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 регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, та розробка програми для її реалізації»

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

Абаза Є.В.
(підпис, прізвище та ініціали)
Керівник Макарова Л.М.
(підпис, прізвище та ініціали)
Рецензент Приходько С.Б.
(підпис, прізвище та ініціали)
Завідувач кафедри Приходько С.Б.
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Абаза Євгеній Васильович
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи «Удосконалення регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, та розробка програми для її реалізації»
керівник роботи Макарова Лідія Миколаївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи

4.Зміст магістерської роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації.)
• Огляд літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту програмного забезпечення
• Організаційно-економічний розділ
• Розділи з охорони праці та охорони навколишнього середовища
• Висновки
• Список використаних джерел
• Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік графічного матеріалу

_____
_____
_____
_____
_____
_____

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ Абаза Є.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Макарова Л.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Абаза Євгеній Васильович

«Удосконалення регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, та розробка програми для її реалізації»

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

Обсяг роботи: 110 стор., 7 табл., 16 рис., 32 використаних джерела, 5 додатків.

Актуальність теми роботи: полягає в тому, що оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, є актуальною задачею, виходячи з такої популярності та поширеності цього фреймворку.

Мета та завдання дослідження: підвищення достовірності оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Об’єкт дослідження: процес оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Предмет дослідження: однофакторна нелінійна регресійна модель для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Методи дослідження: методи теорії ймовірностей та математичної статистики, математичного моделювання, регресійного аналізу, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: полягає в удосконаленні однофакторної нелінійної регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap у порівнянні з існуючою моделлю.

Практичне значення одержаних результатів: ПЗ для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, яке розроблено в рамках кваліфікаційної роботи, дозволить автоматизувати та скоротити час відповідних розрахунків.

Апробація результатів досліджень: Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VI міжнародній науково-технічній конференції «Комп'ютерне моделювання та оптимізація складних систем» (м. Дніпро, 4-6 листопада 2020 р.).

Публікації: Основні результати кваліфікаційної роботи викладено у 1 науковій праці – тезах конференції.

Ключові слова: регресійна модель, нормалізуюче перетворення, десятковий логарифм, розмір програмного забезпечення, веб-застосунок, фреймворк Bootstrap.

ABSTRACT

Abaza Yevhenii Vasylovych

«Improvement of the regression model for size estimating of web applications written using Bootstrap framework 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.

Volume of work: 110 pages of typewritten text, 7 tables, 16 figures, a list of references from 32 names, 5 appendices.

Relevance of the theme: is that estimating the size of web applications developed using the Bootstrap framework is an urgent task based on the popularity and prevalence of this framework.

The goal and objectives of the study: increase the accuracy of estimating the size of web applications written using the Bootstrap framework.

The object of the study: improving the size of web applications that are written using the Bootstrap framework.

The subject of the study: a univariate nonlinear regression model for estimating the size of web applications written using the Bootstrap framework.

Research methods: methods of probability theory and mathematical statistics, mathematical modeling, regression analysis, object-oriented programming.

The scientific novelty of the obtained results: is to improve the univariate nonlinear regression model for estimating the size of web applications written using the Bootstrap framework by using a normalizing transformation based on the decimal logarithm, which made it possible to increase the accuracy of estimating the size of web applications written using the Bootstrap framework compared to the existing model.

The practical significance of the obtained results: Software for estimating the size of web applications written using the Bootstrap framework developed in the course of the qualification work will allow to automate and reduce the time of corresponding calculations.

Approbation of research results: The main provisions and results of the research presented in the qualification work were tested at the VI international scientific and technical conference «Computer modeling and optimization of complex systems» (Dnipro, November 4 – 6, 2020).

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

Keywords: regression model, normalizing transformation, decimal logarithm, software size, web application, Bootstrap framework.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВВ	випадкова величина
ЕД	емпіричні дані
ПЗ	програмне забезпечення
COCOMO	Constructive Cost Model
LOC	Lines Of Code
UI	User Interface

ВСТУП

Актуальність теми. Світ давно усвідомив, що майбутнє за інформаційними технологіями і штучним інтелектом. Вже сьогодні ці дві сфери є основним двигуном прогресу. Чималу частку займає розвиток мережі Інтернет та створення нових сайтів. Згідно зі статистикою, в 2018 році було створено близько 5 мільйонів нових сайтів. Аналітики прогнозують і подальше зростання їх кількості. При цьому зростає і кількість розробників веб-застосунків [1].

Веб-застосунок – це програма, яка призначена для користувача, головна частина якої знаходиться на виділеному сервері, а призначений для користувача інтерфейс UI він бачить у браузері у вигляді веб-сторінки [2]. Веб-додатки складаються з двох частин: серверної та клієнтської. На клієнтській частині найчастіше застосовують XHTML, HTML, CSS, Adobe Flash (Flex), ActiveX, Java, Javascript, Silverlight.

Будь-який веб-розробник і верстальник рано чи пізно замислюється, як спростити і прискорити процес верстання сайту. Поява веб-фреймворків сильно змінила світ розробки веб-застосунків і стала невід'ємною частиною процесу розробки [3]. Веб-фреймворк – інструмент, який полегшує процес написання і запуску веб-застосунку. При його використанні не потрібно самостійно писати велику кількість коду і витрачати час на пошук потенційних прорахунків і помилок.

Для розробки веб-застосунків використовують різні фреймворки в залежності від мови програмування та технології розробки. Якщо говорити безпосередньо про верстання, то тут необхідно використовувати в першу чергу HTML/CSS фреймворки. Найбільш популярних з них – Bootstrap [4], його спільнота в 3 – 5 разів більше, ніж у його конкурентів – Foundation, UIkit, Semantic UI, InK та інших.

Виходячи з такої популярності та поширеності фреймворку Bootstrap, оцінювання розміру веб-застосунків, розроблених з його використанням, є актуальною задачею.

Для оцінювання розміру веб-застосунків існують лише загальні критерії, основані на кількості рядків коду та кількості компонентів проекту. Але більшість з них стають непридатними для використання через брак даних або ресурсів.

Актуальність задачі також полягає в тому, що в епоху інформаційних технологій, що швидко розвиваються, постійно зростаючої кількості проектів в галузі розробки ПЗ, дуже важливим стає вміння оцінити на ранніх етапах можливі вигоди і збитки від проекту, проаналізувати можливі сценарії розвитку подій [5].

Мета і завдання дослідження. Мета: підвищення достовірності оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі моделі для оцінювання розміру ПЗ веб-застосунків, порівняти їх;

– обґрунтувати необхідність удосконалення регресійної моделі для оцінювання розміру ПЗ веб-застосунків;

– розробити удосконалену регресійну модель для оцінювання розміру веб-застосунків із застосуванням нормалізуючих перетворень;

– побудувати удосконалену регресійну модель для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap:

– обрати нормалізуюче перетворення;

– перевірити вихідні ЕД на викиди;

– нормалізувати вихідні ЕД, використовуючи обране нормалізуюче перетворення;

– побудувати лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– перейти від лінійної регресії до нелінійної та побудувати рівняння регресії, довірчий інтервал та інтервал прогнозування для вихідних ЕД;

– перевірити адекватність побудованої регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

– розробити ПЗ для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Об’єктом дослідження є процес оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Предметом дослідження є однофакторна нелінійна регресійна модель для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Методи дослідження. Для вирішення поставлених завдань були застосовані методи теорії ймовірностей та математичної статистики, математичного моделювання, регресійного аналізу, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в удосконаленні однофакторної нелінійної регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap у порівнянні з існуючою моделлю.

Практичне значення одержаних результатів. ПЗ для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, яке розроблено в рамках кваліфікаційної (магістерської) роботи, дозволить автоматизувати та скоротити час відповідних розрахунків.

Особистий внесок здобувача. Кваліфікаційна (магістерська) робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником магістерської роботи [6], здобувачеві належить: побудова удосконаленої однофакторної нелінійної регресійної моделі.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у кваліфікаційній (магістерській) роботі, пройшли апробацію на VI міжнародній науково-технічній конференції «Комп'ютерне моделювання та оптимізація складних систем» (м. Дніпро, 4 – 6 листопада 2020 р.).

Публікації. Основні результати кваліфікаційної (магістерської) роботи викладено у 1 науковій праці – тезах конференції.

1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ЗАСТОСУНКІВ

1.1. Фреймворк Bootstrap та його особливості

Якщо порівнювати динамічну бібліотеку (DLL), яка відрізняється досить обмеженим функціоналом, і фреймворк, що вважається основою програм, можна виділити суттєву перевагу фреймворків. Саме фреймворк є зв'язуючою ланкою, яка об'єднує всі використовувані програмні компоненти. Також всередині фреймворка часто є необхідні тематичні бібліотеки.

Можна використовувати таку класифікацію фреймворків [7]:

– Фреймворки застосунків;

– Фреймворки програмних моделей;

– Фреймворки концептуальних моделей.

Фреймворк Bootstrap, що розглядається, який використовується для розробки сучасних веб-проектів, належить другому пункту в наведеній класифікації.

Bootstrap – це відкритий і безкоштовний HTML, CSS і JS фреймворк, який використовується веб-розробниками для швидкої верстки адаптивних дизайнів сайтів та веб-застосунків. Фреймворк Bootstrap використовується по всьому світу не тільки незалежними розробниками, але іноді й цілими компаніями. На Bootstrap створено дуже багато різних сайтів, подивитися які можна на сторінці Bootstrap Expo [8]. Основна галузь його застосування – це розробка фронтенда сайтів і інтерфейсів адміністративних частин сайтів.

Фреймворк Bootstrap почав розроблятися як внутрішня бібліотека компанії Twitter під назвою Twitter Blueprint. Після кількох місяців розробки він був відкритий під назвою Bootstrap 19 серпня 2011 року [9]. Актуальна версія на сьогоднішній день 4.5.3.

Серед переваг фреймворка Bootstrap можна відзначити, що це не тільки CSS, але і JS-фреймворк. Тобто в Bootstrap написані готові стилі і скрипти, для застосування яких достатньо всього лише прописати необхідні стильові класи і атрибути html-елементів.

В першу чергу за допомогою фреймворку Bootstrap слід розробляти мобільні проекти завдяки сітці Bootstrap, яка дозволяє легко адаптувати будь-який сайт і гарно відображати його на будь-яких пристроях.

Далі, фреймворк Bootstrap бере на себе кроссбраузерность і адаптивність, а це основні речі, про які повинен подбати розробник. Але з Bootstrap реалізувати їх дуже просто. Це дозволяє створити html-шаблон навіть людині, яка раніше дуже мало займалася верстанням і особливо не знайома з CSS.

Фреймворк Bootstrap ідеально підходить при роботі в команді. Верстання на Bootstrap при належному вмінні і розумінні відбувається в 3 – 5 разів швидше, а однаковість коду дозволить будь-якому члену команди вносити правки.

Як перевагу фреймворка Bootstrap можна також відзначити дуже велике російськомовне співтовариство і наявність гарної документації російською мовою. Завдяки такій поширеності для Bootstrap з'явилося багато шаблонів, де вже перероблений дизайн всіх основних елементів.

Серед недоліків Bootstrap можна відзначити два. Перший – коду зазвичай в бібліотеці написано більше, ніж якщо б розробка велася з нуля. У Bootstrap є все на всі випадки життя. Навіть те, що може не знадобитися. Але ця проблема легко вирішується тим, що можна вибирати, які компоненти фреймворка завантажити в css-файл.

Другий недолік – шаблонний дизайн. Але і ця проблема легко вирішується, тому що вона буде існувати тільки в тому випадку, якщо використовувати тільки готові компоненти фреймворка і нічого ніколи не налаштовувати під свої потреби. А якщо, наприклад, підключити тільки сітку

Bootstrap, то можна відтворити будь-який дизайн, при цьому користуючись дуже зручною гнучкою сіткою фреймворку.

1.2 Існуючі моделі для оцінювання розміру програмного забезпечення веб-застосунків

Перш ніж говорити про існуючі моделі оцінювання розміру ПЗ веб-застосунків, треба сказати декілька слів про метрики ПЗ.

Метрика ПЗ – міра, що дозволяє отримати чисельне значення деякої властивості ПЗ або його специфікацій.

Оскільки кількісні методи добре зарекомендували себе в інших областях, багато теоретиків і практиків намагалися перенести даний підхід і в розробку ПЗ.

Оцінювання розміру проекту, особливо на ранньому етапі розробки, відіграє важливу роль у практичних завданнях з його управління, розробки та впровадження. Як правило, кількість рядків коду програми є негаусівською ВВ, яка залежить від ряду факторів – метрик програмного забезпечення – які в тому чи іншому випадку впливають на кінцевий результат.

Перш за все, слід розглянути кількісні характеристики вихідного коду програм. Найелементарнішій метрикою є кількість рядків коду. Дана метрика була спочатку розроблена для оцінки трудовитрат за проектом. Однак через те, що одна і та ж функціональність може бути розбита на кілька рядків або записана в один рядок, метрика стала практично непридатною з появою мов, в яких в один рядок може бути записано більше однієї команди. Тому розрізняють логічні і фізичні рядки коду. Логічні рядки коду – це кількість команд програми. Даний варіант опису так само має свої недоліки, тому що сильно залежить від використовуваної мови і стилю програмування. Ще одним типом метрик ПЗ, що відносяться до кількісних, є метрики Джілбі. Вони показують складність ПЗ на основі насиченості програми умовними

операторами або операторами циклу. Дана метрика, не дивлячись на свою простоту, досить добре відображає складність написання і розуміння програми, а при додаванні такого показника, як максимальний рівень вкладеності умовних і циклічних операторів, ефективність даної метрики значно зростає.

Крім показників оцінки обсягу робіт за проектом дуже важливими для отримання об'єктивних оцінок по проекту є показники оцінки його складності. Як правило, дані показники не можуть бути обчислені на самих ранніх стадіях роботи над проектом, оскільки вимагають, як мінімум, детального проектування. Однак ці показники дуже важливі для отримання прогнозних оцінок тривалості і вартості проекту, оскільки безпосередньо визначають його трудомісткість.

В [10] термін «метрика програмного забезпечення» визначається як показник якості ПЗ, і тлумачиться таким чином: метрика якості програмного забезпечення – це функція, яка відображає програмний блок в числовому значенні.

Метрики програмного продукту враховують його характеристики. Розрізняють динамічні і статичні метрики для зручності використання.

Динамічні метрики програмного забезпечення використовуються для оцінки продуктивності та надійності ПЗ. Прикладами є необхідний час виконання або кількість помилок, що виникли під час виконання.

Статичні метрики програмного забезпечення включають показники дизайну, самої програми або документації. Вони також враховують такі аспекти, як складність і простота обслуговування.

Відмінності між процедурною і об'єктно-орієнтованою розробкою ПЗ враховуються шляхом подальшої диференціації в звичайні та об'єктно-орієнтовані метрики продукту.

У сфері розробки ПЗ управління проектом набуває особливої важливості, оскільки велика частина вартості розробки складається, як правило, з безпосередніх трудовитрат виконавців. Також нові системи

можуть мати високу складність, а вихідні вимоги можуть змінюватися на різних етапах життєвого циклу. Ці фактори ускладнюють планування і підвищують ризик неуспішного завершення проекту. Щоб визначити, чи реалістичні цілі проекту, а також підвищити точність планування, використовується оцінка трудомісткості розробки ПЗ. Вибір методу оцінювання повинен бути обґрунтований, в першу чергу, поставленим завданням. Для одних проектів спочатку необхідно оцінити обсяг функціональності, а потім, виходячи з отриманої оцінки, визначити терміни реалізації та обсяги робіт. В інших можлива протилежна ситуація: спочатку визначаються бюджети і часові рамки, після чого оцінюється обсяг реалізованих функцій. Також на вибір методу оцінки впливають такі чинники, як розмір проекту, стиль розробки, стадія розробки, можлива точність.

Моделі оцінювання розміру ПЗ поділяються на п'ять категорій: аналогові, регресійні, моделі на основі експертних оцінок, моделі, які базуються на функціональних точках, параметричні моделі [11].

Сьогодні розроблено безліч методів оцінки трудомісткості, деякі з них універсальні, інші придумані спеціально для управління певним портфелем проектів. З найбільш поширених методів оцінки можна виділити кілька категорій: експертна оцінка, регресійні моделі, функціональні точки, нейронні мережі, комбіновані методи.

Найбільш перспективні комбіновані методи, оскільки при суміщенні підходів може виникнути принципово нове рішення, що дозволяє отримати більш високу точність або розширити сферу застосування.

Оскільки для різних категорій методів оцінювання існують принципово різні способи, беручи до уваги фактор розміру проекту, виникає потреба у виборі метрики розміру проекту. Серед інших чинників, що впливають на оцінку, розмір проекту є найбільш важливим показником. Хоча оцінки розміру недостатньо для розуміння усього проекту, що розробляється, існує явна залежність між розміром проекту і його трудомісткістю. Щоб

визначити, за якими показниками можна порівнювати розмір ПЗ, необхідно розглянути основні групи метрик.

Кількість рядків коду (Source lines of code) – це розмірно орієнтовна метрика ПЗ, в якій обсяг ПЗ розраховується, виходячи з кількості рядків в тексті вихідного коду. Серед методик підрахунку кількості рядків коду існує дві основні [12]:

– по числу фізичних рядків (LOC) – визначається як загальне число рядків вихідного коду, включаючи коментарі і порожні рядки;

– по числу логічних рядків коду (LLOC) – визначається як загальна кількість команд і залежить від використовуваної мови програмування. Якщо мова підтримує розміщення кількох команд в одному рядку, то один фізичний рядок повинен бути врахованій як кілька логічних, якщо він містить більше однієї команди.

Метрики Холстеда, засновані на аналізі числа рядків і синтаксичних елементів вихідного коду програми, були запропоновані М. Холстедом (MauriceHalstead) в 1977 р. Метрики Холстеда (Halstead complexity measures) частково дозволяють врахувати можливість реалізації однієї і тієї ж функціональності різним числом рядків і операторів. Найбільш частим сценарієм використання цих метрик є оцінка складності проміжних продуктів розробки, проте набір метрик також містить оцінки розміру.

Аналіз функціональних точок (Function points) – це метод вимірювання розміру ПЗ з точки зору користувачів системи. Метод був розроблений Аланом Альбрехтом (Alan Albrecht) в середині 1970-х років, вперше опублікований в 1979 р Широке поширення ця методика отримала в середині 1980-х років, після того як була сформована організація IFPUG, що займається розвитком методу, а також публікує нові версії. На поточний момент актуальна версія методу 4.3. Даний метод призначений для оцінювання обсягу програмного продукту по функціональній моделі, тобто оцінюється обсяг функцій, що розробляється.

Метод UCP (Use Case Points) являє собою оцінку розміру проектів на основі діаграм UML (Unified Modeling Language) і методології RUP (Rational Unified Process). Як і багато інших сучасних методів оцінки, UCP базується приблизно на тих же принципах, що і метод функціональних точок. Головна відмінність полягає в заміні одиниць вимірювання з функціональних точок на варіанти використання (Use Cases).

Крім того, всесвітньо відомою є модель COCOMO II – це модель регресії, заснована на кількість рядків коду (LOC). Ця процедурна модель оцінювання витрат для програмних проектів часто використовується для надійного прогнозування різних параметрів, пов'язаних з проектом, таких, як розмір, зусилля, витрати, час та якість, які необхідні для впровадження ПЗ.

При використанні точної та надійної методології для оцінювання розміру можна зробити висновок про якість ПЗ та навіть усього програмного процесу і, при необхідності, вжити подальших заходів.

В загальному вигляді регресійна модель для оцінювання розміру програмного забезпечення веб-застосунків може бути представлена рівнянням:

$$y = \bar{y} + \varepsilon_t = f(x) + \varepsilon_t, \qquad (1.1)$$

де y – залежна змінна, або результативна ознака;

$f(x)$ – функція, яка визначає вид регресійної моделі: нелінійна або лінійна;

x – незалежна змінна, або фактор;

ε_t – випадкова помилка, або збурення.

У разі нормального розподілу ЕД можливо використання лінійної регресійної моделі:

$$z_y = b_1 z_x + b_0 + \varepsilon, \tag{1.2}$$

де b_1, b_0 – коефіцієнти лінійної регресії, які знаходяться методом найменших квадратів:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - \left(\sum_{i=1}^{n} z_{xi}\right)^2}, \; b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n}, \tag{1.3}$$

ε – ВВ, розподілена за нормальним законом, що визначається як ε~N(0,1).

1.3 Перевірка адекватності регресійної моделі для оцінювання розміру програмного забезпечення веб-застосунків

Для перевірки адекватності рівняння регресії використаємо коефіцієнт детермінації R^2 [13]:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \tag{1.4}$$

де y_i – емпіричне значення y;

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

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ – середнє значення випадкової величини y.

R^2 характеризує частку дисперсії, яка обумовлена регресією, в загальній дисперсії показника *y*. Коефіцієнт детермінації R^2 приймає значення від 0 до 1. Чим ближче значення коефіцієнта до 1, тим тісніше зв'язок результативної ознаки з досліджуваними факторами.

Для перевірки якості знайденого рівняння регресії також використовуються середня величина відносної похибки *MMRE* (Mean of Magnitude of Relative Errors) та рівень прогнозування *PRED*(0,25), які можна розрахувати за наступними формулами:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \ , \tag{1.5}$$

де *n* – кількість значень у вибірці;

$MRE_i = \left|\frac{y_i - \hat{y}_i}{y_i}\right|$ – величина відносної похибки (Magnitude of Relative Errors);

$$PRED(0{,}25) = \frac{k}{n}, \tag{1.6}$$

де *k* – кількість значень з $MRE \leq 0{,}25$.

Ці параметри також використовуються для порівняння різних регресійних моделей.

1.4 Способи удосконалення регресійної моделі для оцінювання розміру програмного забезпечення веб-застосунків

Вихідні дані для оцінювання розміру ПЗ веб-застосунків, як правило, не підпорядковуються нормальному розподілу, що в результаті призводить

до помилок у розрахунках та негативно впливає на достовірність отриманих результатів, у нашому випадку, на оцінювання розміру ПЗ веб-застосунків. Щоб уникнути цієї проблеми, перед тим, як будувати регресійну модель для оцінювання розміру ПЗ веб-застосунків, потрібно нормалізувати вихідні ЕД.

Для нормалізації негаусівських даних можуть бути використані перетворення на основі десяткового або натурального логарифму, перетворення Бокса-Кокса, перетворення Джонсона та інші. У даній роботі в якості нормалізуючого перетворення буде використовуватись нормалізуюче перетворення на основі десяткового логарифму тому, що воно найпростіше для розрахунку.

Після нормалізації вихідних ЕД можна побудувати лінійне рівняння регресії згідно з (1.2). Далі для лінійного рівняння регресії будують довірчий інтервал та інтервал прогнозування традиційним способом із використанням *t*-розподілу Стьюдента:

$$y = \hat{y} \pm t(_{\alpha/2,n-2}) \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}, \tag{1.7}$$

$$y = \hat{y} \pm t_{(\alpha/2,n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}, \tag{1.8}$$

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

$t(_{\alpha/2,n-2})$ – квантіль *t*-розподілу Стьюдента;

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

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

І, нарешті, шляхом застосування відповідного зворотнього нормалізуючого перетворення перейти до нелінійної регресійної моделі вихідних ЕД.

Тим самим отримаємо більш точну регресійну модель для оцінювання розміру програмного забезпечення веб-застосунків та підвищимо достовірність оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

2 УДОСКОНАЛЕННЯ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ BOOTSTRAP

2.1 Розробка удосконаленої регресійної моделі для оцінювання розміру веб-застосунків із застосуванням нормалізуючих перетворень

У випадку негаусівської ВВ побудова рівняння регресії без припущення про нормальність ВВ може бути ускладненою. Використання такого припущення може істотно спотворити результати. Тому перед тим, як будувати рівняння регресії, потрібно нормалізувати вихідні ЕД.

Використання нормалізуючих перетворень та отримання гаусівської ВВ дозволяє перейти до лінійного рівняння регресії, для нього побудувати довірчий інтервал та інтервал прогнозування традиційним способом, і, нарешті, шляхом застосування зворотного перетворення перейти до нелінійного рівняння регресії і його інтервалів [14].

Для нормалізації вихідних ЕД будемо використовувати нормалізуюче перетворення на основі десяткового логарифму:

$$z = \log_{10} x; \tag{2.1}$$

де z – нормована нормально розподілена ВВ;

x – ВВ, яка нормалізується.

Перетворення (2.1) має зворотне перетворення:

$$x = 10^{z}. \tag{2.2}$$

Перевірку відповідності перетворених вибірок нормальному розподілу можна виконати за допомогою критеріїв згоди, наприклад, хі-квадрат Пірсона (χ^2) або Колмогорова-Смирнова [15]. Будемо використовувати критерій χ^2 Пірсона. Він застосовується для зіставлення емпіричного розподілу з теоретичним. Розподіл χ^2 є одним з найбільш широко використовуваних в статистиці для перевірки статистичних гіпотез. На основі розподілу χ^2 побудований один з найбільш потужних критеріїв згоди – критерій Пірсона. Розрахункова формула критерію наступна [16]:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \qquad (2.3)$$

де m – кількість підінтервалів, залежна від обсягу вибірки n;

n_i – кількість значень випадкової величини, що потрапили в i-й підінтервал;

p_i – ймовірність потрапляння випадкової величини в i-й підінтервал відповідно з гіпотетичним законом розподілу випадкової величини.

Кількість підінтервалів m можна розрахувати за наступними формулами [17]:

Старджеса:

$$m = 3{,}31 \cdot \lg n + 1,$$

або Брукса і Карузера:

$$m = 5 \cdot \lg n .$$

Отримавши нормалізований набір даних за допомогою нормалізуючого перетворення на основі десяткового логарифму за формулою (2.1) та

перевіривши нормальність розподілу з допомогою критерія згоди Пірсона χ^2 за формулою (2.3), можна переходити до наступного кроку – побудови лінійної регресійної моделі.

Загальний вигляд лінійної регресійної моделі може бути представлений у вигляді рівняння (1.2). Адекватність отриманого рівняння перевіримо за допомогою коефіцієнта детермінації R^2, формула (1.4), та середньої величини відносної похибки *MMRE* і рівня прогнозування *PRED*(0,25) за формулами (1.5) та (1.6) відповідно.

Далі для лінійного рівняння регресії будуємо довірчий інтервал згідно з (1.7) та інтервал прогнозування згідно з (1.8).

Для побудови нелінійної регресійної моделі використаємо вже побудовану лінійну регресійну модель та зворотне нормалізуюче перетворення (2.2):

$$y = x^{b_1} \cdot 10^{b_0 + \varepsilon}. \qquad (2.4)$$

Адекватність отриманого рівняння перевіримо за допомогою коефіцієнта детермінації R^2, формула (1.4), та середньої величини відносної похибки *MMRE* і рівня прогнозування *PRED*(0,25) за формулами (1.5) та (1.6) відповідно.

(1-α)% довірчий інтервал нелінійного рівняння регресії можна побудувати, використовуючи побудований (1-α)% довірчий інтервал лінійного рівняння регресії за формулою (1.7) та зворотне нормалізуюче перетворення за формулою (2.2).

Аналогічно знайдемо і (1-α)% інтервал прогнозування нелінійного рівняння регресії, перехід до якого можна здійснити, використовуючи побудований (1-α)% інтервал прогнозування лінійного рівняння регресії за формулою (1.8) та зворотне нормалізуючи перетворення за формулою (2.2).

Таким чином, розроблена удосконалена регресійна модель для оцінювання розміру веб-застосунків із застосуванням нормалізуючого перетворення на основі десяткового логарифму.

2.2 Побудова регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap

Для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, однією з основних задач є побудова відповідної математичної моделі, у нашому випадку, це буде нелінійне рівняння регресії [18 – 20]. Також побудувавши довірчий інтервал нелінійної регресії, можна підвищити надійність оцінювання [21, 22].

На інформаційному ресурсі GitHub було знайдено 35 веб-застосунків з відкритим вихідним кодом, розроблених з використанням фреймворку Bootstrap. Кожен з застосунків було ретельно проаналізовано. Було виконано роботу зі збору набору метрик з цих веб-застосунків, у результаті чого було отримано такий набір даних:

– LOC – число рядків коду програми;

– NC – загальна кількість компонентів;

– CPL – кількість плагінів;

– CP – кількість сторінок.

Але побудова багатофакторної нелінійної регресійної моделі на основі декількох метрик може бути достатньо складною та затратною в плані зусиль та часу, необхідних для її реалізації, для людей, які до цього не мали досвіду у цій галузі. Тому, на основі отриманих ЕД та аналізу існуючих моделей оцінювання розміру ПЗ [23], було прийнято рішення побудувати однофакторну нелінійну регресійну модель для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, на основі

таких метрик: кількості рядків коду LOC та загальної кількості компонентів NC, використовуючи нормалізуюче перетворення на основі десяткового логарифму.

Перше, що потрібно зробити – перевірити отримані дані на викиди. Нормальні дані не повинні мати викидів, отже, якщо вони є, то це може бути причиною того, що дані не розподіляються за нормальним законом. Якщо викиди будуть знайдені, потрібно видалити їх з аналізу та провести повторну перевірку. Лише після цього можна переходити до наступного кроку аналізу даних [24, 25].

Для перевірки даних на викиди будемо використовувати квадрат відстані Махаланобіса (Mahalanobis squared distance), який визначається як [26]:

$$d_i^2 = \left(Z_i - \bar{Z}\right)^T S_N{}^{-1} \left(Z_i - \bar{Z}\right), \tag{2.5}$$

де $\bar{Z}$ – середній вектор вибірки,

S_N – матриця кореляції вибірки:

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

Розрахувавши d_i^2 можна використати або критерій Пірсона χ^2, або критерій Фишера F. При використанні критерію Фишера F потрібно додатково розрахувати T_S (Test statistic). T_S для d_i^2 може бути розрахований таким чином:

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}, \tag{2.6}$$

який має апроксимований розподіл F з m та $N - m$ ступенями свободи.

У данній роботі будемо використовувати критерій Фишера F. Тестова статистика для квадрату відстані Махаланобіса порівнюється з квантилем розподілу F, який позначається як $F_{m,N-m,\alpha}$. Тут α – це рівень значущості, який у нашому випадку дорівнюватиме 0,05. Точки, для яких значення T_S, розраховане за формулою (2.6) буде більше, ніж квантиль розподілу F вважаються викидами, і ці значення видаляються з набору даних.

Після усунення викидів отриманий новий набір значень знову нормалізується, використовуючи перетворення (2.1) та для нього знову знаходиться квадрат відстані Махаланобіса. Процедура повторюється до того моменту, поки всі значення T_S не будуть менше або дорівнюватимуть квантилю розподілу F.

Проведемо перевірку на виявлення викидів серед вихідних ЕД. У таблиці 2.1 наведено значення вихідних ЕД, нормалізовані значення та значення T_S для d_i^2 на першій ітерації, для якої F=3,285, m=2, α=0,05.

Таблиця 2.1 – Значення тестової статистики вихідних ЕД для першої ітерації

№ п/п	X, NC	Y, LOC	Z_X	Z_Y	T_S для d_i^2
1	2	3	4	5	6
1	173	13634	2,2380	4,1346	0,1578
2	696	122011	2,8426	5,0864	3,2127
3	31	1940	1,4914	3,2878	1,2203
4	290	76817	2,4624	4,8855	2,2340
5	49	2496	1,6902	3,3972	0,8017
6	24	6069	1,3802	3,7831	2,1331
7	30	1898	1,4771	3,2783	1,2659
8	37	3774	1,5682	3,5768	0,7020
9	93	27485	1,9685	4,4391	1,4614
10	170	8762	2,2304	3,9426	0,3604
11	169	7371	2,2279	3,8675	0,5414
12	63	10350	1,7993	4,0149	0,4829
13	82	11973	1,9138	4,0782	0,2696
14	131	1920	2,1173	3,2833	3,1395
15	223	6858	2,3483	3,8362	1,2482
16	175	9093	2,2430	3,9587	0,3674
17	202	15929	2,3054	4,2022	0,2702
18	196	15525	2,2923	4,1910	0,2457

Продовження табл. 2.1

1	2	3	4	5	6
19	68	6926	1,8325	3,8405	0,1314
20	41	5114	1,6128	3,7088	0,6203
21	211	16849	2,3243	4,2266	0,3076
22	80	18598	1,9031	4,2695	0,9366
23	**963**	**14809**	**2,9836**	**4,1705**	**5,1759**
24	59	5469	1,7709	3,7379	0,2165
25	72	7159	1,8573	3,8549	0,0988
26	75	8971	1,8751	3,9528	0,1449
27	104	10359	2,0170	4,0153	0,0166
28	66	8068	1,8195	3,9068	0,2131
29	68	3948	1,8325	3,5964	0,3482
30	176	15753	2,2455	4,1974	0,1769
31	309	80883	2,4900	4,9079	2,3114
32	286	28947	2,4564	4,4616	0,7198
33	70	5736	1,8451	3,7586	0,1229
34	62	4043	1,7924	3,6067	0,3181
35	36	2099	1,5563	3,3220	1,0536

Як видно з таблиці 2.1, на першій ітерації викидом є 23 набір даних. Видаляємо його з вибірки та повторюємо процедуру перевірки на викиди. У результаті за 2 ітерації було видалено 3 набори даних. Після видалення викидів скоригована вибірка вихідних ЕД містить 32 проекти.

Отримані ЕД для вибірок *X* та *Y* не відповідають нормальному закону розподілу: χ^2_X=72,63 та χ^2_Y=60,04 при критичному значенні χ^2=5,99 для довірчої ймовірності 0,95.

Гістограма емпіричного розподілу для вибірки *X* зображена на рис. 2.1, гістограма емпіричного розподілу для вибірки *Y* зображена на рис. 2.2.

Нормалізація ЕД була виконана з використанням нормалізуючого перетворення на основі десяткового логарифму за формулою (2.1). Отримані нормалізовані вибірки *X* та *Y* відповідають нормальному закону розподілу: χ^2_{Zx}=3,23 та χ^2_{Zy}=1,62 при критичному значенні χ^2=5,99 для довірчої ймовірності 0,95.

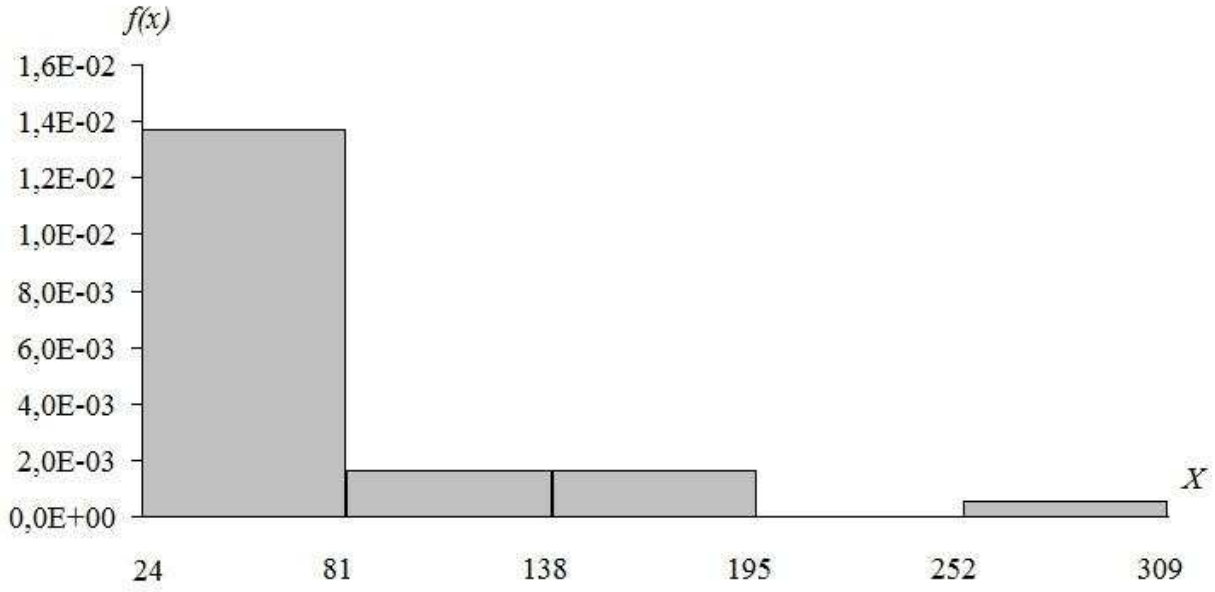

Рисунок 2.1 – Гістограма емпіричного розподілу для вибірки X

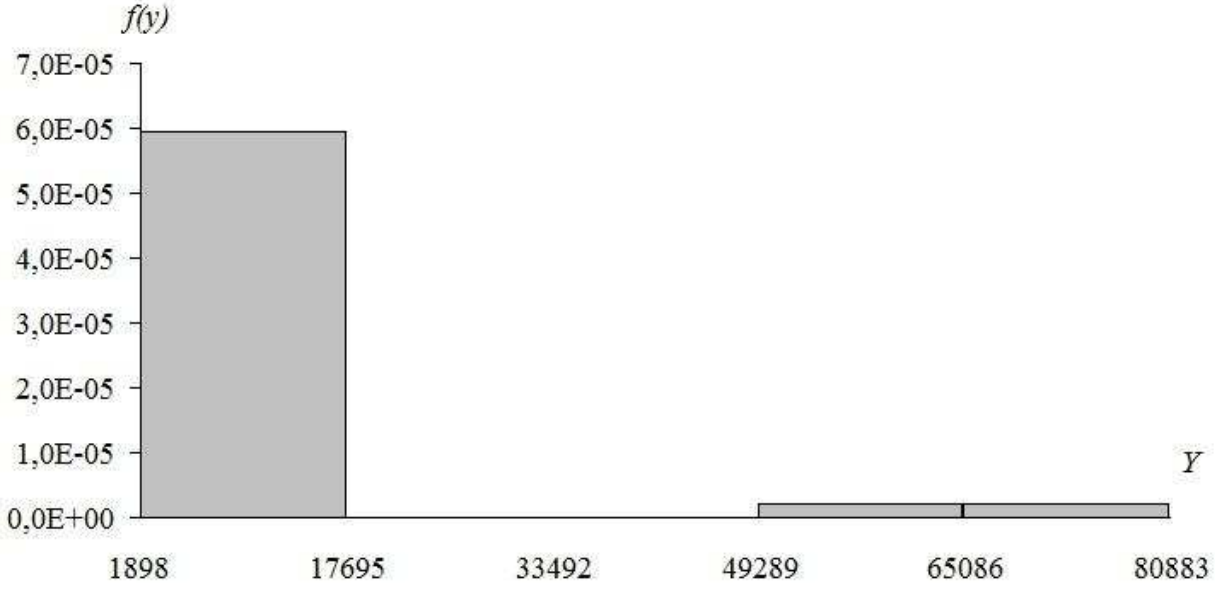

Рисунок 2.2 – Гістограма емпіричного розподілу для вибірки Y

Гістограма розподілу нормалізованих значень для вибірки Z_X зображена на рис. 2.3, гістограма розподілу нормалізованих значень для вибірки Z_Y зображена на рис. 2.4.

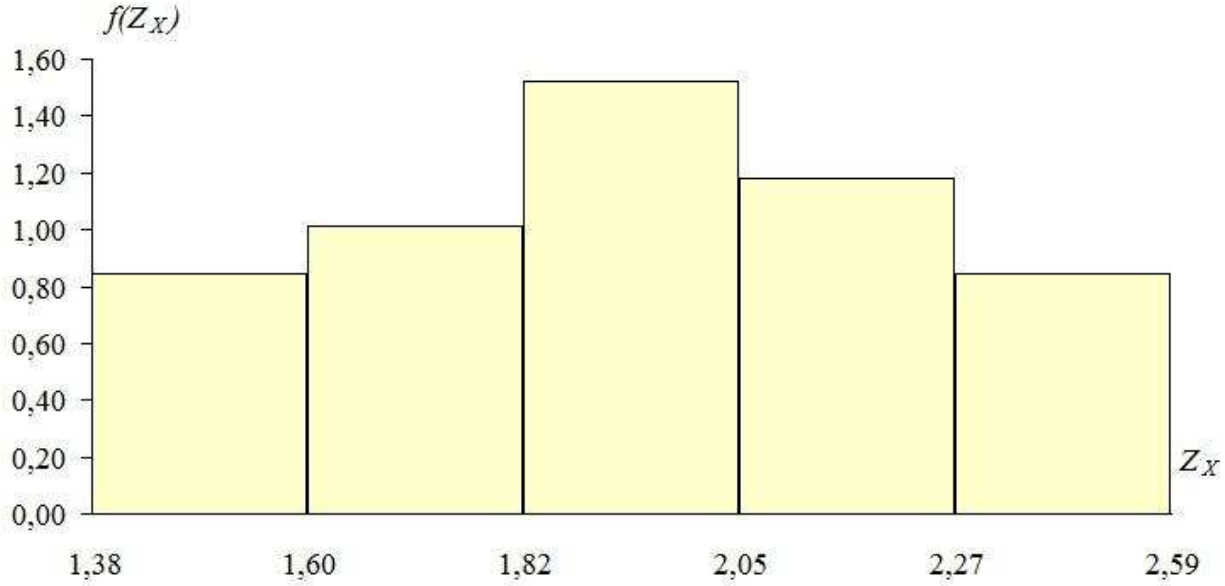

Рисунок 2.3 – Гістограма розподілу нормалізованих значень для вибірки Z_X

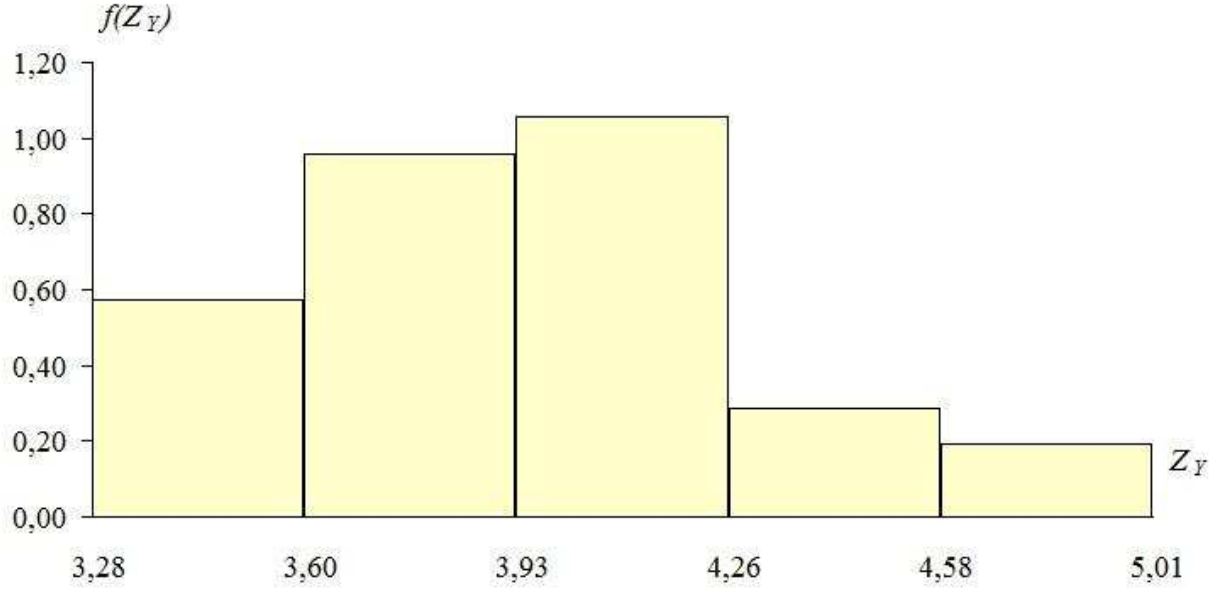

Рисунок 2.4 – гістограма розподілу нормалізованих значень для вибірки Z_Y

Будуємо лінійне рівняння регресії для нормалізованих даних згідно з (1.2), коефіцієнти знаходимо за допомогою методу найменших квадратів за формулою (1.3): b_1=0,9784, b_0=2,0228. Лінійна регресійна модель має вигляд $Z_y = 0{,}9784 Z_x + 2{,}0228 + \varepsilon$.

Далі будуємо довірчий інтервал та інтервал прогнозування лінійного рівняння регресії згідно з (1.7) та (1.8) відповідно, які разом із самим рівнянням та нормалізованими ЕД зображені на рис. 2.5.

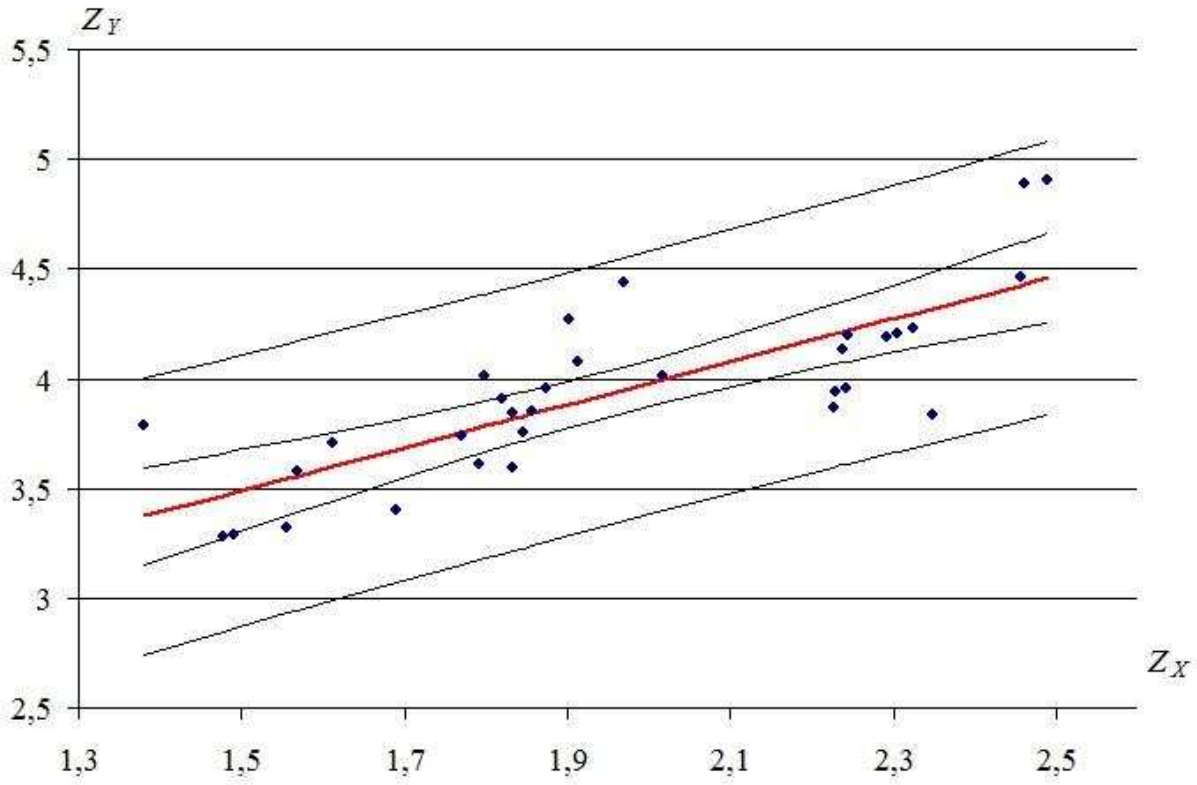

Рисунок 2.5 – Лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

Переходимо до вихідних ЕД. На основі лінійного рівняння регресії та зворотнього нормалізуючого перетворення будуємо нелінійне рівняння регресії згідно з (1.1), довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії. Нелінійна регресійна модель має вигляд $Y = X^{0,9784} \cdot 10^{2,0228+\varepsilon}$.

Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування зображені на рис. 2.6.

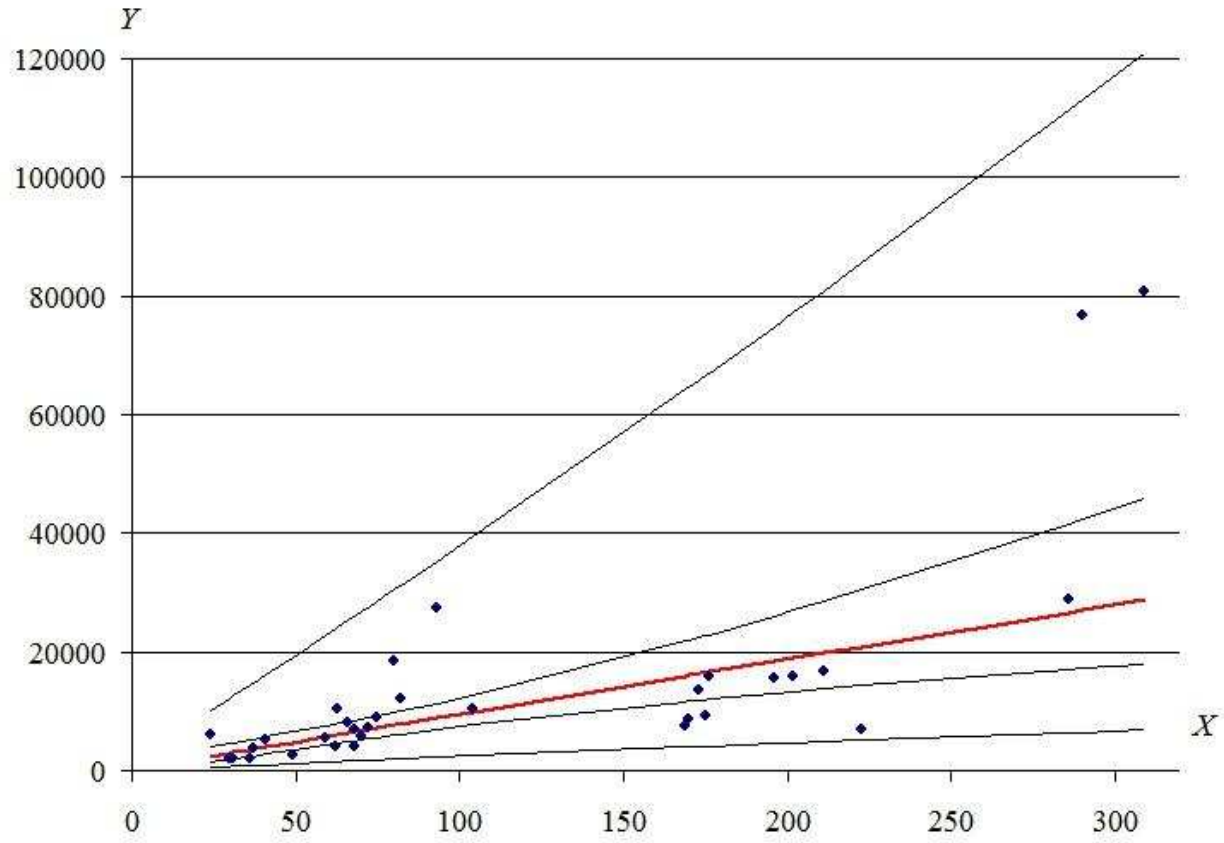

Рисунок 2.6 – Нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

Для порівняння побудуємо лінійну регресійну модель в припущенні про нормальність вихідних ЕД згідно з (1.2), коефіцієнти рівняння також знаходимо за допомогою методу найменших квадратів за формулою (1.3): b_1=157,29, b_0=-4601,01. Лінійна регресійна модель має вигляд $Z_y = 157{,}29 Z_x - 4601{,}01 + \varepsilon$.

Далі будуємо порівняльні довірчий інтервал та інтервал прогнозування лінійного рівняння регресії згідно з (1.7) та (1.8) відповідно, які разом із самим рівнянням та вихідними ЕД зображені на рис. 2.7.

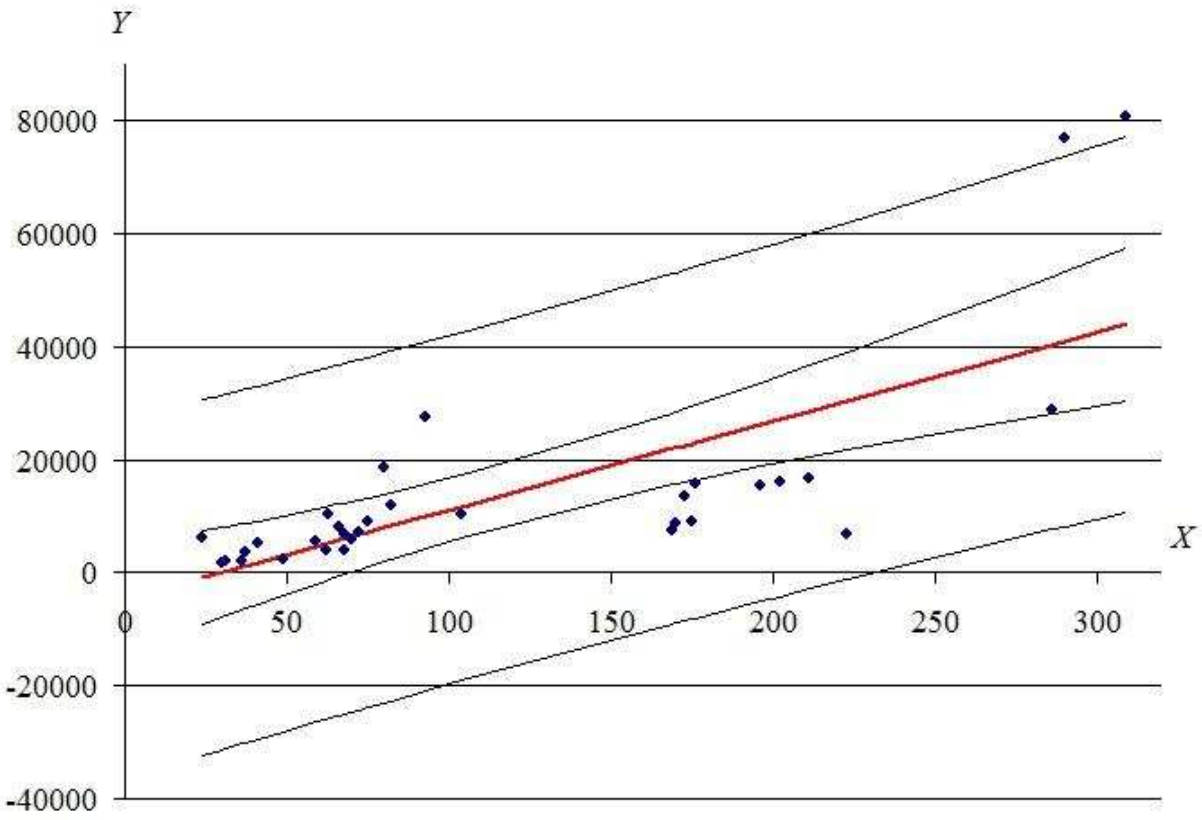

Рисунок 2.7 – Порівняльне лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

В даному випадку використовувати лінійну регресійну модель без виконання нормалізації не можна, оскільки вихідні емпіричні дані не підпорядковуються нормальному закону розподілу, на рис. 2.7 явно видно два викиди. До того ж, і лінійне рівняння регресії, і довірчий інтервал, і інтервал прогнозування мають від`ємні значення.

Перевіримо адекватність побудованих регресійних моделей за значеннями коефіцієнту детермінації R^2, *MMRE* та *Pred*(0,25), які були розраховані за формулами (1.4), (1.5) та (1.6) відповідно. Зведені дані наведено в табл. 2.2.

Як видно із наведених даних, рівняння, побудоване із використанням нормалізуючого перетворення на основі десяткового логарифму, має кращі параметри. Отже, отримане нелінійне рівняння регресії, побудоване з використанням нормалізуючого перетворення на основі десяткового

логарифму можна вважати прийнятним, хоча значення параметрів *MMRE* та *Pred*(0,25) гірші за допустимі (не більше 0,25 для *MMRE* та не менше 0,75 для *Pred*(0,25)). Для отримання кращого результату треба спробувати інші нормалізуючі перетворення, наприклад, нормалізуючи перетворення Джонсона.

Таблиця 2.2 – Перевірка адекватності регресійних моделей

Параметр	Нелінійна регресійна модель з використанням нормалізуючого перетворення	Лінійна регресійна модель без виконання нормалізації
R^2	0,6054	0,5195
MMRE	0,2533	0,6834
Pred(0,25)	0,6688	0,2188

Порівняємо отримані значення для довірчого інтервалу та інтервалу прогнозування нелінійного рівняння регресії із аналогічними результатами, які отримані без виконання нормалізації (в припущенні про нормальність вихідних емпіричних даних). Зведені результати для довірчого інтервалу наведено у табл. 2.3, для інтервалу прогнозування – у табл. 2.4.

Таблиця 2.3 – Довірчий інтервал рівняння регресії

№ про-екта	з використанням нормалізуючого перетворення			без виконання нормалізації		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
1	2	3	4	5	6	7
1	11863,72	22422,68	10558,96	16168,70	29051,55	12882,85
2	1965,06	4682,22	2717,16	-7555,70	8105,66	15661,36
3	17241,32	42397,52	25156,20	28608,91	53417,14	24808,23
4	3451,50	6529,65	3078,14	-3917,06	10129,45	14046,51
5	1422,13	3920,97	2498,85	-8994,71	7342,60	16337,31
6	1885,76	4575,89	2690,13	-7760,55	7995,92	15756,47
7	2450,27	5308,49	2858,22	-6332,17	8769,60	15101,78
8	6989,68	11297,45	4307,77	4405,94	15647,94	11242,00
9	11706,60	21958,96	10252,36	15802,66	28473,84	12671,18

Продовження табл. 2.3

1	2	3	4	5	6	7
10	11653,92	21805,04	10151,13	15679,70	28282,23	12602,53
11	4628,37	7962,31	3333,94	-1164,83	11781,33	12946,16
12	6160,62	10019,35	3858,73	2424,23	14169,27	11745,04
13	14312,99	30544,13	16231,14	21797,93	39151,29	17353,36
14	11967,71	22733,44	10765,73	16410,42	29438,99	13028,57
15	13319,07	27047,95	13728,88	19522,66	34820,39	15297,73
16	13026,55	26070,81	13044,26	18852,22	33603,36	14751,14
17	5041,80	8487,58	3445,77	-202,31	12391,71	12594,02
18	2780,75	5718,20	2937,45	-5522,12	9217,86	14739,98
19	13750,47	28532,14	14781,67	20510,15	36664,12	16153,97
20	6004,90	9794,31	3789,41	2055,75	13908,60	11852,85
21	4294,04	7548,42	3254,39	-1943,02	11301,20	13244,23
22	5367,91	8915,34	3547,43	558,92	12888,80	12329,88
23	5609,31	9241,12	3631,81	1124,38	13267,08	12142,70
24	7771,83	12644,93	4873,11	6304,36	17209,90	10905,53
25	4877,13	8276,30	3399,17	-585,90	12146,14	12732,04
26	5041,80	8487,58	3445,77	-202,31	12391,71	12594,02
27	12019,49	22889,31	10869,82	16530,61	29633,38	13102,77
28	18019,34	45930,51	27911,17	30470,63	57532,43	27061,80
29	17074,95	41662,79	24587,84	28214,21	52553,52	24339,31
30	5205,43	8700,56	3495,13	179,32	12639,24	12459,92
31	4545,04	7858,36	3313,32	-1358,72	11660,64	13019,36
32	2368,42	5205,26	2836,85	-6535,41	8658,26	15193,68

Таблиця 2.4 – Інтервал прогнозування нелінійного рівняння регресії

№ про-екта	з використанням нормалізуючого перетворення			без виконання нормалізації		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
1	2	3	4	5	6	7
1	4043,18	65793,84	61750,66	-8448,00	53668,24	62116,24
2	729,06	12620,16	11891,10	-31100,72	31650,68	62751,40
3	6466,73	113038,38	106571,65	8195,70	73830,35	65634,65
4	1176,71	19152,58	17975,87	-28077,79	34290,18	62367,97
5	554,17	10062,16	9508,00	-32287,80	30635,70	62923,50

Продовження табл. 2.4

1	2	3	4	5	6	7
6	704,07	12255,99	11551,92	-31269,91	31505,29	62775,20
7	878,85	14800,22	13921,37	-30088,33	32525,76	62614,08
8	2237,83	35286,74	33048,91	-20871,45	40925,33	61796,78
9	3977,98	64621,87	60643,89	-8898,09	53174,60	62072,69
10	3956,21	64231,69	60275,47	-9048,39	53010,32	62058,71
11	1519,97	24245,54	22725,57	-25756,45	36372,95	62129,40
12	1977,44	31214,78	29237,34	-22648,38	39241,89	61890,27
13	5105,55	85627,91	80522,35	-1122,85	62072,07	63194,93
14	4086,55	66576,32	62489,78	-8148,61	53998,02	62146,63
15	4664,71	77229,62	72564,91	-4159,29	58502,34	62661,63
16	4537,37	74847,96	70310,59	-5037,43	57493,01	62530,44
17	1641,31	26072,27	24430,96	-24933,79	37123,19	62056,98
18	978,45	16251,07	15272,62	-29416,03	33111,78	62527,81
19	4854,55	80817,09	75962,54	-2850,94	60025,21	62876,15
20	1929,75	30477,47	28547,72	-22973,24	38937,59	61910,82
21	1422,42	22787,44	21365,02	-26417,01	35775,19	62192,20
22	1737,89	27537,24	25799,35	-24278,10	37725,81	62003,92
23	1810,04	28638,18	26828,13	-23787,76	38179,21	61966,97
24	2495,03	39387,99	36892,96	-19111,10	42625,36	61736,46
25	1592,86	25341,00	23748,15	-25262,45	36822,69	62085,14
26	1641,31	26072,27	24430,96	-24933,79	37123,19	62056,98
27	4108,20	66967,91	62859,71	-7999,12	54163,10	62162,22
28	6841,54	120972,41	114130,87	10741,94	77261,12	66519,18
29	6387,24	111376,82	104989,58	7654,44	73113,29	65458,86
30	1689,66	26804,34	25114,69	-24605,68	37424,23	62029,91
31	1495,62	23880,75	22385,13	-25921,39	36223,31	62144,70
32	853,91	14437,30	13583,39	-30256,73	32379,58	62636,31

Як бачимо із наведених таблиць, нижні границі довірчого інтервалу та інтервалу прогнозування, отримані із використанням нормалізуючого перетворення Джонсона, більші нуля, тоді як ті ж самі границі, отримані без використання нормалізації, мають від`ємні значення.

До того ж, довжини обох інтервалів, отримані без використання нормалізації, зазвичай більші відповідних довжин, отриманих із використанням нормалізуючого перетворення.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ BOOTSTRAP

3.1 Ескізний проект програмного забезпечення

3.1.1 Вибір мови моделювання

Універсальна мова моделювання (Unified Modelling Language або UML) – це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG) [27]. Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему. У UML реалізовано підтримку таких типів діаграм:

1. Діаграма випадків використання показує дієвих осіб (людей або інших користувачів системи), випадки використання (сценарії використання системи) та їх взаємодію.

2. Діаграми класів, на яких буде показано класи та зв’язки між ними.

3. Діаграми послідовності, на яких показано об’єкти і послідовність методів, якими ці об’єкти викликають інші об’єкти.

4. Діаграми співпраці, на яких буде показано об’єкти та їх взаємозв’язок з наголосом на об’єкти, які беруть участь у обміні повідомленнями.

5. Діаграми стану, на яких буде показано стани, зміну станів і події у об’єкті або частині системи.

6. Діаграми діяльності, на яких буде показано дії та зміни однієї дії іншою, які є наслідком подій, що сталися у певній частині системи.

7. Діаграми компонентів, на яких буде показано програмні компоненти високого рівня.

8. Діаграми впровадження, на яких буде показано екземпляри компонентів та їх взаємодію.

9. Діаграми взаємозв’язку сутностей, на яких буде показано дані, взаємозв’язки і умови обмеження зв’язків між даними.

3.1.2 Побудова діаграма варіантів використання

Для того, щоб більш точно зрозуміти, як повинна працювати система, все частіше використовується опис функціональності системи через варіанти використання (Use Case чи прецеденти).

Варіанти використання це – опис послідовності дій, які може здійснювати система у відповідь на зовнішні дії користувачів або інших програмних систем. Варіанти використання відображають функціональність системи.

Діаграма варіантів використання – це граф спеціального вигляду, який є графічною нотацією для представлення конкретних варіантів використання,

акторів, можливо деяких інтерфейсів, і відносин між цими елементами. При цьому окремі компоненти діаграми можуть бути поміщені в прямокутник, який позначає проектовану систему в цілому. Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до модельованої системи.

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і керують усім процесом розробки. Всі основні види діяльності, такі як аналіз, проектування, тестування виконуються на основі варіантів використання.

Кожен варіант використання задає сценарій взаємодії системи з дійовими особами або ролями, що дає в підсумку значущий для них результат. Дійовими особами можуть бути не тільки люди, але й інші системи або підсистеми, які взаємодіють з ними.

Модель варіантів використання є основою для проектування та оцінки готовності системи до впровадження. Для фіксації вимог виконаємо побудову діаграми варіантів використання, яка представлена на рисунку 3.1.

Рисунок 3.1 – Діаграма варіантів використання

Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач впливають на

архітектуру системи і як повинні поводитися компоненти системи, для того щоб реалізувати потрібну для користувача функціональність.

3.1.3 Специфікації варіантів використання

Специфікація варіантів використання служить для опису послідовності дій користувача і системи в рамках одного варіанта використання. Розглянемо більш детально кожний варіант використання. Потік подій варіанта використання складається з:

- короткого опису;
- передумов;
- основного потоку подій;
- альтернативного потоку подій;
- постумов.

Ці складові частини для кожного варіанту використання приведені у таблиці 3.1.

Таблиця 3.1 – Опис прецедентів до програмного забезпечення

Складова	Опис
1	2
Імпорт вихідних даних	
Короткий опис	Актор імпортує дані з Excel.
Передумови	Відсутні
Основний потік подій	Актор вибирає імпорт даних, та вибирає файл який необхідно імпортувати. Система перевіряє дані на відповідність типів даних, та повідомляє про успішне або ні, завантаження даних.
Альтернативний потік подій	Відсутній.
Постумови	Система заповнює всі необхідні дані для подальших розрахунків.

Продовження табл. 3.1

1	2
Розрахунок параметрів вихідних даних	
Короткий опис	Розрахунок параметрів вихідних даних для подальшого їх використання.
Передумови	Даний варіант використання буде доступний після успішного завантаження даних.
Основний потік подій	Даний варіант використання почне виконуватись після вибору адміністратором опції «Розрахунок». Коли вибрана опція «Розрахунок» система відобразить поля з розрахунками параметрів вихідних даних.
Альтернативний потік подій	Відсутній.
Постумови	Відсутні.
Розрахунок параметрів нормалізації	
Короткий опис	Розрахунок параметрів нормалізації за допомогою нормалізуючого перетворення на основі десяткового логарифму.
Передумови	Відсутній
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Розрахунок параметрів нормалізації». Система перевірить коректність введення даних та представить розрахунки. Також буде здійснена перевірка даних на нормальній закон розподілу, у разі успішної нормалізації система повідомить про успішність виконаних результатів.
Альтернативний потік подій	Якщо дані введено не коректно, система видасть помилку та запропонує змінити раніше введені дані та зберегти їх.
Постумови	Відсутні.
Побудова лінійного рівняння регресії	
Короткий опис	Побудова лінійного рівняння регресії.
Передумови	Відсутні.
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Лінійне рівняння регресії». Після чого, система автоматично знайде коефіцієнти лінійного рівняння, та побудує рівняння регресії.
Альтернативний потік подій	Відсутній.
Постумови	Відсутні.
Побудова довірчого інтервалу для лін. рівняння	
Короткий опис	Побудова довірчого інтервалу для лін. рівняння
Передумови	Даний варіант використання буде доступний після успішної побудови лінійного рівняння регресії.
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Побудова довірчого інтервалу для лін. рівняння». Після чого, система автоматично знайде нижню та верхню границю та довжину інтервалів, та побудує довірчий інтервал для лінійного рівняння регресії.

Продовження табл. 3.1

1	2
Альтернативний потік подій	Відсутній.
Постумови	Відсутні.
Побудова інтервалу прогнозування для лін. рівняння	
Короткий опис	Побудова інтервалу прогнозування для лінійного рівняння
Передумови	Даний варіант використання буде доступний після успішної побудови лінійного рівняння регресії.
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Побудова інтервалу прогнозування для лінійного рівняння». Після чого, система автоматично знайде нижню та верхню границю та довжину інтервалів, та побудує інтервал прогнозування для лінійного рівняння регресії.
Альтернативний потік подій	Відсутній.
Постумови	Відсутні.
Побудова нелінійного рівняння регресії	
Короткий опис	Побудова нелінійного рівняння регресії
Передумови	Відсутні.
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Нелінійне рівняння регресії». Після чого, система автоматично знайде коефіцієнти нелінійного рівняння, та побудує рівняння регресії.
Альтернативний потік подій	Відсутні.
Постумови	Відсутні.
Побудова довірчого інтервалу для нелін. рівняння	
Короткий опис	Побудова довірчого інтервалу для нелінійного рівняння
Передумови	Даний варіант використання буде доступний після успішної побудови нелінійного рівняння регресії.
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Побудова довірчого інтервалу для нелінійного рівняння». Після чого, система автоматично знайде нижню та верхню границю та довжину інтервалів, та побудує довірчий інтервал для нелінійного рівняння регресії.
Альтернативний потік	Відсутній.
Постумови	Відсутні.
Побудова інтервалу прогнозування для нелінійного рівняння	
Короткий опис	Побудова інтервалу прогнозування для нелінійного рівняння
Передумови	Даний варіант використання буде доступний після успішної побудови лінійного рівняння регресії.

Продовження табл. 3.1

1	2
Основний потік подій	Даний варіант використання почне виконуватись після вибору користувачем опції «Побудова інтервалу прогнозування для нелінійного рівняння». Після чого, система автоматично знайде нижню та верхню границю та довжину інтервалів, та побудує інтервал прогнозування для лінійного рівняння регресії.
Альтернативний потік	Відсутній.
Постумови	Відсутні.
Експорт результатів у файл	
Короткий опис	Актор експортує дані у файл Excel.
Передумови	Відсутні
Основний потік подій	Актор вибирає експорт даних, та вибирає місце куди неохідно експортувати файл. Система експортує дані, та виводить повідомлення про успішність експорту даних.
Альтернативний потік	Повідомлення про помилку експорту даних.
Постумови	Відсутні.

Таким чином, вищенаведені прецеденти описують всю необхідну функціональність програмного забезпечення.

3.1.4 Побудова діаграми діяльності

Для моделювання процесу виконання операцій в UML використовуються так звані діаграми діяльностей. Їх графічна нотація в дечому схожа з нотацією діаграми станів. Відмінність полягає в семантиці станів, які використовуються для представлення не діяльностей, а дій, і у відсутності на переходах сигнатури подій. Кожен стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід до наступного стану спрацьовує тільки при завершенні цієї операції в попередньому стані. Графічно діаграма діяльності представляється в форму графу діяльності, вершинами якого є стани дії, а дугами – переходи від одного стану дії до іншого.

Таким чином, діаграми діяльності можна вважати частковим випадком діаграм станів. Основним напрямком використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їх виконання. При цьому кожен стан може бути виконанням операції деякого класу або її частини, дозволяючи використовувати діаграми діяльності для опису реакцій на внутрішні події системи.

В контексті мови UML діяльність (activity) представляє собою деяку сукупність окремих обчислень, які виконує автомат. При цьому окремі елементарні обчислення можуть приводити до деякого результату або дії (action). На діаграмі діяльності зображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увага фіксується на результаті діяльності. Сам же результат може привести до зміни стану системи або повернення деякого значення.

Таким чином, діаграми діяльності важливі не тільки для моделювання динамічних аспектів поведінки системи, а й для побудови виконуваних систем за допомогою прямого і зворотного проектування.

На основі описаних ключових прецедентів були побудовані діаграми діяльності, зображені на рисунках 3.2 – 3.3.

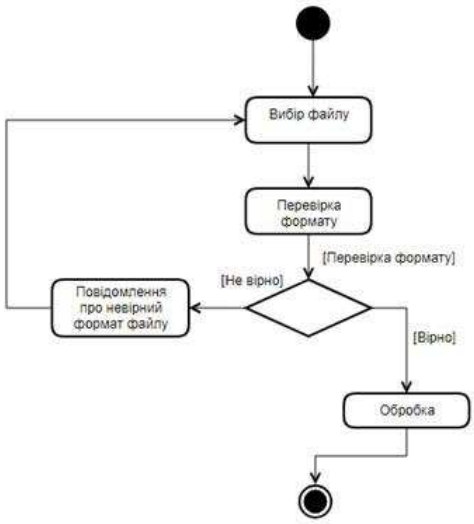

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Імпорт вихідних даних»

Рисунок 3.3 – Діаграма діяльності для варіанту використання «Розрахунок параметрів нормалізації»

Побудовані діаграми діяльності для варіантів використання добре відображають:

- послідовність дій;
- події, що ініціюють дії або є кінцевим результатом.

3.1.5 Розробка прототипу інтерфейсу користувача

Оскільки якість процесу інтерактивної взаємодії користувача із системою (швидкість, зручність, низький рівень втоми) пов’язана з такими психологічними характеристиками людини як короткострокова та середньострокова пам’ять, час реакції, можливості сприйняття візуальної інформації, то при розробці інтерфейсу необхідно він щоб задовольняв основні властивості: адаптованість, юзабіліті (зручність користування), достатність, дружність, гнучкість.

З урахуванням всіх властивостей було спроектовано ескізи інтерфейсу користувача майбутнього програмного забезпечення. Ескіз форми розрахунку вихідних даних зображений на рисунку 3.4

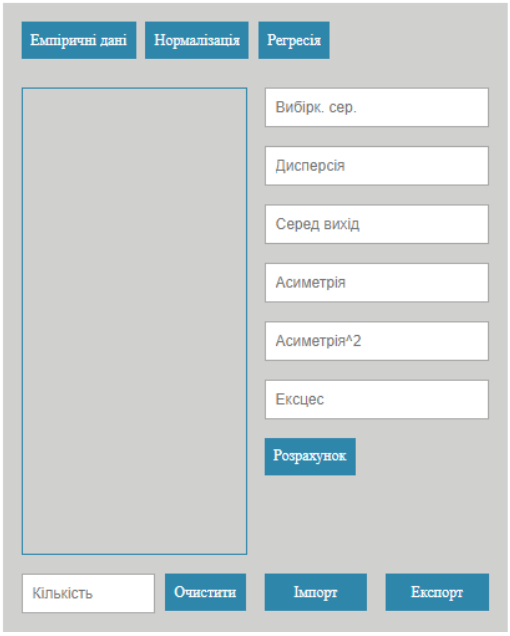

Рисунок 3.4 – Ескіз форми розрахунку вихідних даних

Розроблений прототип інтерфейсу користувача програмного забезпечення допоможе задовольнити основні потреби користувача.

3.2 Технічний проект програмного забезпечення

3.2.1 Архітектура програмного забезпечення

Архітектурним стилем було обрано архітектуру MVC. MVC – це не шаблон проекту, це конструкційний шаблон, який описує спосіб побудови структури застосування, сфери відповідальності та взаємодія кожної з частин в даній структурі. Контролер керує запитами користувача (одержувані у вигляді запитів, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його основна функція – викликати і координувати дію

необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі і вибирає відповідний вид. Модель – це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином. Модель дає контролеру уявлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних. Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з якою ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка (дуже схоже на ведення домашнього господарства). Вид забезпечує різні способи представлення даних, які отримані з моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різних видів, і контролер вибирає, який підходить якнайкраще для поточної ситуації.

Саме очевидна перевага, яку ми отримуємо від використання концепції MVC – це чіткий поділ логіки подання (інтерфейсу користувача) і логіки програми. Підтримка різних типів користувачів, які використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних. Крім ізолювання видів від логіки додатки, концепція MVC істотно зменшує складність великих застосунків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

3.2.2 Статична модель програмного забезпечення

Діаграма класів призначена для надання статичної структури моделі системи в термінології класів об’єктно-орієнтованого програмування. Діаграма класів відображує різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин. На даній діаграмі не вказується інформація про часові аспекти функціонування системи.

Діаграма класів (class diagram) – діаграма на якій представлена сукупність декларативних або статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення, що їх з’єднують.

Складемо проектну модель реалізації варіантів використання відповідно до моделі аналізу. Діаграма класів для варіанту використання «Імпорт вихідних даних» представлено на рисунку 3.5.

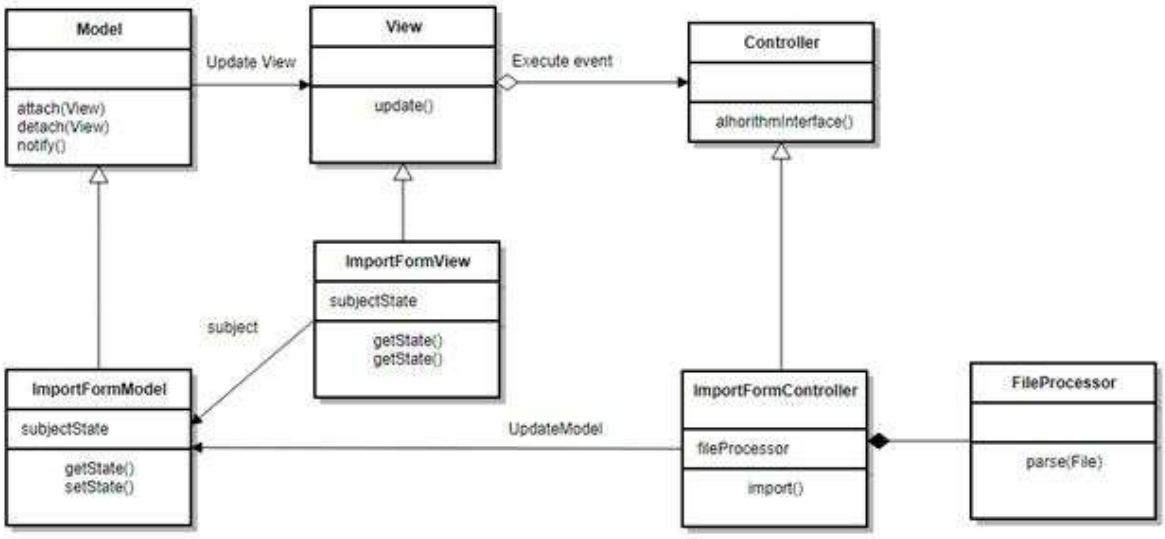

Рисунок 3.5 – Діаграма класів для варіанту «Імпорт вихідних даних»

Діаграма класів для варіанту використання «Розрахунок параметрів нормалізації» представлено на рисунку 3.6

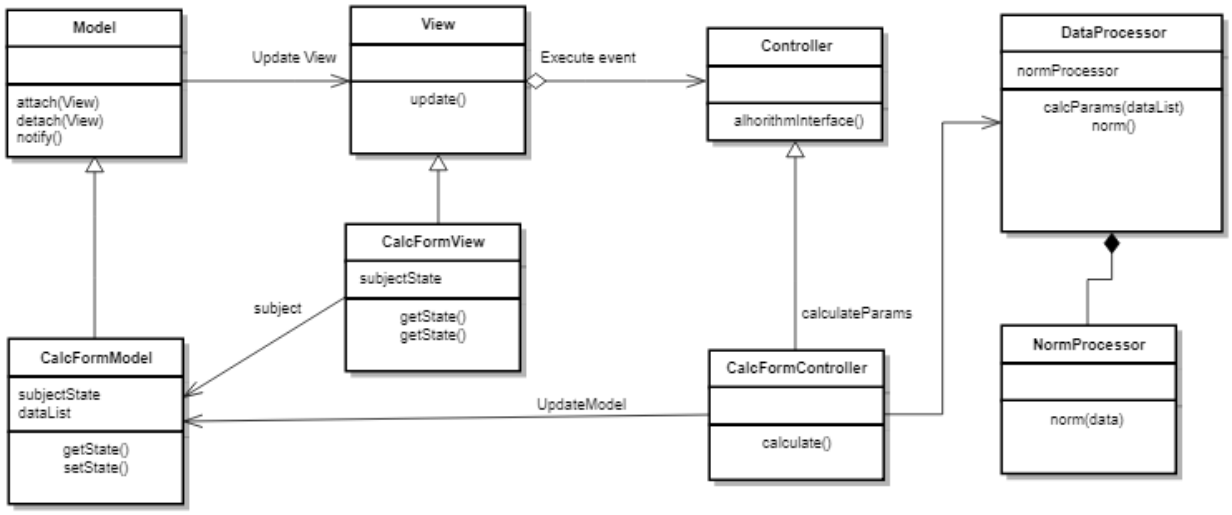

Рисунок 3.6 – Діаграма класів для варіанту використання «Розрахунок параметрів нормалізації»

Діаграми класів для варіантів використання «Імпорт вихідних даних» і «Розрахунок параметрів нормалізації» демонструють класи системи, їх атрибути, методи і взаємозв'язки між ними.

3.2.3 Побудова діаграми послідовності

Діаграма послідовності – діаграма, на якій показані взаємодії об’єктів, упорядковані за часом їхнього прояву. На діаграмі послідовності присутня вісь часу, що дозволяє візуалізувати відношення між переданими повідомленнями.

На діаграмі послідовності зображуються об’єкти, які безпосередньо беруть участь у взаємодії. Для діаграми послідовності ключовим моментом є динаміка взаємодії об’єктів у часі. Кожен об’єкт графічно зображується у формі прямокутника й розташовується у верхній частині своєї лінії життя. У середині прямокутника записується власне ім’я об’єкта, розділені двокрапкою.

Лінія життя об’єкта – вертикальна лінія на діаграмі послідовності, що представляє існування об’єкта протягом певного періоду часу. Лінія життя

об’єкта зображується пунктирною вертикальною лінією. Якщо об’єкт існує в системі постійно, то і його лінія життя триває по всій робочій області діаграми послідовності від самої верхньої її частини до самої нижньої. Окремі об’єкти, закінчивши виконання своїх операцій, можуть бути знищені.

Діаграми послідовності для основних варіантів використання наведено на рисунках 3.7–3.8.

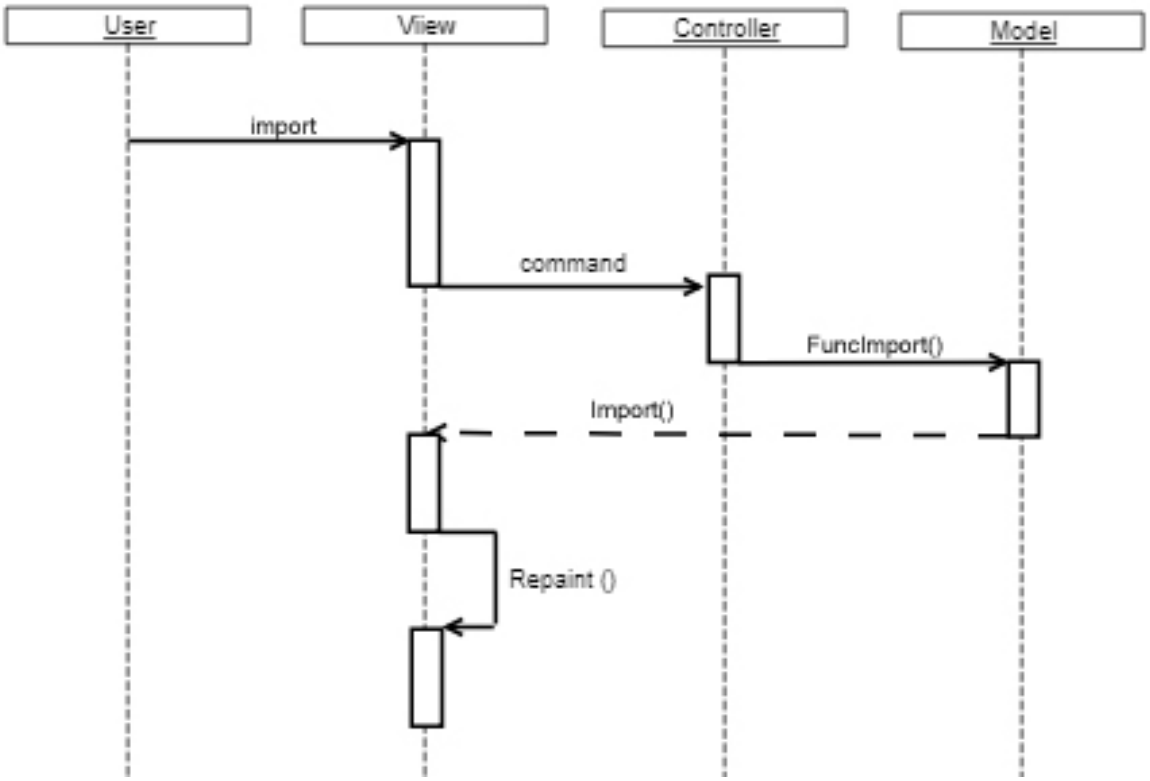

Рисунок 3.7 – Діаграма послідовності реалізації варіанту використання «Імпорт вихідних даних»

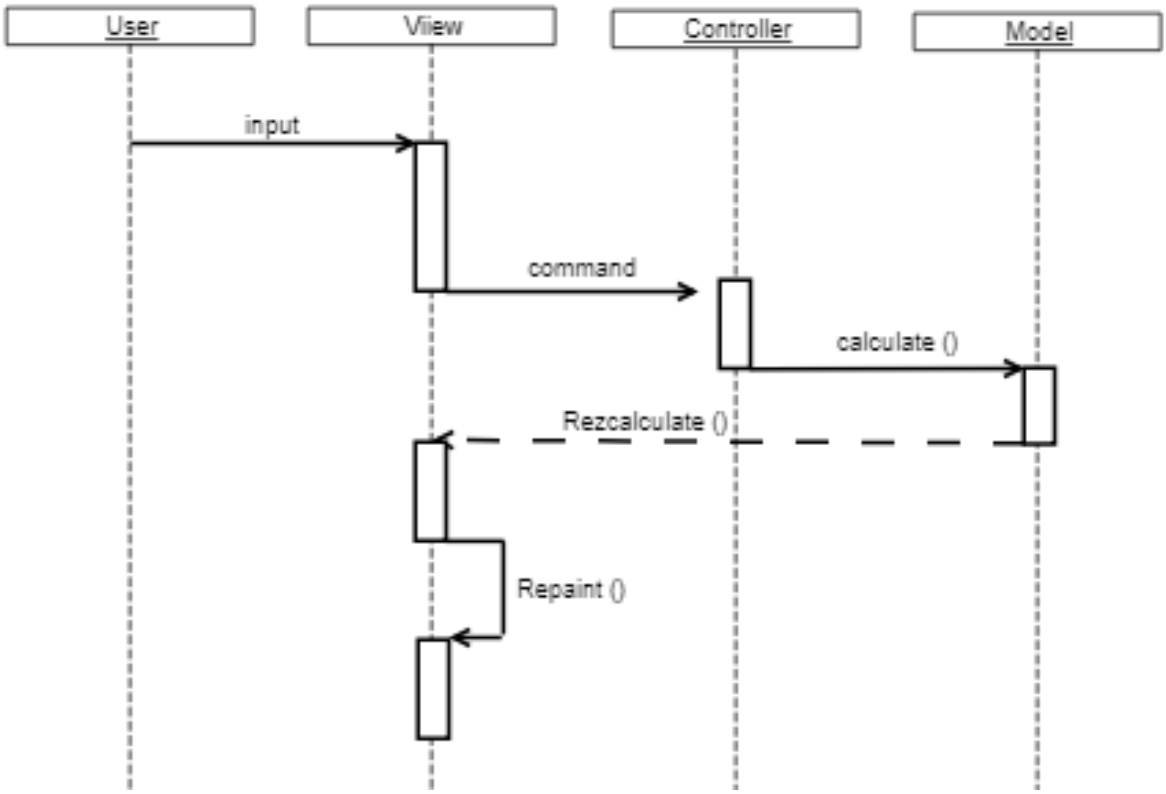

Рисунок 3.8 – Діаграма послідовності реалізації варіанту використання «Розрахунок параметрів нормалізації»

Як правило, кожна окрема діаграма взаємодії описує поведінку тільки в межах одного варіанта використання. На такій діаграмі прийнято відображати екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються один з одним в рамках даного варіанта використання.

На діаграмі послідовностей можна побачити, що відбувається в системі поза очима користувача. Коли користувач відкриває вікно, відображення надсилає запит до контролера для отримання даних, контролер в свою чергу надсилає запит до моделі. Ці дані надходять до відображення пройшовши обробку і відображаються користувачу.

3.3 Робочий проект програмного забезпечення

3.3.1 Обґрунтування вибору мови програмування

В більшості випадків вибір мови програмування відіграє чи не вирішальну роль при створені програмного забезпечення. Деякі мови програмування універсальні та дозволяють вирішувати алгоритмічні задачі будь-якої складності, деякі – спеціалізовані, тобто використовуються для вирішення доволі вузького кола задач. Процес написання програмного коду та сам програмний код повинні бути доволі простим. Обрана мова програмування має полегшувати читання та написання програми.

Після дослідження предметної галузі, огляду способів та засобів вирішення поставлених проблем, аналізу переваг і недоліків існуючих програмних рішень та виходячи із вимог до програмного забезпечення, що наведені у технічному завданні, були вибрані наступні інструменти розробки програмного забезпечення та СКБД, які є оптимальними для вирішення поставлених задач.

В якості мови програмування була обрана Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байт код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU General Public License [28].

Мова значно запозичила синтаксис із C та C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих

програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування [29].

В якості середовища розробки було обрано середовище NetBeans IDE – безкоштовне інтегроване середовище розробки з відкритим вихідним кодом для розробників програмного забезпечення. Середовище надає всі засоби, необхідні для створення професійних десктоп додатків, корпоративних, мобільних і веб-додатків на платформі Java, а також C/C++, PHP, JavaScript, Groovy і Ruby. Робоча область середовища IDE повністю настроюється – існує можливість настройки для користувача дій, які виконуються за допомогою панелі, призначення "гарячих" клавіш і т.д.

Середовище розробки NetBeans IDE має в своєму складі розширений багатомовний редактор для різних мов програмування – Java, C/C++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, документацію Javadoc. Існує можливість розширення функцій редактора з метою підтримки будь-якої іншого мови. Редактор NetBeans робить відступи рядків, перевіряє відповідність дужок і слів, підсвічує синтаксис вихідного коду. Проводиться перевірка помилок під час введення, відображення варіантів для автозавершення коду і фрагментів документації по обраній мові програмування.

Розширені засоби для виконання контекстно-залежного пошуку по всому середовищу IDE, довідкових матеріалах і всіх відкритих проектах та файлах. Існує можливість створення проектів у вільному форматі або починати роботу з проектом з шаблону. У комплекті з середовищем IDE поставляються шаблони і приклади проектів для додатків Java SE, мобільних, веб-додатків і додатків рівня підприємства, додатків JavaFX, модулів NetBeans, додатків Groovy, PHP, C/C++, Ruby і Ruby on Rails. NetBeans має

вбудовану підтримку CVS, Mercurial і Subversion. Для перегляду змін використовується редактор з колірними позначеннями.

3.3.2 Побудова діаграми компонентів

Для створення конкретної фізичної системи необхідно реалізувати всі елементи логічного опису в конкретних матеріальних елементах. Для опису таких реальних елементів призначене фізичне подання моделі. У мові UML це означає сукупність зв'язаних елементів, включаючи програмне і апаратне забезпечення, а також персонал, які організовані для виконання спеціальних завдань. Для фізичного подання моделей систем використовуються діаграми реалізації, які включають дві окремі канонічні діаграми: діаграму компонентів і діаграму розгортання. Компонент призначений для зображення фізичної організації асоційованих з ним елементів моделі. Додатково компонент може мати текстовий стереотип і помічені значення, а деякі компоненти – власне графічне зображення. Правила іменування об'єктів в мові UML вимагають підкреслення імені окремих екземплярів, але стосовно компонентів підкреслення їх імені часто опускають.

Як власні імена компонентів прийнято використовувати імена файлів або блоків апаратури.

В окремих випадках до простого імені компонента може бути додана інформація про ім'я охоплюючого пакету і про конкретну версію реалізації даного компонента. У цьому випадку номер версії записується як помічене значення у фігурних дужках. У інших випадках символ компонента може бути роздільний на секції, щоб явно вказати імена реалізованих в ньому класів або інтерфейсів. Таке позначення компонента називається розширеним. Оскільки компонент як елемент моделі може мати різну фізичну реалізацію, інколи його зображуюсь у формі спеціального графічного символу, що ілюструє конкретні особливості реалізації.

Діаграма компонентів програмного забезпечення представлена на рисунку 3.9.

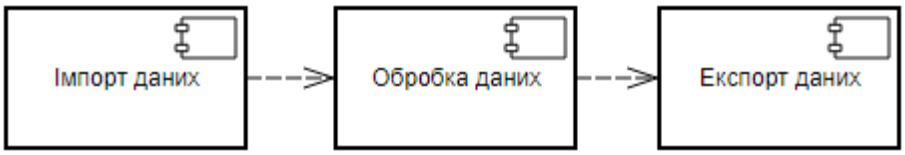

Рисунок 3.9 – Діаграма компонентів

В метамоделі мови UML компонент є нащадком класифікатора. Він надає організацію в рамках фізичного пакету асоційованим з ним елементів моделі. Як класифікатор, компонент може мати також свої власні властивості, такі як атрибути і операції.

3.3.3 Випробування програмного забезпечення

Під час випробувань було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу.

Всі виявлені недоліки ПЗ були зафіксовані в протоколах і усунуті до моменту впровадження ПЗ у дію.

Програму і методику випробувань наведено у Додатку Д.

Методи випробувань

Випробування було проведено за стратегією «чорного ящика». Через велику кількості функцій, які потрібно було випробовувати, представлено результати випробувань основних функцій додатку диспетчера.

Протокол тестування:

1. Проведено перевірку програми на відповідність технічному завданню:

- перевірка роботи функції «Імпорт вихідних даних»:

Після вибору файлу з вихідними даними вони були успішно імпортовані до програми.

- перевірка роботи функції «Розрахунок параметрів вибірки»:

Після імпорту даних для них було успішно розраховано основні параметри вибірки.

- перевірка роботи функції «Перевірка на викиди, та на нормальний закон розподілу»:

Після розрахунку параметрів вибірки програма успішно визначила чи присутні викиди, та припущення про нормальний закон розподілу.

- перевірка роботи функцій «Нормалізація емпіричних даних» та «Перевірка гіпотези про нормальний закон розподілу»:

Далі програма успішно виконала нормалізацію вибірки і її перевірку на нормальний закон розподілу, що підтвердило адекватність нормалізації.

- перевірка роботи функції «Побудова лінійного рівнняння регресії»:

Програма успішно побудувала лінійне рівняння регресії.

- перевірка роботи функції «Побудова довірчих інтервалів для лінійного рівняння регресії»:

- Програма успішно побудувала довірчі інтервали лінійного рівняння регресії.

- перевірка роботи функції «Побудова інтервалів прогнозування для лінійного рівняння регресії»:

Програма успішно побудувала інтервали прогнозування лінійного рівняння регресії.

- перевірка роботи функції перевірка роботи функції «Побудова нелінійного рівнняння регресії»:

Програма успішно побудувала нелінійне рівняння регресії.

- перевірка роботи функції «Побудова довірчих інтервалів для нелінійного рівняння регресії»:

- Програма успішно побудувала довірчі інтервали нелінійного рівняння регресії.

- перевірка роботи функції «Побудова інтервалів прогнозування для нелінійного рівняння регресії»:

Програма успішно побудувала інтервали прогнозування нелінійного рівняння регресії.

- перевірка роботи функції «Експорт результатів у файл»:

Після вибору місця для експорту вихідних даних вони були успішно експортовані в потрібний формат.

2. Проведено перевірку програми на її програмно-апаратну сумісність: виконано тестування на апаратно-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені в технічному завданні; виконано перевірку на наявність і усунення всіх помилок, зазначених у п.1.

3. Проведено візуальний перегляд програми: виконано остаточне налагоджено на предмет виявлення та усунення дрібних помилок.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ BOOTSTRAP

Практичним результатом кваліфікаційної роботи є програма для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap. Було здійснено постановку задачі на розробку ПЗ, розроблено ескізний, технічний та робочий проекти ПЗ.

Згідно з вимогами до ПЗ, наведеними в технічному завданні (Додаток А), програмне забезпечення надає такі функції:

– імпорт вихідних даних;

– розрахунок параметрів вихідних даних;

– перевірка на викиди та нормальний закон розподілу;

– нормалізація емпіричних даних за допомогою десяткового логарифму;

– перевірка гіпотези про нормальний закон розподілу;

– побудова лінійного рівняння регресії;

– побудова довірчих інтервалів для лінійного рівняння регресії;

– побудова інтервалів прогнозування для лінійного рівняння регресії;

– побудова нелінійного рівняння регресії;

– побудова довірчих інтервалів для нелінійного рівняння регресії;

– побудова інтервалів прогнозування для нелінійного рівняння регресії;

– оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

– експот результатів у файл.

Програмне забезпечення повністю відповідає вимогам до організації вхідних та вихідних даних, вимогам до надійності та іншим вимогам, зазначеним в технічному завданні (Додаток А).

Також була розроблена наступна програмна документація:

– технічне завдання (Додаток А);

– опис програми (Додаток Б)

– текст програми (Додаток В);

– інструкція користувача (Додаток Г);

– програма та методика випробувань ПЗ (Додаток Д).

Програмне забезпечення було розроблено на мові програмування Java в середовищі NetBeans IDE.

Створене програмне забезпечення було протестоване на відповідність технічному завданню. Під час тестування програмне забезпечення показало повну працездатність.

Розроблене програмне забезпечення може бути використано користувачем, не здатним до програмування. Інтерфейс є інтуїтивно зрозумілим.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Впровадження інформаційних технологій пов'язано з капітальними вкладеннями як на придбання техніки, так і на розробку проектів, виконання підготовчих робіт і підготовку кадрів. Тому впровадженню повинно передувати економічне обґрунтування доцільності впровадження інформаційної системи. Це означає, що повинна бути визначена ефективність використання автоматизованих комп'ютерних технологій.

Система управління промислових підприємств характеризується різноманітністю функцій управління. Для забезпечення їх інформаційною підтримкою потрібне використання різних класів інформаційних систем.

Для автоматизації проектування використовуються системи класу САПР. Вони в свою чергу підрозділяються на два підкласи: САПР В (САПР виробу) і САПР ТП (САПР технологічних процесів).

До підкласу САПР В відносяться системи розрахунків і інженерного аналізу CAE (Computer Aided Engineering) і системи конструкторського проектування CAD (Computer Aided Design). Система CAM (Computer Aided Manufacturing), в якій виконується проектування технологічних процесів, відноситься до класу САПР ТП.

Для забезпечення єдиного інформаційного простору між CAE/CAD/CAM системами використовується система управління проектними даними PDM (Product Data Management).

Використання даного програмного продукту приведе до значної економії, що виражається в наступних показниках:

- підвищення продуктивності праці;
- економія часу на обробку інформації.

Найбільш важливим моментом для розробника, з економічної точки зору, є процес формування вартості програмного продукту. Очевидно, що

вона являє собою дуже специфічний товар з безліччю особливостей.

Створення програмного продукту вимагає одноразових витрат на його розробку, придбання необхідних технічних засобів і поточних витрат на функціонування системи. Економія від функціонування програмного продукту визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

5.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку ПЗ складаються з витрат на зарплату розробника, на амортизацію комп`ютера, на якому виконується розробка, на експлуатацію цього комп`ютера, на засоби розробки та витратні матеріали і комплектуючі.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 10000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 12000 грн/міс, а вартість сучасного комп`ютера складає 15000 грн. (приведена вартість комп`ютера на базі Intel Core i9).

Вартість розробки ПЗ розраховується по формулі:

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

де Т – тривалість розробки, міс.;

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

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

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

$З_е$ – витрати на електроенергію;

$З_м$ – витрати на основні і допоміжні матеріали.

$З_е$ при споживаній потужності 0,5 кВт, тривалості роботи на місяць, рівної 22*6 = 132 годин, вартості кіловат-години електроенергії 1,68 грн. складає

$$З_е = 132 * 1,68 * 0,5 = 110,88 \text{ грн.}$$

Витрати на допоміжні матеріали приведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума, грн.
Папір	90,00
Заправлення картриджа до принтера	120,00
Література	450,00
Непередбачені витрати	250,00
Разом	910,00

Витрати на розрозку ПЗ приведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку системи

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Міс.	1
Основна і додаткова заробітна плата	Грн.	12000,00
Відрахування на соціальні заходи	Грн.	4560,00
Загальногосподарські витрати	Грн.	1000,00
Витрати на допоміжні матеріали	Грн.	910,00
Витрати на електроенергію	Грн.	110,88

Відповідно до формули (5.1), вартість розробки ПЗ складає:

$$С_{пр} = (12000 + 4560 + 1000 + 110,88) * 1 + 910 = 18580,88 \text{ грн.}$$

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

$$А_{об} = 15000 * 0,6 = 9000 \text{ грн.}$$

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

$$В_м = 15000 * 0,05 = 750 \text{грн.}$$

Річний обсяг робіт комп`ютера у годинах визначається в такий спосіб:

$$Ф_м = 264,5 * Т_з \qquad (5.2)$$

де $Т_з$ – це середнє місячне завантаження устаткування (близько 4 годин);

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе:

$$Ф_м = 264,5 * 4 = 1058 \text{ годин.}$$

Витрати на електроенергію $З_е$ при 1058 годинах роботи устаткування в рік складуть

$$З_е = 1058 * 1,68 * 0,5 = 888,72 \text{ грн.}$$

Експлуатаційні витрати для комп`ютера за рік складуть:

$$З_{зр} = 9000 + 750 + 888,72 = 10638,72 \text{ грн.}$$

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

$$З_{се} = 18580,88 + 10638,72 = 29219,60 \text{ грн.}$$

5.2 Економічна ефективність розробки і впровадження програмного забезпечення

Основним показником економічної ефективності функціонування системи є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

– високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);

– підвищені витрати на канцтовари;

– витрати, пов'язані з коригуванням раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з впровадженням ПЗ, відносяться:

– підвищення продуктивності праці;

– вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності ПЗ є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження ПЗ вивільняє 0,6 працівника (за експертною оцінкою фахівців).

Зарплата 0,6 працівника в рік складає:

(8000 * 12)* 0,6 = 57600 грн.

Річний економічний ефект розраховується по формулі:

$$Э_{год} = \Delta C_n - E_n \cdot k,$$

де ΔC_n – вивільнені кошти після впровадження ПЗ (57600 грн.) мінус експлуатаційні витрати (10638,72 грн.) = 46961,28грн.

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

k – одноразові витрати на впровадження продукту (29219,60)

$$Э_{год} = 46961,28 – 29219,60 * 0,6 = 10645,01 \text{ (грн.)}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{29219,60}{46961,28} \approx 0,62$$

Отже строк окупності системи складає приблизно 6 місяців.

6 ОХОРОНА ПРАЦІ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [30].

Метою політики охорони праці е зведення до мінімуму показників виробничого травматизму та професійних захворювань. Ця мета набула нових форм у ЄС протягом останніх років і поширилася сьогодні до пропаганди "добробуту на роботі", що означає моральний, фізичний та соціальний добробут, а не лише відсутність нещасних випадків та професійних захворювань.

Крім того, необхідно також досягти низки допоміжних цілей:

– профілактика соціальних ризиків (стресів, домагань на робочому місці, депресій та роздратування, а також ризиків, які пов'язані з алкогольною, наркотичною залежністю);

– аналіз ризиків, пов'язаних із роботою, а також ергономічні, психологічні та соціальні ризики;

– урахування змін у формах зайнятості, організації роботи та робочого часу працівників з нестандартною та тимчасовою зайнятістю;

– урахування розмірів підприємства (конкретні заходи щодо інформування, підвищення рівня обізнаності, програм попередження ризиків на малих та середніх підприємствах, приватних підприємців, домашньої обслуги тощо);

– інтенсивна профілактика професійних захворювань (спричинених азбестом, втрата слуху, проблеми опорно-рухового апарату);

– урахування демографічних змін.

Основними завданнями управління охороною праці є:

1) Опрацювання заходів щодо здійснення державної політики з охорони праці на регіональному та галузевому рівнях;

2) підготовка, прийняття та реалізація заходів, спрямованих на забезпечення:

– належних, безпечних і здорових умов праці;

– утримання в належному стані виробничого устаткування, будівель і споруд, інженерних мереж, безпечного ведення технологічних процесів;

– необхідних засобів індивідуального захисту для працівників;

– організації і проведення навчання працівників з питань охорони праці;

– пропаганди охорони праці;

– обліку, аналізу та оцінки стану умов і безпеки праці;

– професійного добору працівників окремих спеціальностей;

– страхування працівників від нещасного випадку на виробництві та профзахворювань;

– організаційно-методичне керівництво на регіональному та галузевому рівнях;

– стимулювання інтеграції управління охороною праці в єдину систему загального управління організацією виробництва;

– широке впровадження позитивного досвіду у сфері охорони праці. За умови економічної, екологічної та демографічної кризи в Україні, склалася надзвичайна ситуація з безпекою та умовами праці на більшості підприємств, особливо середнього і малого бізнесу.

6.1 Аналіз шкідливих та небезпечних факторів при роботі з персональним комп’ютером

До фізичних шкідливих і небезпечних чинників відносяться:

– підвищені рівні електромагнітного, рентгенівського, ультрафіолетового і інфрачервоного випромінювання;

– підвищений рівень статичної електрики і запилене повітря робочої зони;

– підвищений вміст позитивних іонів і понижений вміст негативних іонів в повітрі робочої зони;

– підвищений рівень яскравості;

– нерівномірність розподілу яскравості в полі зору;

– підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини.

Хімічні шкідливі і небезпечні чинники наступні: підвищений вміст в повітрі робочої зони двоокису вуглецю, озону, аміаку, фенолу і формальдегіду.

Психофізіологічні шкідливі і небезпечні фактори:

– напруга зору і уваги;

– інтелектуальні, емоційні і тривалі статичні навантаження;

– монотонність праці;

– великий обсяг інформації, що обробляється в одиницю часу;

– нераціональна організація робочого місця.

Типовими відчуттями, які відчувають до кінця робочого дня оператори ПЕОМ, є: перевтома очей, головний біль, тягучий біль в м'язах шиї, рук і спини, зниження концентрації уваги.

Уже в перші роки комп'ютеризації було відзначено специфічне зорове стомлення у користувачів дисплеїв, що отримало загальну назву «комп'ютерний зоровий синдром». Однією з причин є те, що сформована за мільйони років еволюції зорова система людини пристосована для сприйняття об'єктів у відбитому світлі (друковані тексти, малюнки тощо), а не для роботи за дисплеєм. Зображення на дисплеї принципово відрізняється від звичних оку об'єктів спостереження – воно світиться, мерехтить, складається з дискретних точок, а кольорове комп'ютерне зображення не відповідає природним кольорам. Але не тільки особливості зображення на екрані викликають зорове стомлення.

Велике навантаження орган зору відчуває при введенні інформації, так як користувач змушений часто переводити погляд з екрану на текст і клавіатуру, що знаходяться на різній відстані і по-різному освітлені. Зорове стомлення проявляється скаргами на затуманення зору, труднощі при перенесенні погляду з ближніх предметів на дальні і з далеких на ближні, що здаються зміни забарвлення предметів, їх двоїння, відчуття печіння, «піску» в очах, почервоніння повік, болі при рухання очима.

Тривала і інтенсивна робота на комп'ютері може стати джерелом важких професійних захворювань, таких, як травма повторюваних навантажень (ТПН), що представляє собою поступове накопичення нездужань, котрі переходять у захворювання нервів, м'язів і сухожиль руки.

До професійних захворювань, пов'язаних з ТПН, відносяться:

– тендовагініт – запалення сухожиль кисті, зап'ястя, плеча;

– тендосиновіт – запалення синовіальної оболонки сухожильного основи кисті і зап'ястя;

– синдром зап'ястного каналу (СЗК) – викликається утиском серединного нерва в зап'ястному каналі. Травма викликає утворення продуктів розпаду в області зап'ястного каналу, в результаті чого спочатку виникає набряк, а потім СЗК.

З'являються скарги на пекучий біль і поколювання в зап'ясті, долоні, а також пальцях, крім мізинця. Спостерігається хворобливість і оніміння, ослаблення м'язів, що забезпечують рух великого пальця.

Ці захворювання зазвичай настають в результаті безперервної роботи на неправильно організованому робочому місці.

Механізм порушень, що відбуваються в організмі під впливом електромагнітних полів, обумовлений їх специфічною (нетепловою) і тепловою дією.

6.2 Розрахунок захисного заземлення офісного приміщення

Допустимий опір: $R_o = 4$

Ом. Тип грунту: садова

земля.

Довжина труби заземлювача: $l = 3$

м. Діаметр труби заземлювача: $d =$

0,045 м.

Глибина закладення грубі від поверхні: $t_o =$

0,5 м. Розрахунок:

1) Питомий опір ґрунту за даними таблиці – $p = 50$ Ом * м

2) Для вибраного типу заземлювача і його розмірів визначаємо опір одного заземлювача. Для трубного заземлювача розташованого в грунті, опір розтікання:

$$t = \frac{3}{2} + 0.5 = 1.5 \qquad R_i = \frac{p}{2\pi * l}\left(\ln\frac{2l}{d} + \frac{1}{2} * \ln\frac{4t + l}{4t - l}\right), t = \frac{l}{2} + t_0$$

Тоді:

$$R_i = \frac{50}{18.84}\left(\ln\frac{6}{0.045} + \frac{1}{2} * \ln\frac{4 * 1 + 3}{4 * 1 - 3}\right) = 2.653 * \left(\ln 133.33 + 0.5 * \ln\frac{8 + 3}{8 - 3}\right) = 14.\ldots \text{ Ом}$$

3) Якщо опір одного заземлювача не перевищує допустимий опір пристрою $R_j < R_o$. то приймаємо 1 штучний заземлювач. Відповідно, якщо $R_j > R_o$, то необхідно приймати кілька штучних заземлювачів, з'єднаних паралельно. У випадку, що розглядається, $R_i > R_o$.

4) Визначаємо необхідну кількість паралельно з'єднаних заземлювачів:

$n = \frac{R_i}{\eta_c * R_o} = \ldots$ – коефіцієнт використання заземлителя ( ).

$n = \frac{14.026}{4} = 3.51$ -> n ≥ n: n = 4 заземлювача необхідно.

5) Для зв'язку вершин заземлювачів приймають сполучну лінію. Опір розтікання сполучної риси визначають за формулою:

l_c=1.05*a*n*l=1.05*2*4*3= 25.2 Ом

$$\frac{\rho}{2\pi l_c} \ln\frac{l^2}{d \cdot t} = \frac{58}{2 \cdot 3.14 \cdot 25.2} \cdot \ln\frac{3}{1.5 \cdot 10^{-2}} = \frac{58}{158.256} \cdot \ln\frac{3}{0.05} = 0.31 \cdot 4.6 = 1.426 \text{ Ом}$$

$$R_{черн} = \frac{1.426}{0.89} = 1.6 \text{ Ом}$$

6) Визначаємо фактичний опір заземлювального пристрою:

$$\frac{R_в \cdot R_{черн} \cdot k_г}{R_в + R_{черн}} = \frac{\cdot 1.4}{+ 1.6} = \frac{}{} = 1.17 \text{ Ом}$$

6.3 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Розглянемо детальніше чинники, котрі призводять до проблем із зором при роботі за персональним комп’ютером.

Основні чинники, які призводять до виникнення проблем із зоровим апаратом:

- недостатнє освітлення робочого місця;
- високий контраст між монітором та оточуючим середовищем.
- відблиски на моніторі;
- мала дистанція між користувачем на монітором;
- мала частота кліпання повіками.

Для покращення освітлення робочого місця зазвичай використовують одну настільну лампу – лампу розжарювання чи компактну ртутну ЛДС, рідше світлодіодну. Але одне джерело світла забезпечує нерівномірне освітлення робочого місця та часто створює відблиски на екрані монітору. Щоб забезпечити рівномірне освітлення також використовують два

когерентні джерела світла, наприклад: дві ртутні ЛДС, частоти випромінювання яких є когерентні і через це, вони посилюють одна одну. Але приведений метод, дозволяє тільки частково зменшити контраст між працюючим екраном та оточуючим середовищем і є не економічним.

Принцип дії запропонованого методу досить простий і полягає в розміщенні 4 напрямлених перпендикулярно до користувача джерел світла на корпусі монітору для тилового підсвічування. В якості джерел світла були використані світлодіодні стрічки. Колір випромінювання був обраний спектрально-оптимальний – білий «теплий» (2700–3200 К). А залежно від моделі LED, мають кут розсіювання від 30 до 270 градусів, що дозволяє використовувати їх як для акцентного підсвічування, так і для загального освітлення. Варто також зауважити, що світлодіоди не несуть ніякої загрози навколишньому середовищу.

Проведено експеримент з фотоапаратом та опитування серед програмістів. За допомогою цифрового фотоапарата було створено серію фотознімків у «ручному» режимі, тобто без автоматичного налаштування яскравості, витримки та балансу білого, з тиловою підсвіткою та без неї. Дане рішення дозволяє помітно зменшити контраст, різкий перепад яскравості; завдяки використання LED-стрічки – отримати мініатюрні габарити, низьку собівартість, високу тривалість роботи та безпечність джерел освітлення [31].

Вимоги безпеки під час роботи з комп’ютером Щодня перед початком роботи оператор повинен:

– оглянути своє робоче місце;

– про виявлення ознак пошкодження обладнання інформувати свого безпосереднього керівника;

– відрегулювати освітленість на робочому місці, переконатися в відсутності відблисків на екрані комп’ютера, відсутності зустрічного світла;

– перевірити правильність підключення обладнання ЕОМ до

електромережі;

– очистити екран комп’ютера від пилу та інших забруднень;

– перевірити правильність організації робочого місця й за необхідності провести відповідні корегування.

Оператор під час роботи зобов'язаний:

– виконувати тільки ту роботу, яку йому було доручено;

– підтримувати порядок і чистоту на робочому місці;

– тримати відкритими всі вентиляційні отвори обладнання;

– коректно закрити всі активні завдання у разі припинення роботи з комп’ютером;

– негайно відключити комп’ютером від електричної мережі у разі виникнення аварійної ситуації.

У ході виконання робіт оператор комп’ютера повинен:

– витримувати відстань від очей до екрана комп’ютером в межах 60- 70 см;

– дотримуватися змінного режиму праці та відпочинку, регламентованих перерв у роботі;

– для розробників програм – тривалістю 15 хвилин через кожну годину роботи;

– для інших категорій працівників – тривалістю 15 хвилин через кожні дві години роботи;

– для операторів комп'ютерного набору – тривалістю 10 хвилин, після кожної години роботи.

Під час регламентованих перерв рекомендується виконувати комплекси вправ для очей, рук, хребта, поліпшення мозкового кровообігу тощо. Про виявлення несправності обладнання або інших факторів, які створюють загрозу для життя або здоров'я працівників, необхідно негайно інформувати свого безпосереднього керівника.

Не допускається:

– виконання ремонту та налагодження комп’ютерної техніки безпосередньо на робочому місці оператора;

– зберігання біля комп’ютера паперу, дискет, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;

– відключення захисних пристроїв, самочинні зміни в конструкції комп’ютера;

– використання комп’ютерів, на екранах яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;

– доторкання до задньої панелі системного блоку при включеному живленні;

– вимикання живлення під час виконання активного завдання;

– попадання вологи на поверхню системного блоку, монітора, клавіатури, дисководів, принтерів та інших пристроїв;

– приймання напоїв та їжі на робочому місці.

Після закінчення роботи з використанням необхідно дотримуватися такої послідовності вимикання обладнання:

– закрити всі активні завдання;

– переконатися у відсутності дискет та дисків у дисководах;

– використавши опцію "Завершення роботи" у меню "Пуск", вимкнути живлення системного блоку;

– вимкнути живлення всіх комп’ютерів;

– вимкнути блок аварійного живлення (за наявності);

– відключити комп’ютер від електромережі, при цьому забороняється тягнути штепсельну вилку за дріт.

У випадку виникнення аварійної ситуації оператор зобов'язаний:

– у всіх випадках виявлення пошкодження проводів електричного живлення, несправності заземлення та інших пошкодженнях електрообладнання, виникненні запаху гарі, диму – негайно вимкнути

електричне живлення і повідомити про аварійну ситуацію свого безпосереднього керівника й чергового електрика;

– при попаданні людини під електричну напругу негайно звільнити її від дії струму шляхом вимкнення електричного живлення, до прибуття лікаря надати потерпілому долікарську медичну допомогу;

– при будь-яких випадках порушень роботи технічного обладнання або програмного забезпечення негайно викликати представника технічної служби з питань експлуатації обчислювальної техніки;

– у випадку виникнення різі в очах, різкого погіршення зору, виникнення головного болю, больових відчуттів у пальцях та кистях рук, посилення серцебиття – негайно припинити роботу з використанням ЕОМ, повідомити про те, що сталося, свого безпосереднього керівника й звернутися до медичної установи;

– при загорянні обладнання негайно відключити його від електромережі;

– про загорання повідомити свого безпосереднього керівника, оперативного чергового, пожежну службу; ужити заходів щодо ліквідації вогню за допомогою вуглекислотного або порошкового вогнегасника.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього середовища являє собою форму відносин між суспільством і природою. Вона здійснюється різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними та іншими.

В загальному випадку проблема охорони навколишнього середовища зводиться до вирішення двох завдань:

– організації раціонального природокористування;

– забезпечення чистоти природних (екологічних) систем. При здійсненні різних видів економічної діяльності суб'єкти господарювання використовують різноманітні природні ресурси: землю, воду, корисні копалини тощо. Проте ресурси ці обмежені. Обмеженість природних ресурсів була і залишається головною і дуже жорсткою умовою, що накладається на розвиток економіки і відповідно зростання суспільного добробуту.

Наслідком обмеженості природних ресурсів є конкуренція за їх застосування, тобто суперництво між альтернативними цілями використання ресурсів. Адже майже всі ресурси можуть використовуватися для задоволення найрізноманітніших потреб. Наприклад, нафта може служити сировиною для одержання палива, виробництва синтетичних волокон, пластмас, лакофарбових виробів, побутової хімії тощо. І всі ці альтернативні цілі конкурують за використання сирої нафти, обсяги якої, як відомо, обмежені.

Раціональне природокористування означає розробку та здійснення концепції і конкретних заходів щодо раціонального використання і відтворення природних ресурсів, гармонічну взаємодію суспільства і природи, людини і навколишнього природного середовища.

Завдання організації раціонального природокористування вирішується

шляхом:

– оптимального розподілу ресурсів між різними господарськими цілями;

– використання технологій, що зберігають ресурси;

– проведення заходів щодо поповнення природних ресурсів.

Іншим, не менш важливим, завданням охорони навколишнього середовища є забезпечення чистоти природних екологічних систем, тобто водного середовища, повітряного басейну, ґрунтових покровів тощо, з тим, щоб забезпечити населення екологічно чистими продуктами харчування, водою, повітрям і, в остаточному підсумку, зберегти високий рівень здоров'я населення та його активного довголіття.

Економічна діяльність у всіх її проявах здійснює забруднення навколишнього середовища. У процесі цієї діяльності забруднюються і стають дефіцитними ресурси повітря, води, територій, що здавалися нескінченними. Нині рівень забруднення досяг загрозливих розмірів, набувши по суті кризового характеру.

Основними принципами охорони навколишнього природного середовища є:

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

б) гарантування екологічно безпечного середовища для життя і здоров'я людей;

в) запобіжний характер заходів щодо охорони навколишнього природного середовища;

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

д) збереження просторової та видової різноманітності і цілісності природних об'єктів і комплексів;

е) науково обґрунтоване узгодження екологічних, економічних та соціальних інтересів суспільства на основі поєднання міждисциплінарних знань екологічних, соціальних, природничих і технічних наук та прогнозування стану навколишнього природного середовища;

є) обов'язковість екологічної експертизи;

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

з) науково обґрунтоване нормування впливу господарської та іншої діяльності на навколишнє природне середовище;

и) безоплатність загального та платність спеціального використання природних ресурсів для господарської діяльності;

і) стягнення збору за забруднення навколишнього природного середовища та погіршення якості природних ресурсів, компенсація шкоди, заподіяної порушенням законодавства про охорону навколишнього природного середовища;

ї) вирішення питань охорони навколишнього природного середовища та використання природних ресурсів з урахуванням ступеня антропогенної змінності територій, сукупної дії факторів, що негативно впливають на екологічну обстановку;

й) поєднання заходів стимулювання і відповідальності у справі охорони навколишнього середовища;

к) вирішення проблем охорони навколишнього природного середовища на основі широкого міждержавного співробітництва.

Природні ресурси України є власністю народу України, який має право на володіння, використання та розпорядження природними багатствами [32].

7.1 Забруднення навколишнього середовища

Охорона навколишнього середовища є формою відносин між природою та суспільством. Існують різні засоби, через які вона може здійснюватись. Серед них: економічні, правові, гігієнічні, науково-технічні, біологічні та інші.

Дві основні проблеми, які вирішуються у процесі охорони навколишнього середовища – це забезпечення чистоти, її підтримання та організація раціонального природокористування.

Будь-яка економічна діяльність забруднює природнє середовище. Як результат – дефіцит та забрудненні повітряні ресурси, території, вода.

За порушення у сфері охорони природи, незаконне використання природних ресурсів законодавством України передбачена адміністративна (статті 52-92 Кодексу про адміністративні правопорушення) та кримінальна (статті 236-254 Кримінального кодексу) відповідальність.

Утилізація – це обов'язкова процедура для підприємств і організацій різної форми власності. Неналежне виконання цієї процедури призводить до податкової та адміністративної відповідальності.

Банківські установи, які експлуатують термінальні мережі – банкомати та інформаційно-платіжні термінали самообслуговування – використовують у своїй діяльності персональні комп`ютери, монітори, клавіатури, маніпулятори «миша», ноутбуки, копіровальну техніку, інші електронні компоненти, а також різноманітне канцелярське приладдя (бумагу, тонер для принтерів та копіровальної техніки, батарейки і т.п.). Все це потрібно правильно утилізувати, щоб мінімізувати шкоду для навколишнього середовища.

7.2 Розробка заходів щодо зменшення забруднення навколишнього середовища

Використанні батарейки збирають у спеціальний контейнер, який потім здається до відповідних інстанцій, які займаються переробкою батарейок. Адже використані батарейки містять у собі низку небезпечних токсичних складових, як свинець, марганець, літій і цинк. Якщо їх викидати на звичайні звалища, а не переробляти, то токсини починають отруювати воду і ґрунт, а також те, що поблизу росте. Зрештою, це, так чи інакше, потрапляє в людський організм.

Комп'ютери та оргтехніка, як відомо, далеко не вічні – і, як тільки вони вичерпають свій фізичний ресурс, їх потрібно утилізувати. Для чого потрібна утилізація оргтехніки? Справа в тому, що офісна техніка має в своєму складі як матеріали на основі фенолформальдегіда і полівінілхлориду, так і майже всі метали з періодичної таблиці Менделєєва. В процесі роботи комп'ютерів, принтерів, сканерів та іншого обладнання дані компоненти не представляють ніякої небезпеки для здоров'я людини. Зовсім інша справа – коли відпрацьовавший свій термін виріб відправляється на міське звалище.

Завдяки впливу вологи, метали, що містяться в електронних компонентах (миш'як, кадмій, цинк і свинець) переходять в розчинні сполуки, які представляють собою сильні отрути. Нагальною екологічною проблемою є і утилізація пластиків, які містять в собі хлорні сполуки, а також ароматичні вуглеводні.

Крім того, утилізація оргтехніки та комп'ютерів – це обов'язкова процедура для підприємств і організацій різної форми власності. Неналежне виконання цієї процедури призводить до податкової та адміністративної відповідальності.

Завдяки комплексному підходу до проблеми утилізації оргтехніки кількість не переробляються відходів зводиться до мінімуму, а цінні

матеріали і компоненти, такі як чорні і кольорові метали, люмінофор, рідкісні метали вилучаються і повертаються у виробництво. Золото і срібло міститься в техніці в малих кількостях, але після переробки на заводі воно здається в Державний фонд.

За фактом виконання утилізації оргтехніки і комп'ютерів оформляється акт на списання основних засобів підприємства..

Таким чином, правильно утилізувати оргтехніку потрібно як по природоохоронним, так і з економічних міркувань.

ВИСНОВКИ

Мета кваліфікаційної (магістерської) роботи – підвищення достовірності оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap – досягнута за рахунок удосконалення однофакторної нелінійної регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Під час написання магістерської роботи поставлені завдання було виконано у повному обсязі. Одержані наступні результати:

– проаналізовано та порівняно існуючі моделі оцінювання розміру ПЗ веб-застосунків;

– обґрунтувано необхідність удосконалення регресійної моделі для оцінювання розміру ПЗ веб-застосунків;

– розроблена удосконалена регресійна модель для оцінювання розміру веб-застосунків із застосуванням нормалізуючих перетворень;

– побудувана регресійна модель для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

– обрано нормалізуючи перетворення;

– перевірено вихідні ЕД на викиди;

– нормалізовано отримані ЕД, використовуючи обране нормалізуюче перетворення;

– побудувана лінійна регресійна модель, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– побудувана нелінійна регресійна модель, довірчий інтервал та інтервал прогнозування для вихідних даних;

– перевірена адекватність побудованої регресійної моделі для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

– розроблено ПЗ для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

– виконані спецрозділи з охорони труда, охорони навколишнього середовища та економічний розділи.

ПЗ для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap, яке розроблено в рамках магістерської роботи, дозволить автоматизувати та скоротити час відповідних розрахунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Самые Популярные CSS Фреймворки в 2020 для Мобильных Устройств [Електронний ресурс] / Режим доступу: \www/ URL: https://merehead.com/ru/blog/trends-of-css-frameworks-2020/ – 07.09.2020 р. – Загол. з екрану.

2. Що таке веб додаток? [Електронний ресурс] / Режим доступу: \www/ URL: http://www.ittexnoall.com/index.php/sozdanie-sajta/10-sozdanie-sajta/246-shcho-take-web-dodatok.html – 07.09.2020 р. – Загол. з екрану.

3. Web Frameworks: How To Get Started [Електронний ресурс] / Режим доступу: \www/ URL: https://djangostars.com/blog/what-is-a-web-framework/ – 07.09.2020 р. – Загол. з екрану.

4. Build fast, responsive sites with Bootstrap [Електронний ресурс] / Режим доступу: \www/ URL: https://getbootstrap.com/ – 07.09.2020 р. – Загол. з екрану.

5. Briand, L.C. Property Based Software Engineering Measurement [Text] / L.C. Briand, S. Morasca, V.R. Basili // IEEE Transaction on Software Engineering, vol. 22, no. 1, 2009. – pp. 68-86.

6. Макарова, Л.М. Математична модель часу відновлення працездатності банкоматів Wincor Nixdorf Procash 2000xe [Text] / Л.М. Макарова, Є.В. Абаза // VI міжнародна науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем». – Дніпро, ДВНЗ УДХТУ 4 – 6 листопада 2020. – С. 49-50.

7. Фреймворк – важный инструмент программиста. Обзор HTML/CSS, PHP и Python-фреймворков [Електронний ресурс] / Режим доступу: \www/ URL: https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks/ – 17.09.2020 р. – Загол. з екрану.

8. Bootstrap Expo [Електронний ресурс] / Режим доступу: \www/ URL: https://expo.getbootstrap.com/ – 17.09.2020 р. – Загол. з екрану.

9. Mark, Otto. Bootstrap from Twitter. Developer Blog. Twitter [Електронний ресурс] / Режим доступу: \www/ URL: https://blog.twitter.com/developer/en_us/a/2011/bootstrap-twitter.html – 17.09.2020 р. – Загол. з екрану.

10. 1061-1998-IEEE Standard for a Software Quality Metrics Methodology [Електронний ресурс] / Режим доступу: \www/ URL: https://ieeexplore.ieee.org/document/749159 – 17.09.2020 р. – Загол. з екрану.

11. Briand, L.C. Resource Estimation in Software Engineering [Text] / L.C. Briand, I. Wieczorek // Encyclopedia of Software Engineering. – John Wiley & Sons, Inc., 2002. – 83 p.

12. Tan, H.B.K. Estimating LOC for information systems from their conceptual data models [Text] / H.B.K. Tan, Y. Zhao, H. Zhang // in Proceedings of the 28th International Conference on Software Engineering (ICSE '06), May 20-28, 2006, Shanghai, China, pp. 321-330.

13. Магнус, Я.Р. Эконометрика. Начальный курс: Учеб. – 6-е изд., перераб. и доп. [Текст] / Я.Р. Магнус, П.К. Катышев, А.А. Пересецкий. – М.: Дело, 2004. – 576 с.

14 Приходько, С.Б. Доверительный интервал нелинейной регрессии времени восстановления работоспособности устройств терминальной сети / С.Б. Приходько, Л.Н. Макарова // Восточно-Европейский журнал передовых технологий. – 2014. – № 3(4). – С.26-31.

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

16. Орлов, А.И. Прикладная статистика. [Текст] / А.И. Орлов. – М.: Экзамен, 2004. – 117 с.

17. Приходько, С.Б. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Обробка експериментальних даних на комп`ютері» [Текст] / С.Б. Приходько, Л.М. Макарова, К.С. Пугаченко. – Миколаїв: НУК, 2018. – 76 с.

18. Грешилов, А.А. Математические методы построения прогнозов [Текст] / А.А. Грешилов, В.А. Стакун, А.А. Стакун. – М.: Радио и связь, 1997. – 112 с.

19. Демиденко, Е.З. Линейная и нелинейная регрессии [Текст] / Е.З. Демиденко. – М.: Финансы и статистика, 1981. – 302 с.

20. Bates, D.M. Nonlinear Regression Analysis and Its Applications [Text] / D.M. Bates, D.G. Watts, – Wiley, 1988. – 384 p.

21. Pardoe, I. Applied regression modeling [Text] / I. Pardoe. – Wiley, 2012. – 325 p.

22. Seber, George A.F. Nonlinear Regression [Text] / George A.F. Seber, C.J. Wild. – John Wiley & Sons, Inc., 2003. – 792 p.

23. Регресійна модель для оцінювання розміру веб-додатків, розроблених з використанням фреймворку VUE / Л.М. Макарова, С.І. Димченко // Інформаційні управляючі системи і технології (ІУСТ-Одеса – 2019) : матеріали VIII Міжнародної науково-практичної конференції (23–25 верес. 2019 р., м. Одеса) / відп. ред. В. В. Вичужанін ; Одес. нац. політех. ун-т. – Одеса : Екологія, 2019. – С.209-210.

24. Transform data to normal distribution [Електронний ресурс] / Режим доступу: \www/ URL: http://www.sigmamagic.com/forum/archives/297 – 05.10.2020 р. – Загол. з екрану.

25. Аппроксимация закона распределения экспериментальных данных [Електронний ресурс] / Режим доступу: \www/ URL: http://opds.sut.ru/old/electronic_manuals/oed/f05.htm#r054 – 05.10.2020 р. – Загол. з екрану.

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

27. Фаулер, М. UML. Основы [Text] / М. Фаулер // 3-е издание.– СПб Символ Плюс, 2004. – 192 с.

28. The GNU General Public License [Електронний ресурс] / Режим доступу: \www/ URL: http://www.gnu.org/licenses/licenses.en.html – 12.11.2020 р. – Загол. з екрану.

29. Подробнее о технологии Java [Електронний ресурс] / Режим доступу: \www/ URL: https://www.java.com/ru/about/ – 12.11.2020 р. – Загол. з екрану.

30. Закон України «Про охорону праці» [Електронний ресурс] / Режим доступу: \www/ URL: http://zakon3.rada.gov.ua/laws/show/2694-12 – 15.11.2020 р. – Загол. з екрану.

31. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень [Електронний ресурс] / Режим доступу: \www/ URL: http://zakon0.rada.gov.ua/rada/show/va042282-99 – 15.11.2020 р. – Загол. з екрану.

32. Кисельов, М. Екологічна політика. Філософський енциклопедичний словник [Текст] / В.І. Шинкарук; Л.В. Озадовська, Н.П. Поліщук; І.О. Покаржевська // Київ : Абрис, 2002. – 742 с.

Додаток А Технічне завдання

Вступ

Назва розроблюваного проекту: «Програма для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap». Галузь застосування – підприємства, які спеціалізуються на розробці ПЗ веб-застосунків.

1. Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну (магістерську) роботу, видане кафедрою ПЗАС НУК ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Дана розробка призначена для побудови лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

2.2 Функціональне призначення програми

Функціональним призначенням програми є автоматизація розрахунків, яка полегшить побудову лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

– імпорт вихідних даних з файлу;

– розрахунок параметрів вихідних даних;

– перевірка даних на викиди та нормальний закон розподілу для значень X та У;

- нормалізація даних за допомогою нормалізуючого претворення на основі десяткового логарифму для значень X та У;
- перевірка нормалізованих даних на нормальній закон розподілу для значень X та У;
- побудова лінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;
- побудова нелінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;
- оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;
- експот результатів у файл.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Обмін інформацією між програмним забезпеченням відбувається за допомогою файлів з довільним доступом, з розширенням .csv.

Вхідні дані: NC, LOC.

Вихідні дані: нормалізовані дані за допомогою нормалізуючого перетворення на основі десяткового логарифму.

3.2 Вимоги до надійності

Програмний продукт повинен функціонувати при безперебійній роботі персонального комп’ютера. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Для забезпечення надійності інформації повинна використовуватися система керування базою даних, що забезпечує цілісність транзакцій і цілісність інформації.

У разі введення користувачем некоректної інформації система повинна повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні комп’ютером. Комп’ютер призначений для роботи в закритому

опалювальному приміщені при наступних умовах навколишнього середовища:

- температура повітря від $+10^0$С до $+30^0$С;
- атмосферний тиск від 630 мм до 800 мм ртутного стовпа;
- відносна вологість повітря не більше 80%;
- запиленість повітря не більше 0,75 мг/м^2.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти мінімальній комплекції:

- Pentium IV – 400 МГц або AMD – 300 МГц;
- ОЗУ – 2 Гб;
- вільної пам’яті на жорсткому диску не менше 5 Гб.

3.5 Вимоги до інформаційної та програмної сумісності

Система також повинна відповідати вимогам до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему не нижче Windows 7 – 32-bit/64-bit.

3.6 Вимоги до маркування та пакування

Програмне забезпечення буде упаковано в електронний архів і мати найменування LOC.rar.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування не висовуються у зв’язку з відсутністю фізичних носів.

4. Вимоги до програмної документації

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

5. Техніко-економічні показники

Не розраховуються в зв’язку з тим, що продукт розробляється в рамках кваліфікаційної (магістерської) роботи.

6. Стадії та етапи розробки

Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Термін виконання робіт	
		Початок етапу	Кінець етапу
1	2	3	4
1. Технічне завдання	1.1Обгрунтування необхідності розробки програми	06.09.20	11.09.20
	1.2 Розробка технічного завдання	11.09.20	16.09.20
	1.3 Затвердження технічного завдання	16.09.20	28.09.20
2. Ескізний проект	2.1 Розробка ескізного проекту	28.09.20	08.10.20
	2.2 Затвердження ескізного проекту	08.10.20	15.10.20
3. Технічний проект	3.1 Розробка технічного проекту	15.10.20	21.10.20
	3.2 Затвердження технічного проекту	21.10.20	01.11.20
4. Робочий проект	Розробка робочого проекту	01.11.20	07.11.20
	Затвердження робочого проекту	07.11.20	15.11.20
	Розробка програмної документації	15.11.20	20.11.20

7. Порядок контролю та прийому

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

Прийом проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б Опис програми

1 Загальні відомості

Програма призначена для побудови лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

2 Функціональне призначення

Функціональним призначенням програми є автоматизація розрахунків, яка полегшить побудову лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

Програма повинна виконувати наступні функції:

– імпорт вихідних даних з файлу;

– розрахунок параметрів вихідних даних;

– перевірка даних на викиди та нормальний закон розподілу для значень X та У;

- нормалізація даних за допомогою нормалізуючого претворення на основі десяткового логарифму для значень X та У;

- перевірка нормалізованих даних на нормальній закон розподілу для значень X та У;

- побудова лінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- побудова нелінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap;

- експот результатів у файл.

3 Опис логічної структури

Програма розроблена на модульній основі, тобто кожний модуль програми відповідає за конкретну дію: імпорт даних, нормалізація., побудова лінійного та нелінійного рівняння регресії та інші.

4 Технічні та програмні засоби

Система повинна задовольняти мінімальній комплекції:

- Pentium IV – 400 МГц або AMD – 300 МГц;
- ОЗУ – 2 Гб;
- вільної пам’яті на жорсткому диску не менше 5 Гб.

Система також повинно відповідати вимоги до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему не нижче Windows 7 – 32-bit/64-bit.

5 Виклик та завантаження

Програма для побудови лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap встановлюється безпосередньо на комп’ютер користувача. Програмне забезпечення відкривається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми». Програма має один режим використання.

6 Вхідні та вихідні дані

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Обмін інформацією між програмним забезпеченням відбувається за допомогою файлів з довільним доступом, з розширенням .csv.

Вхідні дані: NC, LOC.

Вихідні дані: нормалізовані дані за допомогою нормалізуючого перетворення на основі десяткового логарифму.

Додаток В Текст програми

```
package app.controller;

import java.util.List;

import app.framework.Application;
import app.framework.Controller;

import app.model.ImportModel;
import app.model.ImportModel.ImportItem;

import app.view.ImportView;

public final class ImportController extends Controller<ImportModel, ImportView>
{
  /**
   * Initialize a new {@link ImportController} instance for the specified
   * {@link Application}.
   *
   *   @param   application   The   {@link   Application}   that   the   {@link
ImportController}
   *                    is associated with.
   */
  public ImportController(final Application application) {
    super(application);


    this.on("Imports:clear", (data) -> this.clear());
  }

  public void create(final String description) {
    if (description == null) {
      throw new NullPointerException();
    }

    // Trim the description for trailing whitespace.
    String trimmedDescription = description.trim();

    // Check that the trimmed description isn't empty. If the description is
    // indeed empty, throw an exception for the callers to act upon. If the
    // caller is a view, then a suitable error message could be displayed.
    if (trimmedDescription.isEmpty()) {
      throw   new   IllegalArgumentException("Import   descriptions   cannot   be
empty.");
    }

    // Create a new Import item using the now sanitized description.
    ImportItem Import = new ImportItem(trimmedDescription);

    // Update the model with the newly created Import item.
    this.model().add(Import);
  }

  public void complete(final ImportItem Import) {
    if (Import == null) {
      throw new NullPointerException();
    }

    // Remove the Import item from the model.
    this.model().remove(Import);
  }

  public void complete(final List<ImportItem> Imports) {
    if (Imports == null) {
      throw new NullPointerException();
    }
```

```
    for (ImportItem Import: Imports) {
      this.complete(Import);
    }
  }

  public void clear() {
    this.model().clear();
  }



package app.model;

import java.util.Collection;
import java.util.LinkedHashSet;

// Framework
import app.framework.Application;
import app.framework.Model;

/**
 * Example {@link Model} using the MVC micro-framework presented in
 * {@link app.framework}.
Import);
  }

  public void clear() {
    if (this.Imports.isEmpty()) {
      return;
    }

    this.Imports.clear();
    this.emit("Imports:changed");
  }

  public static final class ImportItem {
    private final String description;

    public ImportItem(final String description) {
      if (description == null) {
        throw new NullPointerException();
      }

      this.description = description;
    }

    public String description() {
      return this.description;
    }

    @Override
    public String toString() {
      return this.description;
    }
  }
}



package app.view;

// General utilities
import java.util.List;

// AWT utilities
import java.awt.BorderLayout;
import java.awt.FlowLayout;

// Swing utilities
import javax.swing.BoxLayout;
import javax.swing.JButton;
```

```
import  javax.swing.JList;
import  javax.swing.JPanel;
import  javax.swing.JScrollPane;
import  javax.swing.JTextField;

// Swing borders
import  javax.swing.border.EmptyBorder;

// Framework
import  app.framework.Application;
import  app.framework.View;

// Models
import  app.model.ImportModel;
import  app.model.ImportModel.ImportItem;

// Controllers
import  app.controller.ImportController;

/**
 * The {@link ImportView} class takes care of rendering the view for creating,
 * displaying, and completing Import items.
 */
public final class ImportView extends View<ImportModel, ImportController> {
  /**
   * Initialize a new {@link ImportView} instance for the specified
   * {@link Application}.
   *
   * @param application The {@link Application} that the {@link ImportView} is
   *                    associated with.
   */
  public ImportView(final Application application) {
    super(application);

    // Initialize the model and controller of the view. The order in which these
    // are initialized does not matter as they will automagically be wired
    // together regardless.
    this.model(new  ImportModel(application));
    this.controller(new  ImportController(application));
  }

  /**
   * Render the {@link ImportView}.
   */
  public JPanel render() {
    JPanel viewPanel = new JPanel(new BorderLayout());
    viewPanel.setBorder(new  EmptyBorder(10,  10,  10,  10));

    JPanel inputPanel = new JPanel();
    inputPanel.setLayout(new  BoxLayout(inputPanel,  BoxLayout.LINE_AXIS));
    inputPanel.setBorder(new EmptyBorder(0,  0,  10,  0));
    viewPanel.add(inputPanel,  BorderLayout.NORTH);

    // Create a 10-column text field for inputting Import descriptions.
    JTextField ImportInput = new JTextField(10);
    inputPanel.add(ImportInput,  BorderLayout.NORTH);

    ImportInput.addActionListener(e  -> {
      try {
        // Delegate creation of the Import item to the controller.
        this.controller().create(ImportInput.getText());
      }
      catch (IllegalArgumentException ex) {
        // Simply print out the error message to the error console whenever
        // the user enter invalid input. This should ideally be displayed in a
        // nice looking error dialog of sorts in a real application.
        System.err.println(ex.getMessage());
      }

      // Clear the text input field.
      ImportInput.setText(null);
```

```
            });

            JButton ImportSubmit = new JButton("Create Import");
            inputPanel.add(ImportSubmit, BorderLayout.NORTH);

            // Trigger the action event of the text input field whenever the submit
            // button is pressed. This make pressing the button equivalent to hitting
            // enter when the text input field is focused.
            ImportSubmit.addActionListener(e -> ImportInput.postActionEvent());

            JList<ImportItem> ImportsList = new JList<>(this.model().Imports());

            this.model().on("Imports:changed", (ImportItem Import) -> {
              ImportsList.setListData(this.model().Imports());
            });

            JScrollPane ImportsPane = new JScrollPane(ImportsList);
            viewPanel.add(ImportsPane, BorderLayout.CENTER);

            JPanel actionsPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
            actionsPanel.setBorder(new EmptyBorder(10, 0, 0, 0));
            viewPanel.add(actionsPanel, BorderLayout.SOUTH);

            JButton ImportComplete = new JButton("Complete Import");
            ImportComplete.setEnabled(!ImportsList.isSelectionEmpty());
            actionsPanel.add(ImportComplete);

            ImportComplete.addActionListener(e -> {
              // Get the currently selected Import items.
              List<ImportItem> selectedImports = ImportsList.getSelectedValuesList();

              // Delegate deletion of the Import item to the controller.
              this.controller().complete(selectedImports);
            });

            ImportsList.addListSelectionListener(e -> {
              //  Enable/disable  the  "Complete  Import"  button  whenever  the  list  of
Imports
              // goes in and out of focus.
              ImportComplete.setEnabled(!ImportsList.isSelectionEmpty());
            });

            return viewPanel;
          }
        }



        package app.framework;

        // General utilities
        import java.util.HashSet;
        import java.util.Set;

        // Swing utilities
        import javax.swing.JFrame;

        /**
         * The {@link Application} class describes a complete MVC application.
         *
         * <p>
           * Initialize a new {@link Application} instance.
           */
          public Application() {
            // Construct the main frame of the application.
            JFrame frame = new JFrame();

            // Exit the application when the main frame is closed.
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            // Start the application.
```

```
    this.start(frame);

    // Pack the application frame.
    frame.pack();

    // Make the application frame visible.
    frame.setVisible(true);
  }

  /**
   * Access the {@link Radio} used for {@link Application}-wide communication.
   *
   * <p>
   * This method can only be used within framework classes.
   *
   * @return The {@link Radio} used for {@link Application}-wide communication.
   */
  final Radio radio() {
    return this.radio;
  }

  /**
   * Start the {@link Application}.
   *
   * <p>
   * This method is called as part of the {@link Application} initialization.
   * @param frame The main frame of the {@link Application}.
   */
  protected abstract void start(final JFrame frame);
}
public abstract class Controller<M extends Model, V extends View> {
  /**
   * The {@link Application} that the {@link Controller} is part of.
   */
  private Application application;

  /**
   * The {@link Model} that the {@link Controller} operates on.
   */
  private M model;

  /**
   * The {@link View} that the {@link Controller} operates on.
   */
  private V view;

  /**
   * Initialize a new {@link Controller} instance for the specified
   * {@link Application}.
   *
   * @param application The {@link Application} that the {@link Controller} is
   *                    associated with.
   */
  public Controller(final Application application) {
    if (application == null) {
      throw new IllegalArgumentException(
        "An Application must be specified when initialising a Controller."
      );
    }

    this.application = application;
  }

  /**
   * Access the {@link Application} that the {@link Controller} is associated
   * with.
   *
   * @return The {@link Application} that the {@link Controller} is associated
   *         with.
   */
  protected final Application application() {
```

```
  return this.application;
}

/**
 * Access the {@link Model} that the {@link Controller} operates on.
 *
 * @return The {@link Model} that the {@link Controller} operates on.
 */
protected final M model() {
  return this.model;
}

/**
 * Set the {@link Model} that the {@link Controller} operates on.
 *
 * @param model The {@link Model} that the {@link Controller} operates on.
 */
@SuppressWarnings("unchecked")
final void model(final M model) {
  if (model == null) {
    throw new NullPointerException();
  }

  if (this.model != null) {
    throw new IllegalStateException(
      "A Model has already been set on the Controller."
    );
  }

  this.model = model;

  if (this.view != null) {
    try {
      this.view.model(this.model);
    }
    catch (IllegalStateException ex) {
      ;
    }
  }
}

/**
 * Access the {@link View} that the {@link Controller} operates on.
 *
 * @return The {@link View} that the {@link Controller} operates on.
 */
protected final V view() {
  return this.view;
}

/**
 * Set the {@link View} that the {@link Controller} operates on.
 *
 * @param view  The {@link View} that the {@link Controller} operates on.
 */
@SuppressWarnings("unchecked")
final void view(final V view) {
  if (view == null) {
    throw new NullPointerException();
  }

  if (this.view != null) {
    throw new IllegalStateException(
      "A View has already been set on the Controller."
    );
  }

  this.view = view;

  try {
    this.view.controller(this);
```

```
    }
    catch (IllegalStateException ex) {
      ;
    }

    if (this.model != null) {
      try {
        this.view.model(this.model);
      }
      catch (IllegalStateException ex) {
        ;
      }
    }
  }

  /**
   * Emit an event with the specified name.
   *
   * @param event The name of the event to emit.
   * @return      A boolean indicating whether or not the event was picked up
   *              by a {@link Consumer}.
   */
  protected final boolean emit(final String event) {
    return this.application.radio().emit(event);
  }

  /**
   * Emit an event with the specified name and data.
   *
   * @param <T>       The type of data to utilize in {@link Consumer Consumers}.
   * @param event     The name of the event to emit.
   * @param data      The data to pass on to {@link Consumer Consumers}.
   * @return          A boolean indicating whether or not the event was picked
   *                  up by a {@link Consumer}.
   */
  @SuppressWarnings("unchecked")
  protected final <T extends Object> boolean emit(final String event, final T
data) {
    return this.application.radio().emit(event, data);
  }

  /**
   * Attach a {@link Consumer} to the specified event.
   *
   * @param <T>       The type of data to utilize in {@link Consumer Consumers}.
   * @param event     The name of the event to attach the {@link Consumer} to.
   * @param consumer  The {@link Consumer} to attach to the specified event.
   * @return          A boolean indicating whether or not the operation affected
   *                   the set of {@link Consumer Consumers} attached to the
   *                  specified event.
   */
  @SuppressWarnings("unchecked")
  protected final <T extends Object> boolean on(
    final String event,
    final Consumer<T> consumer
  ) {
    return this.application.radio().on(event, consumer);
  }

  /**
   * Detach a {@link Consumer} from the specified event.
  public Model(final Application application) {
    if (application == null) {
      throw new IllegalArgumentException(
        "An Application must be specified when initialising a Model."
      );
    }

    this.application = application;
  }
```

```
  /**
   * Access the {@link Application} that the {@link Model} is associated with.
   *
   * @return The {@link Application} that the {@link Model} is associated with.
   */
  protected final Application application() {
    return this.application;
  }

  /**
   * Emit an event with the specified name.
   *
   * @param event The name of the event to emit.
   * @return      A boolean indicating whether or not the event was picked up
   *              by a {@link Consumer}.
   */
  protected final boolean emit(final String event) {
    return this.radio.emit(event);
  }

  /**
   * Emit an event with the specified name and data.
   *
   * @param <T>       The type of data to utilize in {@link Consumer Consumers}.
   * @param event     The name of the event to emit.
   * @param data      The data to pass on to {@link Consumer Consumers}.
   * @return          A boolean indicating whether or not the event was picked
   *                  up by a {@link Consumer}.
   */
  @SuppressWarnings("unchecked")
  protected final <T extends Object> boolean emit(
    final String event,
    final T data
  ) {
    return this.radio.emit(event, data);
  }

  /**
   * Attach a {@link Consumer} to the specified event.
   *
   * @param <T>       The type of data to utilize in {@link Consumer Consumers}.
   * @param event     The name of the event to attach the {@link Consumer} to.
   * @param
  @SuppressWarnings("unchecked")
  public final <T extends Object> boolean on(
    final String event,
    final Consumer<T> consumer
  ) {
    return this.radio.on(event, consumer);
  }
   */
  @SuppressWarnings("unchecked")
  public final <T extends Object> boolean off(
    final String event,
    final Consumer<T> consumer
  ) {
    return this.radio.off(event, consumer);
  }
}
```

Додаток Г Інструкція користувача

Для запуску програми потрібно відкрити програму або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми», після чого з’явиться головна форма, яка представлена на рисунку Г.1.

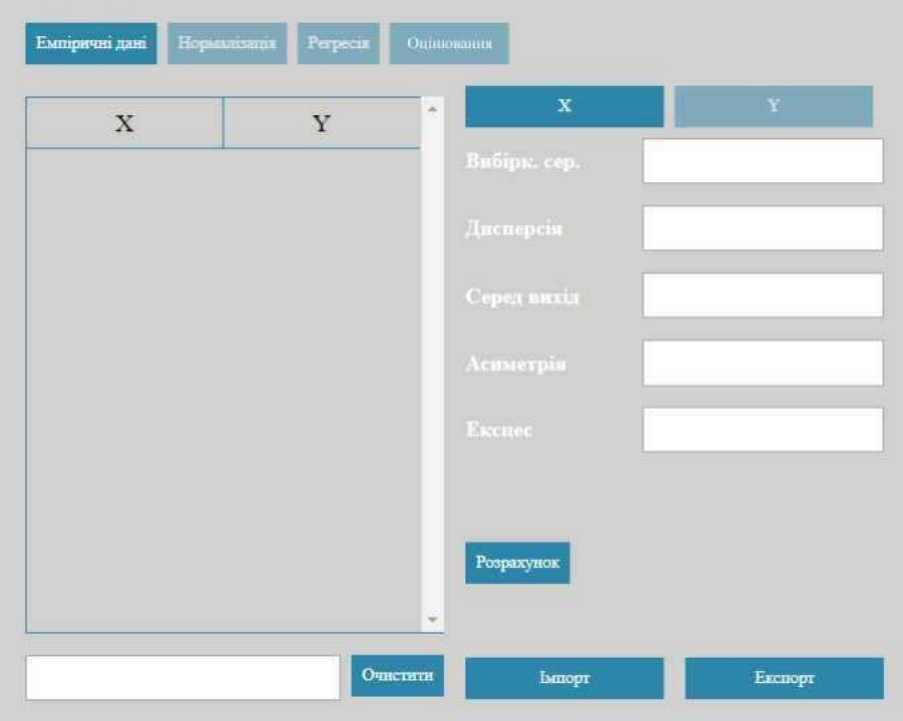

Рисунок Г.1 – Головна форма програми

Для того, щоб імпортувати дані, потрібно натиснути на кнопку «Імпорт» в результаті чого з’явиться діалогове вікно, в якому потрібно вибрати шлях та файл з розширенням .csv. Діалогове вікно зображено на рисунку Г.2.

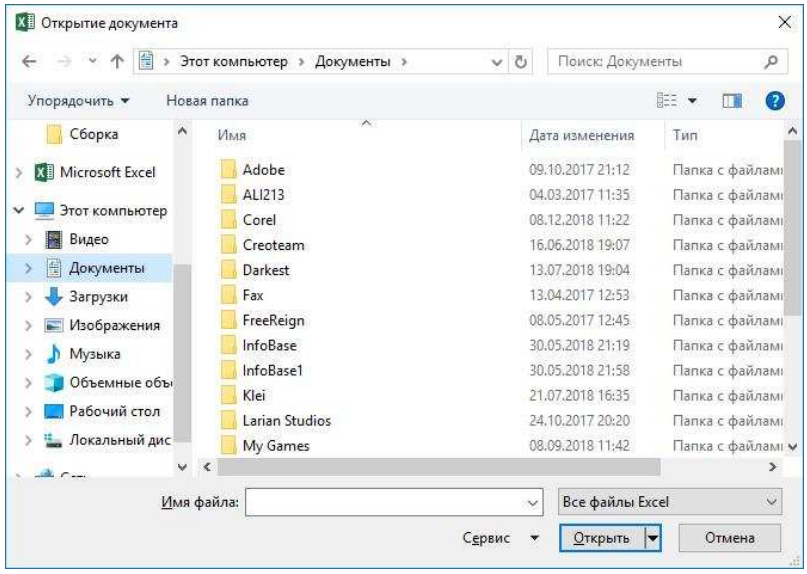

Рисунок Г.2 – Діалогове вікно для імпорту даних

Після вибору файлу для імпорту дані пройдуть обробку і в разі успішної перевірки внесуться до таблиці вихідних даних. Для того, щоб здійснити розрахунки, потрібно натиснути на кнопку «Розрахунок» у вкладці вихідних даних. Після чого програма автоматично розрахує вихідні параметри вибірки та виведе результат. Результат представлено на рисунку Г.3.

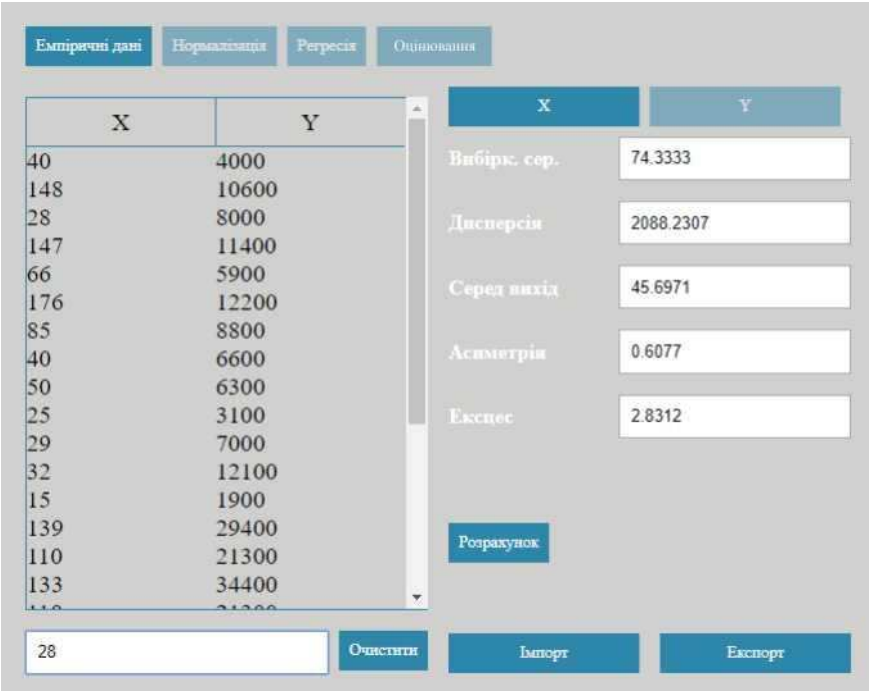

Рисунок Г 3 – Форма результатів розрахунків параметрів вихідних даних

Якщо обрано не вірний файл для імпорту, можна натиснути кнопку «Очистити» та всі дані в програмі буде очищено. Розрахункові дані можна експортувати, для цього потрібно натиснути кнопку «Експорт», в результаті чого з’явиться діалогове вікно, в якому потрібно вибрати місце експорту та ввести ім’я файлу и натиснути кнопку «Зберігти». Діалогове вікно для експорту даних зображено на рис. Г.4.

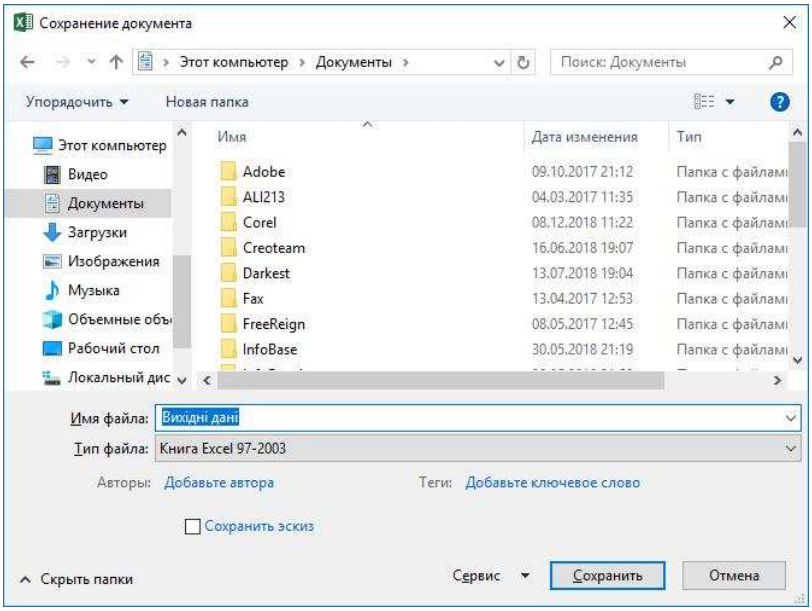

Рисунок Г.4 – Діалогове вікно для експорту даних

Для нормалізації даних потрібно перейти на вкладку «Нормалізація» та натиснути кнопку «Розрахунок», в результаті чого, система розрахує та виведе на екран результати нормалізації, які представлені на рисунку Г.5.

Для побудови рівняння регресії потрібно перейти у вкладку регресія, та натиснути кнопку «Побудувати», в результаті чого буде побудовано лінійне та нелінійне рівняння регресії та для кожного з них побудовано довірчий інтервал та інтервал прогнозування.

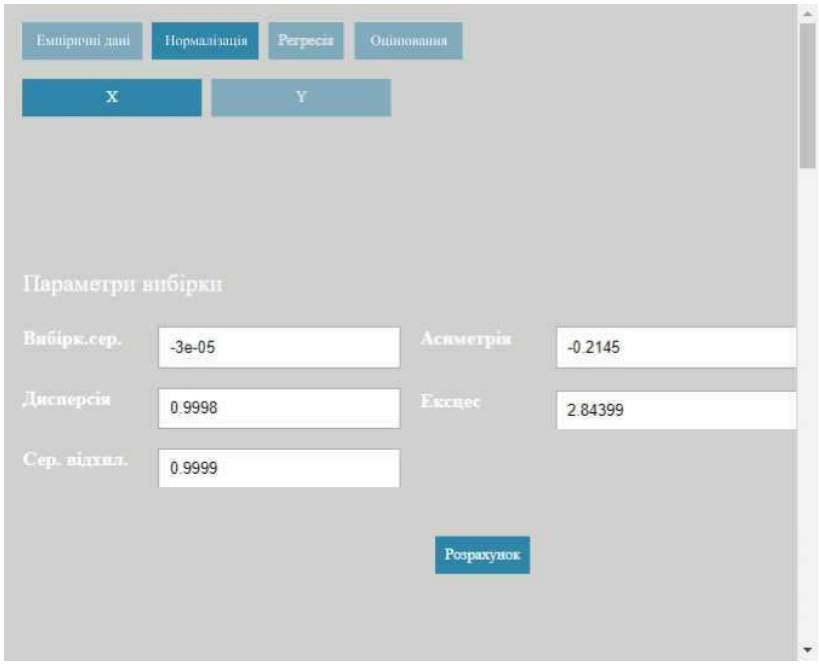

Рисунок Г.5 – Форма нормалізації даних

Для отримання значення оцінки розміру веб-застосунка, розробленого з використанням фреймворку Bootstrap, потрібно перейти у вкладку «Оцінювання» та ввести значення фактора для розрахунку, а саме загальну кількості компонентів NC та натиснути кнопку «Розрахунок». В результаті отримаємо оцінку кількості рядків коду LOC для розроблюваного веб-застосунка. Результат представлено на рисунку Г.6.

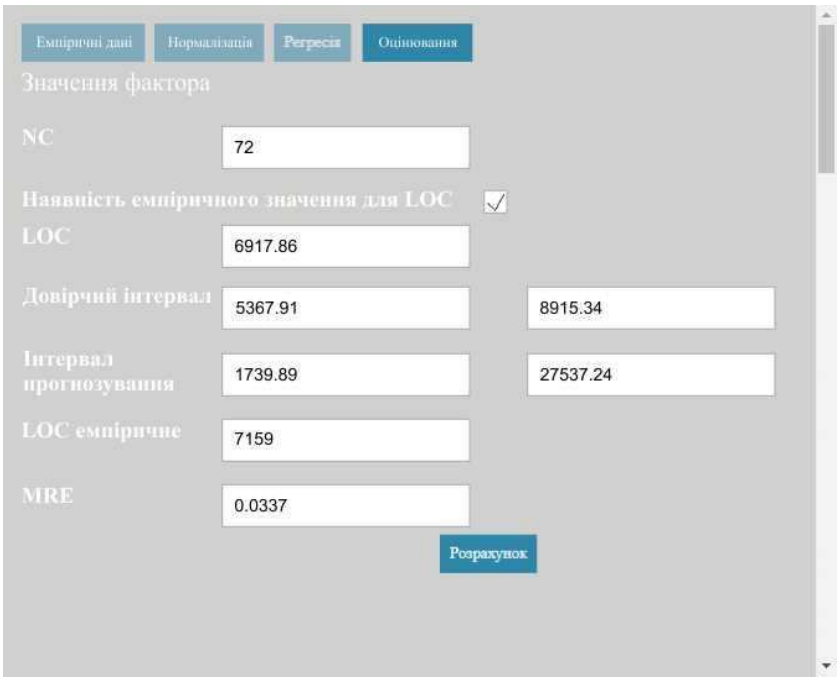

Рисунок Г.6 – Форма оцінювання

Додаток Д Програма та методика випробувань ПЗ

1. Об’єкт випробувань

1.1 Назва об’єкту

Назва розроблюваного проекту: «Програма для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap». Галузь застосування – підприємства, які спеціалізуються на розробці ПЗ веб-застосунків.

Коротка назва: програма.

1.2 Область застосування

Функціональним призначенням програми є автоматизація розрахунків, яка полегшить побудову лінійної та нелійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для оцінювання розміру веб-застосунків, розроблених з використанням фреймворку Bootstrap.

2 Мета випробувань

Мета проведення випробувань – оцінка експлуатаційних характеристик програми, перевірка і підтвердження працездатності програми в умовах, максимально наближених до умов реальної експлуатації.

3 Вимоги до програми

Програма має реалізовувати функції, описані в технічному завданні, яке наведено в додатку А.

4 Вимоги до програмної документації

Для проведення тестування програми повинна бути надана наступна документація:

- технічне завдання на розробку програми;
- текст програми;
- опис програми;
- інструкція користувача;
- програма і методика випробувань ПЗ.

5 Склад, порядок та методи випробувань

5.1 Перевірка програми на відповідність технічному завданню:

– перевірка роботи функції «Імпорт даних»;

– перевірка роботи функції «Нормалізіція даних»;

– перевірка роботи функції «Перевірка на нормальний закон розподілу»;

– перевірка роботи функції «Експорт даних у файл».

5.2 Перевірка програми на її програмно-апаратну сумісність: тестування на апаратних-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені в технічному завданні; перевірка на наявність і усунення всіх помилок, зазначених у п. 5.1.

5.3 Візуальний перегляд програми: остаточне налагодження на предмет виявлення та усунення дрібних помилок.