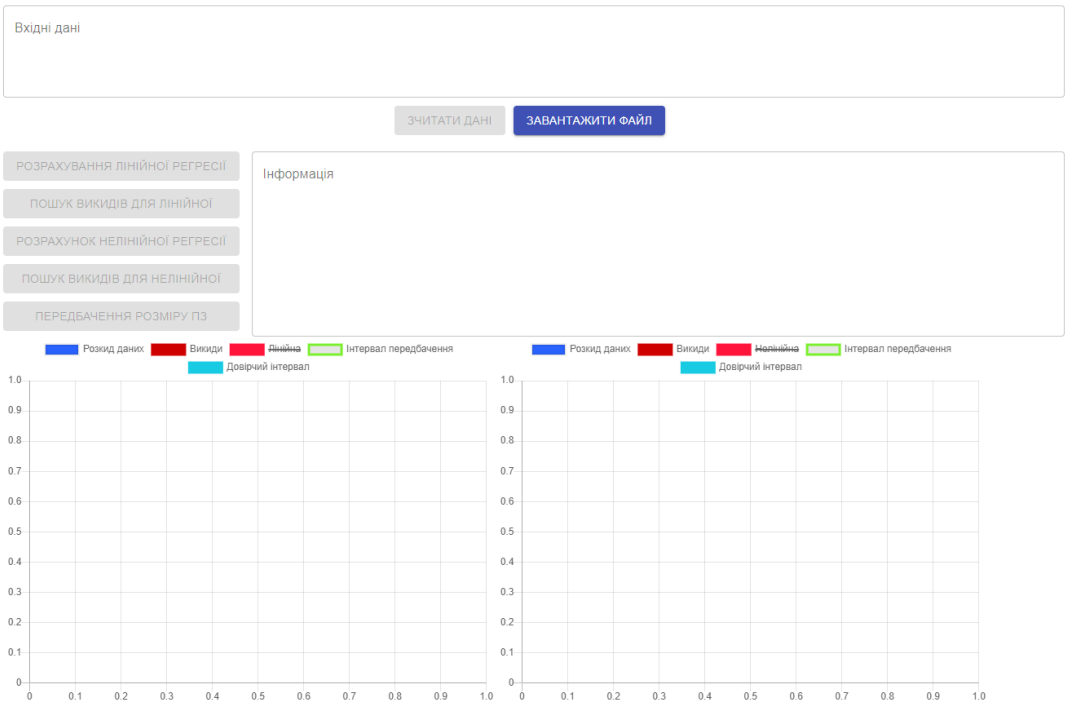


Рисунок Д.1 – Головний інтерфейс програми

Після завантаження даних зручним для нас способом, на графіку зображується розкид точок та розблокуються основні функції програми надаючи змогу натискати інші кнопки.

Як показано на рисунку Д.2.

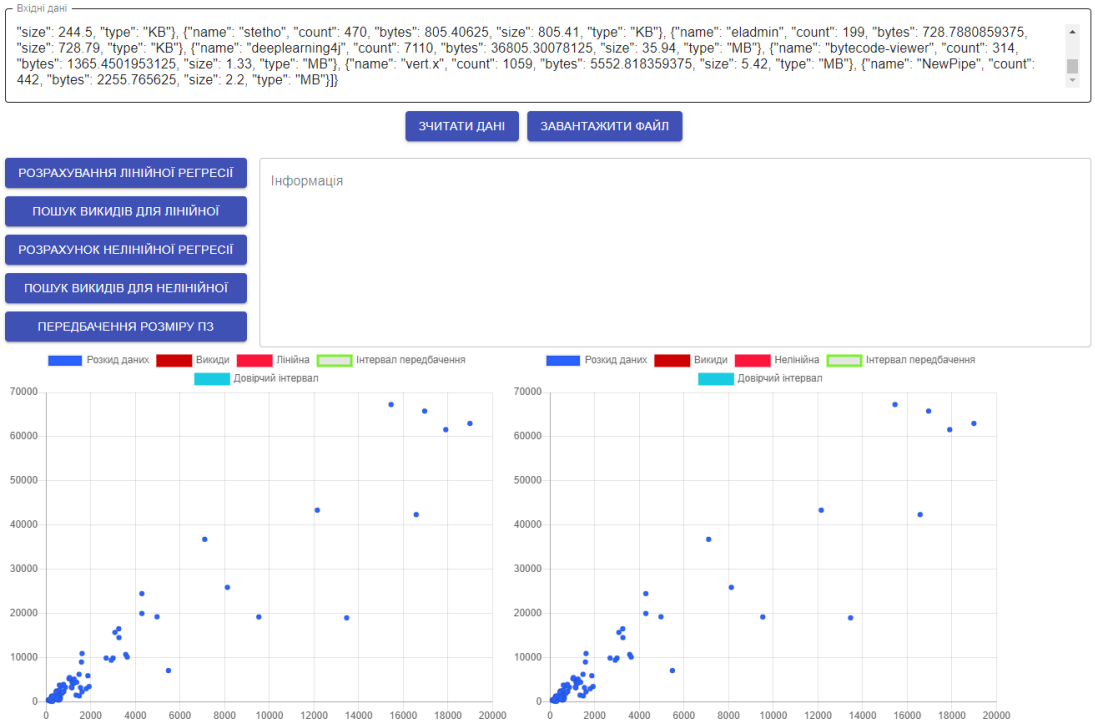


Рисунок Д.2 – Інтерфейс після завантаження даних

Для розрахунку регресії достатньо натиснути кнопку «Розрахування лінійної регресії» або «Розрахунок нелінійної регресії». Після розрахунку регресії вся інформація буде виведена на графік та в поле інформація.

Даний випадок зображений на рисунку Д.3.

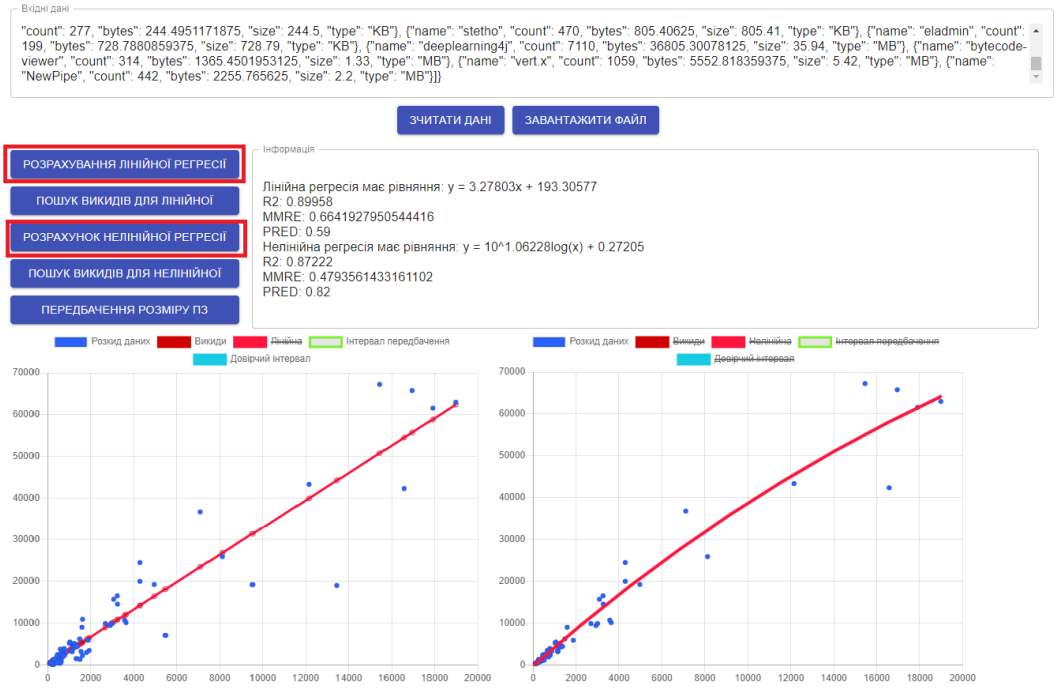


Рисунок Д.3 – Розрахунок моделі регресії та зображення її на графіку

Також в програмному забезпеченні є функція пошуку залишків на основі «видалених залишків», для цього є окремі кнопки «Пошук викидів для лінійної» та «Пошук викидів для нелінійної».

Пошук викидів зображений на рисунку Д.4.

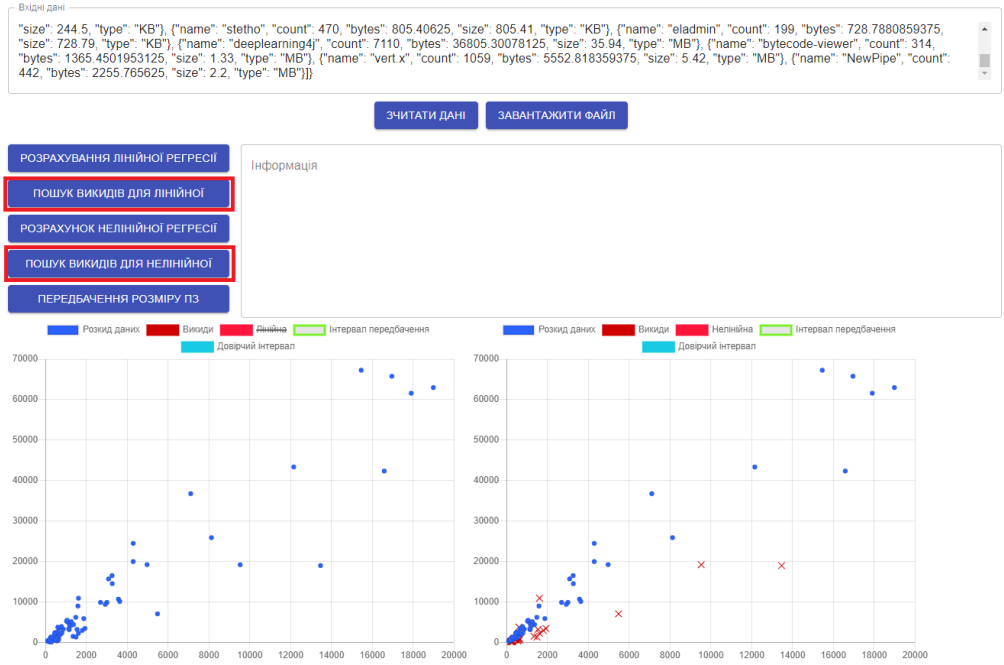


Рисунок Д.4 – Пошук викидів моделі

Після пошуку викидів, ми маємо змогу перерахувати нашу регресійну модель для отримання новою на основі даних що лишились. Також зверху графіку де написані заголовки для даних, можна натискати на кожний з них для переключення показу (показано на рисунку Д.5).

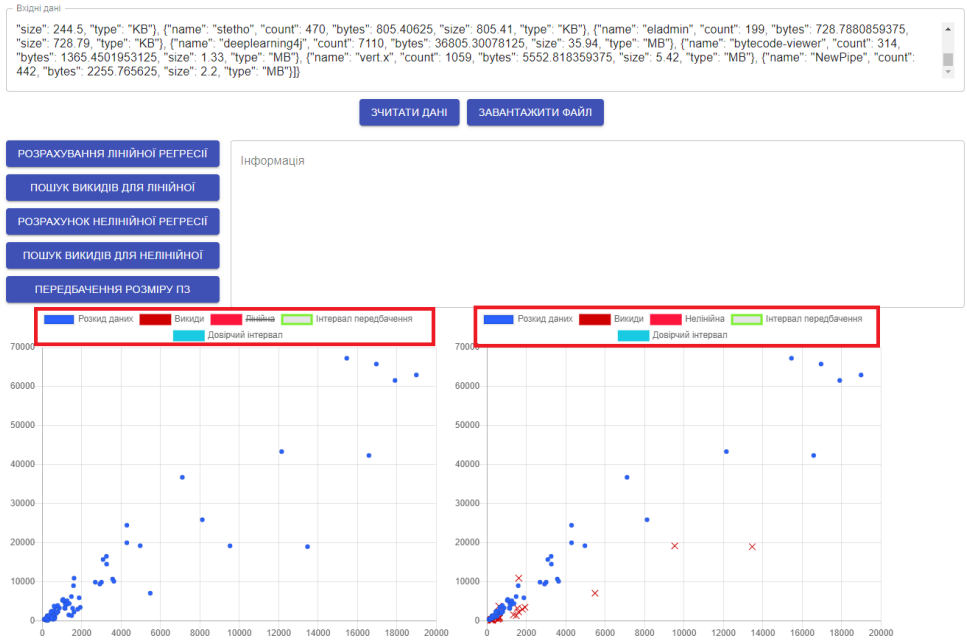


Рисунок Д.5 – Фільтрація даних на графіку

Були наведені основні функції програмного додатку.

Для передбачення розміру пз потрібно натиснути на кнопку «Передбачення розміру ПЗ» (див. рис. Д.6).

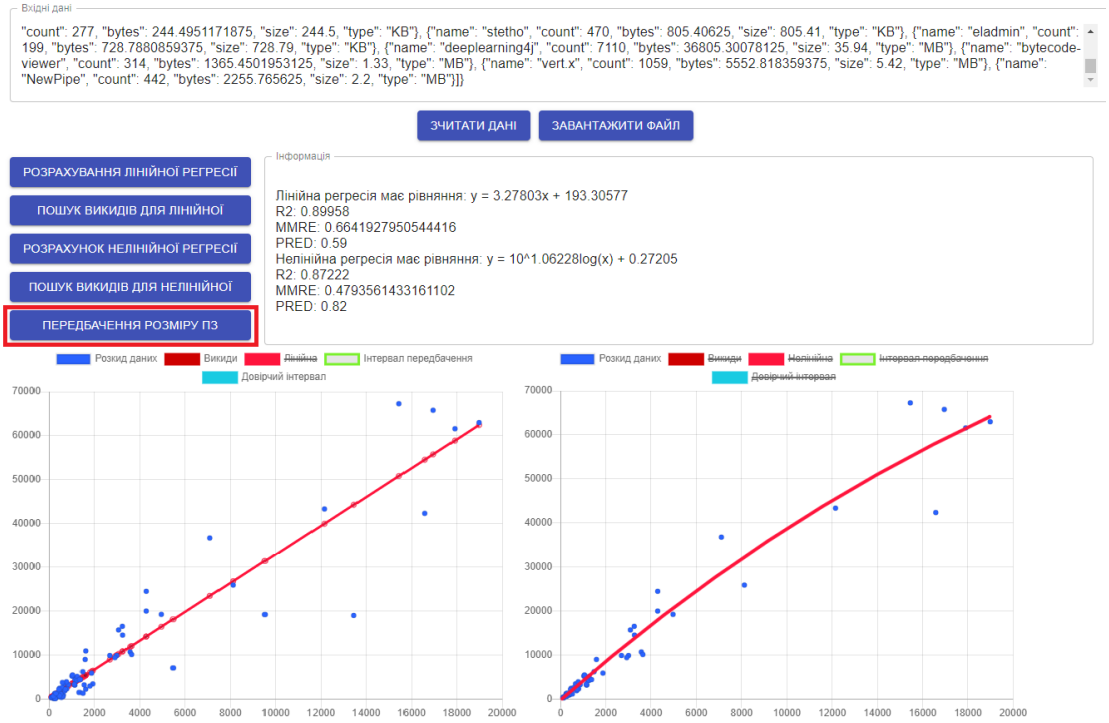


Рисунок Д.6 – Кнопка для передбачення розміру ПЗ, та розрахунку інтервалу передбачення та довірчого інтервалу