МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2021 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: «Математична модель для оцінювання якості програмного забезпечення розпізнавання номерних знаків та розробка програми для її реалізації»

Виконав: студент групи 6151м
_____ Полтєв О.С.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.т.н., професор НУК
(посада, науковий ступень, вчене звання)
_____ Фаріонова Т.А.
(підпис, ПІБ)

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«___» _____ 2021 р.

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

Студенту Полтєву Олександру Сергійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Математична модель для оцінювання якості програмного забезпечення розпізнавання номерних знаків та розробка програми для її реалізації»
Керівник роботи доцент кафедри ПЗАС, к.т.н., професор НУК Фаріонова Т.А.
Затверджені наказом ректора № 1239уч від «13» 10 2021 р.
2. Термін подання роботи: 06.12.2021 р.
3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Проект програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 13.10.2021 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	18.10.2021	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	19.10.2021	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	22.10.2021	НС*
4.	Підготовка розділу з проекту програмного забезпечення	16.11.2021	
5.	Підготовка організаційно-економічного розділу	19.11.2021	
6.	Підготовка розділу з охорони праці	22.11.2021	
7.	Підготовка розділу з охорони навколишнього середовища	24.11.2021	
8.	Підготовка розділу Висновки	25.11.2021	
9.	Оформлення списку використаних джерел та додатків	29.11.2021	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	06.12.2021	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
11.	Підготовка презентації та доповіді	09.12.2021	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	10.12.2021	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	20.12.2021	

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

Студент _____ Полтєв О.С.
(підпис) (ПІБ)

Керівник роботи _____ Фаріонова Т.А.
(підпис) (ПІБ)

РЕФЕРАТ

Полтєв Олександр Сергійович

«Математична модель для оцінювання якості програмного забезпечення розпізнавання номерних знаків 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, 2021 р.

Обсяг роботи: 111 стор., 24 табл., 23 рис., 57 використаних джерел, 5 додатків.

Актуальність теми роботи: визначається необхідністю підвищення достовірності оцінювання якості ПЗ з розпізнавання номерних знаків автомобілів.

Мета та завдання дослідження: метою роботи є підвищення достовірності оцінювання якості ПЗ з розпізнавання номерних знаків автомобілів.

Об'єкт дослідження: процес оцінювання якості ПЗ розпізнавання номерних знаків.

Предмет дослідження: математична модель для оцінювання якості ПЗ розпізнавання номерних знаків.

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

Наукова новизна одержаних результатів: полягає в удосконаленні математичної моделі для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання якості ПЗ розпізнавання номерних знаків в порівнянні з існуючою моделлю.

Практичне значення одержаних результатів: полягає в розробці програми для оцінювання якості ПЗ розпізнавання номерних знаків на основі удосконаленої математичної моделі.

Апробація результатів досліджень: основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на IV Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2021 р.).

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

Ключові слова: нелінійна регресійна модель, однофакторна регресія, нормалізуюче перетворення на основі десяткового логарифму, розробка програмного забезпечення, якість програмного забезпечення.

ABSTRACT

Poltiev Oleksandr Serhiiovych

«Mathematical model of quality assessment of software for license plate recognition and development of 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, 2021.

Volume of work: 111 pages of typewritten text, 24 tables, 23 figures, a list of references from 57 names, 5 appendices.

Relevance of the theme: determined by the need to increase the reliability of estimating of the quality of software for license plate recognition of a cars.

The goal and objectives of the study: the purpose of this work is to improve the reliability of estimating the quality of software for license plate recognition of a cars.

The object of the study: the process of estimating the quality of software for license plate recognition.

The subject of the study mathematical model for estimating the quality of software for license plate recognition.

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

The scientific novelty of the obtained results: is to improve the mathematical model for estimating the quality of software for license plate recognition, using the normalizing transformation based on the decimal logarithm, which increased the reliability of estimating the quality of software for license plate recognition, compared to existing model.

The practical significance of the obtained results: software development for estimating the quality of software for license plate recognition, based on the improved mathematical model.

Approbation of research results: the main provisions and research results presented in the qualification work have been approbat at the IV All-Ukrainian scientific-practical Internet conference of students, graduate students and young scientists on "Modern computer systems and networks in management" (Kherson, November 30, 2021).

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

Keywords: nonlinear regression model, univariate regression, normalizing transformation based on the decimal logarithm, software development, software quality.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ВВ	Випадкова величина
ЕД	Емпіричні дані
МНК	Метод найменших квадратів
ПЗ	Програмне забезпечення
CSV	Comma Separated Values
LPR	License Plate Recognition
OpenCV	Open Source Computer Vision Library
UML	Unified Modelling Language

ВСТУП

Актуальність теми. Сьогодні технологія розпізнавання номерних знаків автомобілів як ніколи є актуальною на тлі постійного збільшення автомобільного потоку в усьому світі. Ідентифікація автомобіля за реєстраційним номерним знаком є важливим аспектом контролю та забезпечення як безпеки дорожнього руху, так і функціонування підприємств у найрізноманітніших сферах: від автотранспортних підприємств та автомобільних стоянок до пунктів контролю в'їзду-виїзду на територію об'єктів [1].

Розпізнавання номерних знаків автомобілів (LPR) є складним завданням з технічного погляду та штучного інтелекту і є програмно-апаратним комплексом, який реалізує алгоритми автоматичного розпізнавання номерних знаків для реєстрації подій, пов'язаних з переміщенням автомобілів [2]. Використовувані технології найчастіше є комерційною таємницею компаній-розробників.

Існує багато LPR-систем з різним рівнем якості розпізнавання, швидкістю дії та набором додаткових функцій. Чим вище швидкість розпізнавання та його точність, тим дорожче відповідна система. Більшість існуючих на ринку систем розпізнавання номерних знаків автомобілів розраховані на корпоративних користувачів. Але також існують на ринку готові програмні рішення, які надають можливість малому та середньому бізнесу отримати таку систему за невеликі гроші та забезпечити своє підприємство інструментом контролю та обліку.

ПЗ для розпізнавання номерних знаків автомобілів дуже затребуване. Існує багато відповідного ПЗ, зокрема, система розпізнавання автомобільних номерів "NumberOK" [3], система розпізнавання номерних знаків "Орион Авто" [4], безкоштовний мобільний застосунок для розпізнавання автомобільних номерів "Smart Engines" з технологією Smart PlateReader [5] та

інші. Деякі компанії розробляють власне ПЗ для розпізнавання номерних знаків автомобілів [6], зокрема, інформація про такі розробки представлена як на сайті спільноти ІТ-фахівців Хабр [7, 8], так і в репозитарії GitHub [9]. До того ж, МВС України відкрило доступ до реєстру транспортних засобів [10]. Тепер за номерним знаком стало можливим перевіряти деяку інформацію про автомобіль, наприклад, марку, модель, рік випуску, колір і т.д., що відкриває великі можливості для інтеграції ПЗ та спільної обробки різних масивів інформації.

При розробці ПЗ виникає питання оцінювання якості ПЗ, для чого використовують метрики якості ПЗ – підмножину метрик ПЗ, які фокусуються на аспектах якості продукту, процесу та проекту, серед яких можна виділити щільність дефектів, тобто кількість дефектів відносно розміру ПЗ, вираженого у вигляді кількості рядків коду або функціональних точок. Цей показник використовується у багатьох комерційних системах ПЗ [11].

Існує ціла низка стандартів, методологій та моделей якості в галузі інженерії ПЗ [12–17], найбільш вдалі та змістовні з яких SQuaRE (Software Quality Requirements and Evaluation), CapabilityMaturityModel (CMM) та ISO/IEC 15504 (SPICE). Питанням оцінювання якості ПЗ присвячено багато публікацій в періодичних виданнях, зокрема, [18–20]. Крім того, для оцінювання кількості дефектів ПЗ використовуються також моделі з застосуванням регресійного аналізу та нормалізуючих перетворень випадкових величин [21].

Мета і завдання дослідження. Метою дослідження є підвищення достовірності оцінювання якості ПЗ з розпізнавання номерних знаків автомобілів. Для досягнення поставленої мети треба вирішити такі завдання:

– провести аналіз сучасних методів та алгоритмів, математичних моделей та існуючого ПЗ для розпізнавання номерних знаків автомобілів;

– провести аналіз існуючих метрик та моделей для оцінювання якості ПЗ;

– обґрунтувати необхідність удосконалення математичної моделі для оцінювання якості ПЗ для розпізнавання номерних знаків;

– побудувати удосконалену математичну модель для оцінювання якості ПЗ для розпізнавання номерних знаків за рахунок використання нормалізуючого перетворення;

– розробити програму для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів, використовуючи побудовану математичну модель.

Об’єктом дослідження є процес оцінювання якості ПЗ розпізнавання номерних знаків.

Предметом дослідження є математична модель для оцінювання якості ПЗ розпізнавання номерних знаків.

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

Наукова новизна одержаних результатів полягає в удосконаленні математичної моделі для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання якості ПЗ розпізнавання номерних знаків в порівнянні з існуючою моделлю.

Практичне значення отриманих результатів полягає в розробці програми для оцінювання якості ПЗ розпізнавання номерних знаків на основі удосконаленої математичної моделі.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві [22], здобувачеві належить: удосконалення математичної моделі для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів.

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

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

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

1.1 Методи, алгоритми та моделі для розпізнавання номерних знаків автомобілів

Методи розпізнавання номерних знаків автомобілів, як окремий випадок розпізнавання зображень, можна розділити на три групи методів.

Перша група – це попередня фільтрація та підготовка зображення. Друга група – це логічна обробка результатів фільтрації. Третя група – це алгоритми прийняття рішень на основі логічної обробки. Межі між цими групами дуже умовні [23].

Перша група – попередня фільтрація. До цієї групи відносяться методи, які дозволяють виділити на зображеннях області, що цікавлять, без їх аналізу. Більшість цих методів застосовує якесь єдине перетворення до всіх точок зображення. На рівні фільтрації аналіз зображення не проводиться, але точки, що проходять фільтрацію, можна розглядати як області з особливими характеристиками.

Найпростіше перетворення – це бінаризація зображення по порогу. Для RGB зображення та зображення в градаціях сірого порогом є значення кольору. Вибір порога, по якому відбувається бінаризація, багато в чому визначає процес бінаризації. Зазвичай бінаризація здійснюється за допомогою алгоритму, що адаптивно вибирає поріг. Таким алгоритмом може бути вибір математичного очікування або моди, зображення може бути бінаризоване за середнім кольором або за найбільшим піком гістограми.

Окремий клас фільтрів – фільтрація меж та контурів. Контури дуже корисні, коли потрібно перейти від роботи із зображенням до роботи з об'єктами на цьому зображенні. Коли об'єкт досить складний, але добре

виділяється, часто єдиним способом роботи з ним є виділення його контурів. Існує цілий ряд алгоритмів, що вирішують завдання фільтрації контурів. Це оператор Кенні, оператор Собеля, оператор Лапласа, оператор Прюїтта, оператор Робертса [23]. Найчастіше використовується саме оператор Кенні, який добре працює та реалізація якого є в OpenCV [24].

Друга група – логічна обробка результатів фільтрації. Фільтрування дає набір придатних для обробки даних. Але часто не можна просто взяти та використовувати ці дані без їх обробки. Далі використовуються класичні методи, що дозволяють перейти від зображення до властивостей об'єктів, або самих об'єктів. Це контурний аналіз.

Особливі точки – це унікальні характеристики об'єкта, які дозволяють зіставляти об'єкт сам із собою або зі схожими класами об'єктів. Існує кілька десятків способів, що дозволяють виділити такі точки. Деякі способи виділяють особливі точки в сусідніх кадрах, деякі через великий проміжок часу та при зміні освітлення, деякі дозволяють знайти особливі точки, які залишаються такими навіть при поворотах об'єкта [25].

Третя група – це методи, які не працюють безпосередньо із зображенням, але які дозволяють приймати рішення. В основному це різні методи машинного навчання та прийняття рішень. У 80% ситуацій суть навчання у завданні розпізнавання в наступному. Є тестова вибірка, на якій є кілька класів об'єктів. Алгоритм навчання повинен побудувати таку модель, за якою він зможе проаналізувати нове зображення і прийняти рішення, який об'єкт є на зображенні. У випадку, коли вимірів більше двох, можна використовувати такі алгоритми [26]:

– k-means – один із найпростіших алгоритмів навчання. Звичайно, він переважно для кластеризації, але й навчити через нього теж можна. Працює в ситуації, коли групи об'єктів мають непогано рознесений центр мас і не мають великого перетину [27].

– AdaBoost (Adaptive Boosting) – один із найпоширеніших класифікаторів. Наприклад, каскад Хаара збудований саме на ньому. Зазвичай

використовують, коли потрібна бінарна класифікація, але нічого не заважає навчити на більшу кількість класів [28].

– SVM – один із найпотужніших класифікаторів, що має безліч реалізацій. Вважається досить швидким, але його навчання складніше, ніж у AdaBoost, і потрібний вибір правильного ядра [29].

Ще одним алгоритмом, що використовується, є гістограмний аналіз регіонів. Метод заснований на тому, що символи на номері, фон та рамка контрастні, тобто після виділення меж для зображення будується гістограма. Номер перебуває там, де фіксується максимум білих точок, тобто фіксується глобальний максимум. Далі, знаючи пропорції номера, можна зробити висновок про довжину рамки. Після цього проводиться пошук кандидата на номер – вертикальна лінія, після якої йдуть часті зміни яскравості. Після цього одержаний прямокутник передається на наступний етап. Даний алгоритм нестійкий до шумів та бруду. Крім того, у випадку великої перспективи (великого кута нахилу камери під час зйомки) або маленького зображення машини в порівнянні з навколишніми об'єктами, виділення максимуму проекції буде утруднено і неможливо в принципі. Даний алгоритм є простим у розумінні та реалізації, але працює лише у разі фотографування машини як головного об'єкта фотографії, без захоплення сторонніх предметів та під прямим кутом. В реальних умовах цей алгоритм використовуватися не може.

Метод Віоли-Джонса. У 2001 році Паул Віола і Майкл Джонс запропонували алгоритм для знаходження обличчь з використанням примітивів Хаара [30]. У класичному методі Віоли-Джонса застосовуються ознаки прямокутної форми, зображені на рис. 1.1.

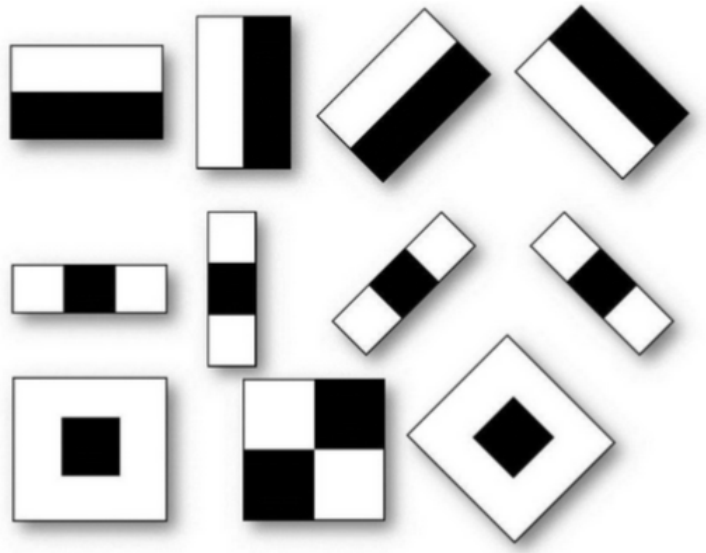

Рисунок 1.1 – Примітиви Хаара

У доповненому методі Віоли-Джонса, який використовується в бібліотеках OpenCV, застосовуються ознаки, доповнені наступними примітивами, зображеними на рис. 1.2 [31]:

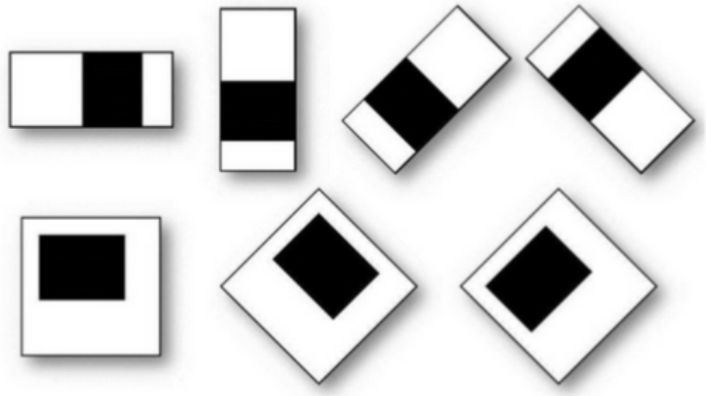

Рисунок 1.2 – Примітиви Хаара, які використовуються у розширеному методі

Для обчислення значення ознаки такого типу застосовується формула $F=X-Y$, де X – сумарне значення яскравостей точок закритих світлою частиною ознаки, а Y – сумарне значення яскравостей точок закритих темною частиною ознаки. Примітиви Хаара дають точкове значення перепаду яскравості по осі X та Y відповідно [32].

Суть даного алгоритму у тому, що по зображенню рухається вікно, до якого прикладаються дані примітиви по черзі. У разі знаходження ознак, що підходять під умову завдання, це вікно фіксується [33]. Навчений каскад Хаара для розпізнавання автомобільних номерів вже доданий до бібліотеки OpenCV. Як алгоритм пошуку номерів, дає результати близько 90% вірного визначення.

Нейромережевий підхід. За допомогою продукту компанії NVIDIA Deep Learning GPU Training System (DIGITS) [34] дослідники можуть за допомогою нативного графічного інтерфейсу вирішувати проблеми підготовки даних, створення згорткової мережі, паралельного навчання кількох моделей, спостереження за процесом навчання в реальному часі, а також вибору кращої моделі . Повністю інтерактивний інструмент DIGITS позбавляє програмування та налагодження. У версії DIGITS 4 реалізується новий підхід до завдання знаходження об'єктів, який дозволяє навчати нейронні мережі для знаходження об'єктів, таких, як обличчя, транспортні засоби або пішоходи на зображеннях та визначати обмежуючі прямокутники (bounding boxes) навколо об'єктів.

У мережі Інтернет існує достатня кількість ресурсів з відкритим доступом для професійної спільноти розробників систем технічного, комп'ютерного та машинного зору, обробки та аналізу зображень, наприклад, [35].

Крім того, існує бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV [24], яка включає алгоритми комп'ютерного зору, обробки зображень та чисельні алгоритми загального призначення, реалізована для C/C++, Python, Java, Ruby, Matlab, Lua та інших мов , яка може вільно використовуватися в академічних та комерційних цілях (розповсюджується в умовах ліцензії BSD).

1.2 Аналіз існуючого програмного забезпечення для розпізнавання номерних знаків автомобілів

ПЗ для розпізнавання номерних знаків автомобілів дуже затребуване у бізнес-середовищі. Існує багато відповідного ПЗ, орієнтованого на різні потреби споживачів. Розглянемо деякі з цього ПЗ.

Система розпізнавання автомобільних номерів "NumberOK" [3]. ПЗ "NumberOk" розпізнає номерні знаки, марку, модель, тип і колір автомобілів та керує виконавчими механізмами (шлагбаумами, автоматичними воротами). Працює з будь-якими IP камерами та аналоговими відеореєстраторами за RTSP протоколом. ПЗ "NumberOk" розпізнає шаблони номерних знаків всіх регіонів світу (США, Європа, Азія, Африка, Океанія). В основі ПЗ "NumberOk" та LPR SDK – технологія власної розробки Automatic Number Plate Recognition. Існуючі варіанти програми "NumberOK" зображені на рис. 1.3.

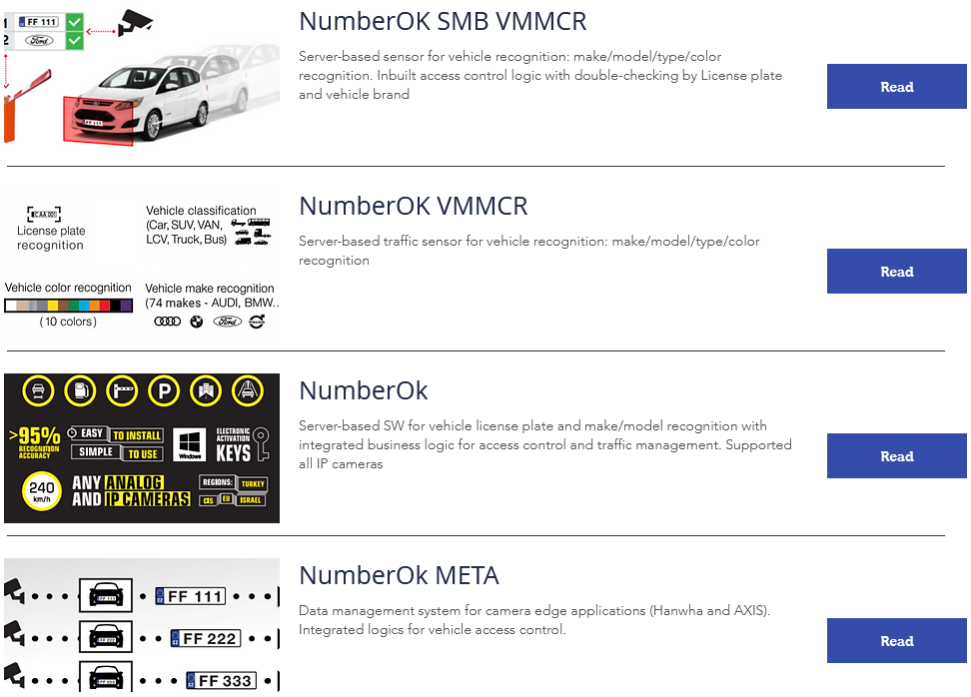

Рисунок 1.3 – Існуючі варіанти програми "NumberOK"

Система розпізнавання номерних знаків "Орион Авто" [4]. ПЗ "Авто Оріон Про" для організації системи розпізнавання автомобільних номерів працює тільки спільно з електронним ключем захисту Guardant, що підключається до USB-порту комп'ютера, у складі АРМ "Оріон Про". ПЗ "Оріон-Авто" має повноцінну сервер-клієнт ієрархію побудови системи, вся БД (автомобільних ДРЗ, подій, фотографій транспортних засобів тощо) може зберігатися на будь-якому терміналі чи іншому зовнішньому сервері, де встановлено сервер БД. Весь комплект ПЗ може бути встановлений також на одному робочому місці. Кількість камер, що підключаються до модуля розпізнавання, обмежена 64. Ручного коригування автомобільного номера оператором немає. Можливість реагування системи на номер або групу номерів теж не реалізована, інформація про розпізнавання обмежена лише збігом з базою номерів (знайдено в базі або не знайдено в базі). Умови розпізнавання ПЗ "Оріон Авто" зображені на рис. 1.4.

Рисунок 1.4 – Умови розпізнавання ПЗ "Оріон Авто"

"Smart Engines" [5] – безкоштовний мобільний застосунок з технологією Smart PlateReader для розпізнавання автомобільних номерів. Він

відноситься до класу рішень ANPR (Automatic number plate recognition) і дозволяє розпізнавати в відеопотоці номера на рухомих і припаркованих автомобілях в режимі реального часу. Область застосування включає страхові компанії, соціальні і державні сервіси фіксації порушень правил паркування та інших правил дорожнього руху, організації каршерінга та ін. Застосунок Smart Engines, представлений у Play Market, наведено на рис. 1.5.

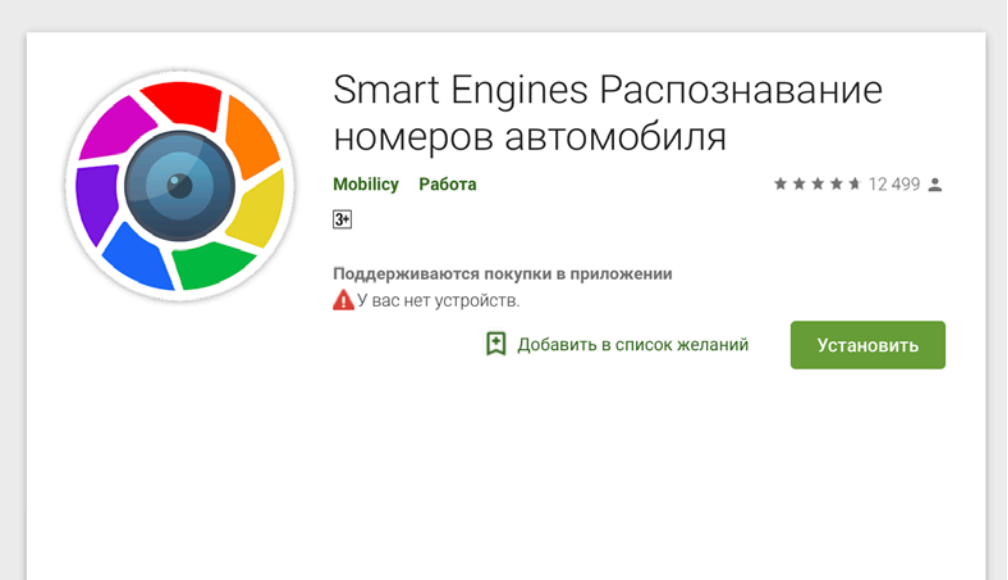

Рисунок 1.5 – Застосунок "Smart Engines" у Play Market.

Високу швидкість і якість розпізнавання в Smart PlateReader забезпечують використовувані надшвидкі нейронні мережі. Глибока алгоритмічна оптимізація під мобільні платформи дозволяє виконувати всі обчислення на смартфоні або планшеті в режимі реального часу. Smart PlateReader використовує тільки оперативну пам'ять пристрою, не створює копій і не передає дані на обробку через інтернет в хмару або на виділений зовнішній сервер.

Деякі компанії розробляють власне ПЗ для розпізнавання номерних знаків автомобілів. Зокрема, ПЗ "Nomeroff Net" [6], яке навіть представлена у репозитарії GitHub [9].

2 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Метрики для оцінювання якості програмного забезпечення

Процес створення ПЗ настільки складний та багатогранний одночасно, що навіть будь-які недоліки процесу розробки можуть сказатися на якості ПЗ, і як наслідок, на його надійності. Основу будь-якої системи вимірювання складають окремі показники, які називаються метриками. Метрики ПЗ є в тому числі, і засобом для вимірювання якості ПЗ. Стандарт [36] регламентує зовнішні і внутрішні характеристики якості ПЗ. Зовнішні характеристики відображають вимоги до функціонування програмного продукту. Для кількісного встановлення критеріїв якості, за якими буде здійснюватися перевірка і підтвердження відповідності ПЗ заданим вимогам, визначаються відповідні зовнішні вимірювані властивості (зовнішні атрибути) ПЗ, метрики (наприклад, час виконання окремих компонентів), діапазони зміни значень і моделі їх оцінки. Метрики, які використовуються на стадії тестування або функціонування, називаються зовнішніми метриками. Вони являють собою моделі оцінки атрибутів.

Метрика ПЗ (Software metric) – захід, що дозволяє отримати чисельне значення деякої властивості ПЗ або його специфікацій [18].

Розглянемо одні з найважливіших метрик для оцінювання якості ПЗ. Це будуть такі метрики, які дозволять охопити загальну картину того, що відбувається на проекті з точки зору якості та визначити кроки щодо її покращення. Метрики стосуватимуться 5 різних областей: вимоги, якість ПЗ, ефективність команди тестування, якість роботи QA та зворотний зв'язок [36].

У контексті оцінки якості ПЗ, що розробляється, буде цікава група 2: "Якість розроблюваного продукту", яка демонструє якість ПЗ, а також і якість

самої розробки. Ця група містить 5 метрик: щільність дефектів, коефіцієнт регресії, коефіцієнт повторно відкритих дефектів, середня вартість виправлення дефекту, кількість дефектів у коді конкретного розробника. Слід зазначити, що всі наведені метрики спираються на один показник – кількість дефектів (у всьому ПЗ, в окремому модулі, в старому функціоналі, повторно виявлених і т.і.).

Розглянемо метрику "Щільність дефектів".

1. Щільність дефектів = кількість дефектів в окремому модулі / загальна кількість дефектів у ПЗ.

Обчислюється частка дефектів, що припадає на окремий модуль протягом ітерації чи релізу. Призначення метрики: підсвітити, яка частина є найбільш проблемною. Ця інформація допоможе при оцінці та плануванні робіт з даним модулем, а також при аналізі ризиків.

Причини великої кількості дефектів у одному конкретному модулі (коефіцієнт більше 0,3) можуть бути різні: неякісні вимоги, кваліфікація розробника, технічна складність і т.д. У будь-якому випадку ця метрика відразу зверне увагу на проблемну зону.

Якщо розглядати метрику "Щільність дефектів", можна побачити і таке її визначення: "Вона вимірює дефекти відносно розміру ПЗ, вираженого у вигляді рядків коду або функціональних точок" [11], тобто вимірює якість коду на одиницю. Цей показник використовується в багатьох комерційних системах ПЗ.

Кількість рядків коду (Source lines of code) – це розмірно орієнтована метрика ПЗ, в якій обсяг ПЗ розраховується, виходячи з кількості рядків в тексті вихідного коду [37].

Серед метрик кількості рядків коду є дві основні:

– кількість фізичних рядків коду (LOC) – визначається як загальна кількість рядків вихідного коду, включаючи коментарі і порожні рядки;

– кількість логічних рядків коду (LLOC) – визначається як загальна кількість команд і залежить від використовуваної мови програмування.

У данної метрики є ряд переваг, наприклад, дані щодо кількості рядків коду в завершених проектах або модулях програм можуть бути легко зібрані за допомогою інтегрованих середовищ розробки (IDE) або спеціальних програм. Недоліком є те, що метрика залежить від обраної мови програмування та середовища розробки.

Функціональні точки – стандартна метрика вимірювання розміру програмного продукту з погляду користувачів системи. Метод розроблений Аланом Альбрехтом (Alan Albrecht) у середині 70-х. Метод було вперше опубліковано у 1979 році. У 1986 році було сформовано Міжнародну Асоціацію Користувачів Функціональних Точок (International Function Point User Group – IFPUG), яка опублікувала кілька ревізій методу [38].

Метод призначений для оцінки розміру ПЗ на основі логічної моделі обсягу програмного продукту кількістю функціоналу, затребуваного замовником і поставленого розробником. Перевагою методу є те, що виміри не залежать від технологічної платформи, на якій буде розроблятися продукт, і мови програмування, що використовується.

2.2 Аналіз існуючих моделей для оцінювання якості програмного забезпечення

Для кількісного оцінювання показників якості ПЗ використовують різноманітні моделі, під якими розуміють моделі, побудовані для оцінювання залежності якості від заздалегідь відомих або визначених у ході роботи параметрів. Такі моделі можна розділити на дві основні групи: аналітичні та емпіричні.

Аналітичні моделі в свою чергу поділяються на статичні та динамічні. Статичні моделі відрізняються від динамічних насамперед тим, що в них не

враховується час появи помилок. До статичних моделей відносяться модель Міллса, модель Нельсона, модель Коркорена.

Серед динамічних моделей можна виділити безперервні та дискретні. При використанні безперервної динамічної моделі передбачається, що функціонування ПЗ описується набором послідовних станів, перехід між якими відбувається у випадку виникнення відмови. До безперервної динамічної моделі відносяться модель Джелінского – Моранді і модель перехідних ймовірностей Маркова. У дискретних моделях передбачається, що спочатку проводиться тестування ПЗ, можливо, у кілька етапів. У разі появи відмов шукаються і виправляються всі помилки, через які сталися відмови. Після цього починається період експлуатації ПЗ. Прикладами дискретних моделей можуть служити модель Шумана, модель Мусса.

Емпіричні моделі засновані на аналізі накопиченої інформації про функціонування раніше розробленого ПЗ. Найбільш проста емпірична модель пов'язує кількість помилок ПЗ з його обсягом [39].

Оскільки між якістю та складністю ПЗ існує зв'язок, то слід відмітити ще одну групу моделей – моделі, призначені для оцінювання складності ПЗ. Всі існуючі моделі складності визначають тільки окремі, приватні характеристики складного ПЗ. Загальним поняттям для всіх видів складного ПЗ є його структура.

Існують також інші моделі, які в тому чи іншому вигляді дають оцінки якості. Вони використовують будь-яку додаткову інформацію про процес розробки ПЗ, наприклад, дані про покриття коду тестами або інформацію про організацію процесу розробки ПЗ. Однак можливість застосування таких моделей часто обмежена відсутністю цієї додаткової інформації, через що вони знаходять мало застосування на практиці.

Крім того, гарні результати для оцінювання кількості дефектів ПЗ дають також моделі із застосуванням як регресійного аналізу [40, 41], так і нормалізуючих перетворень випадкових величин [21].

2.3 Удосконалення математичної моделі для оцінювання якості програмного забезпечення

Якщо ЕД для оцінювання якості ПЗ не відповідають нормальному закону розподілу, потрібно їх нормалізувати. Для нормалізації таких даних можуть бути використані різні нормалізуючі перетворення, зокрема перетворення на основі десяткового логарифму. На практиці це перетворення досить просте для виконання розрахунків та дає гарні результати:

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

де z –нормально розподілена ВВ,

x – ВВ, яка нормалізується.

Перетворення (2.1) має зворотне перетворення:

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

Лінійна регресійна модель у разі нормально розподілених ВВ може бути представлена наступним чином:

$$z_y = b_1 z_x + b_0 + \varepsilon, \tag{2.3}$$

де b_1 та b_0 – коефіцієнти лінійного рівняння регресії,

ε – випадкова помилка, яка розподілена за нормальним законом, ε~N(0,1).

Коефіцієнти b_1 та b_0 знаходяться методом найменших квадратів:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - \left(\sum_{i=1}^{n} z_{xi}\right)^2}, \tag{2.4}$$

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

Для підвищення достовірності оцінювання можна побудувати довірчий інтервал та інтервал прогнозування лінійного рівняння регресії.

Довірчий інтервал лінійного рівняння регресії може бути представлений наступним чином:

$$y = \hat{y} \pm t(_{\alpha/2,n-2}) \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}, \tag{2.6}$$

де $\hat{y}$ – розрахункове значення y за рівнянням лінійної регресії,

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

α – рівень значущості,

n – кількість значень ВВ у вибірці,

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

Інтервал прогнозування лінійного рівняння регресії може бути представлений наступним чином:

$$y = \hat{y} \pm t_{(\alpha/2,n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}, \tag{2.7}$$

де $\hat{y}$ – розрахункове значення y за рівнянням лінійної регресії,

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

α – рівень значущості,

n – кількість значень ВВ у вибірці,

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

Нелінійна регресійна модель у загальному вигляді може бути представлена наступним чином:

$$y = \overline{y} + \varepsilon_t = f(x) + \varepsilon_t, \tag{2.8}$$

де y – залежна змінна,

$f(x)$ – функція, яка визначає вид регресійної моделі,

x – незалежна змінна,

ε – випадкова помилка, яка розподілена за нормальним законом, $\varepsilon \sim N(0,1)$.

У разі застосування нормалізуючого перетворення на основі десяткового логарифму (2.1) та перетворення, зворотного до нього (2.2), отримаємо нелінійну регресійну модель, яка може бути представлена наступним чином:

$$y = x^{b_1} \cdot 10^{b_0 + \varepsilon}. \tag{2.9}$$

Далі для нелінійного рівняння регресії і можна побудувати довірчий інтервал та інтервал прогнозування [42]. Таким чином, використовуючи нормалізуюче перетворення на основі десяткового логарифму та інтервальні оцінки, отримаємо більш достовірну оцінку кількості дефектів ПЗ в порівнянні з існуючими методами.

2.4 Попередня обробка вихідних емпіричних даних

Попередню перевірку ВВ на відповідність нормальному закону розподілу можна виконати за допомогою критеріїв згоди, наприклад, критерію згоди χ^2 Пірсона [43], який дозволяє зіставити теоретичний та емпіричний розподіли ВВ.

Обчислити значення критерію χ^2 Пірсона можна наступним чином [44]:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \tag{2.10}$$

де m – кількість підінтервалів,

n – обсяг вибірки,

n_i – кількість значень ВВ, які потрапили в i-й підінтервал,

p_i – ймовірність потрапляння ВВ в i-й підінтервал.

Кількість підінтервалів m можна розрахувати в залежності від обсягу вибірки n [44].

Для перевірки необхідно порівняти фактично отримане значення величини χ^2 з його критичним значенням, розрахованним в залежності від рівня значимості α та кількості ступенів вільності ν. Якщо χ^2 менше або дорівнює критичному значеню, то з ймовірністю $1–\alpha$ можна зробити висновок, що закон розподілу ВВ є нормальним.

Для знаходження викидів у вихідних ЕД можна використати квадрат відстані Махаланобіса [45, 46], який може бути представлений наступним чином:

$$d_i^2 = (z_i - \bar{z})^T S_n^{-1} (z_i - \bar{z}), \tag{2.11}$$

де $\bar{z}$ – середній вектор вибірки;

S_N – матриця кореляції вибірки, яка визначається наступним чином:

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})(z_i - \bar{z})^T .$$

Далі, використовуючи критерій Пірсона χ^2, треба порівняти отримані значення d_i^2 з квантілем розподілу χ^2 для відповідного рівняь значущості, який у нашому випадку дорівнюватиме 0,005. Точки, для яких значення d_i^2 більше, ніж квантиль розподілу χ^2, є викидами, які потрібно видалити.

Алгоритм ітераційно повторюється доти, доки всі значення d_i^2 не будуть менше або дорівнюватимуть квантилю розподілу χ^2.

2.5 Оцінка якості регресійних моделей

Для перевірки якості рівняння регресії використаємо коефіцієнт детермінації R^2 [47], який може бути представлений наступним чином:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \right), \qquad (2.12)$$

де y_i – емпіричне значення y,

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

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

При значенні $R^2 \geq 0{,}5$ вважається, що така модель є прийнятною. При значенні $R^2 \geq 0{,}8$ вважається, що така модель є достатньо ефективною.

Використаємо величину відносної похибки *MRE* (Magnitude of Relative Errors), яка може бути представлена наступним чином:

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

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

y_i – емпіричне значення *y*,

для подальшого обчислення середньої величини відносної похибки *MMRE* (Mean of Magnitude of Relative Errors). *MMRE* може бути представлена наступним чином:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \, . \qquad (2.13)$$

Рівень прогнозування PRED(0,25) може бути представлений наступним чином:

$$PRED(0{,}25) = \frac{k}{N}, \qquad (2.14)$$

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

N – кількість значень у вибірці.

3 МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

Метою цього розділу є побудова математичної моделі для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів у вигляді нелінійної регресійної моделі залежності кількості дефектів ПЗ розпізнавання номерних знаків в залежності від його розміру. Для побудови нелінійної регресійної моделі були використані дані 31 проекта відповідного ПЗ (включаючи різні версії та релізи). ВВ *X* представляє собою кількість рядків коду ПЗ, KLOC. ВВ *Y* представляє собою загальну кількість дефектів, знайдених у ПЗ. Параметри вибірок ВВ представлені в табл. 3.1. Означені ЕД представлені на рис. 3.1.

Таблиця 3.1 – Параметри вибірок вихідних ЕД

Параметр	ВВ *X*	ВВ *Y*
Кількість значень *n*	31	31
Вибіркове середнє *m*	28,72	87,10
Середньоквадратичне відхилення σ	65,05	101,92
Асиметрія *A*	4,44	2,69
Ексцес ε	24,55	10,72

На етапі попередньої обробки вихідні ЕД були перевірені на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.10). Для довірчої ймовірності 0,95 було обчислене значення χ^2 критичне, яке дорівнює 5,99. χ^2 для ВВ *X* дорівнює 61,21. χ^2 для ВВ *Y* дорівнює 49,08. Звідси можна зробити висновок, що вихідні ЕД не відповідають нормальному закону розподілу.

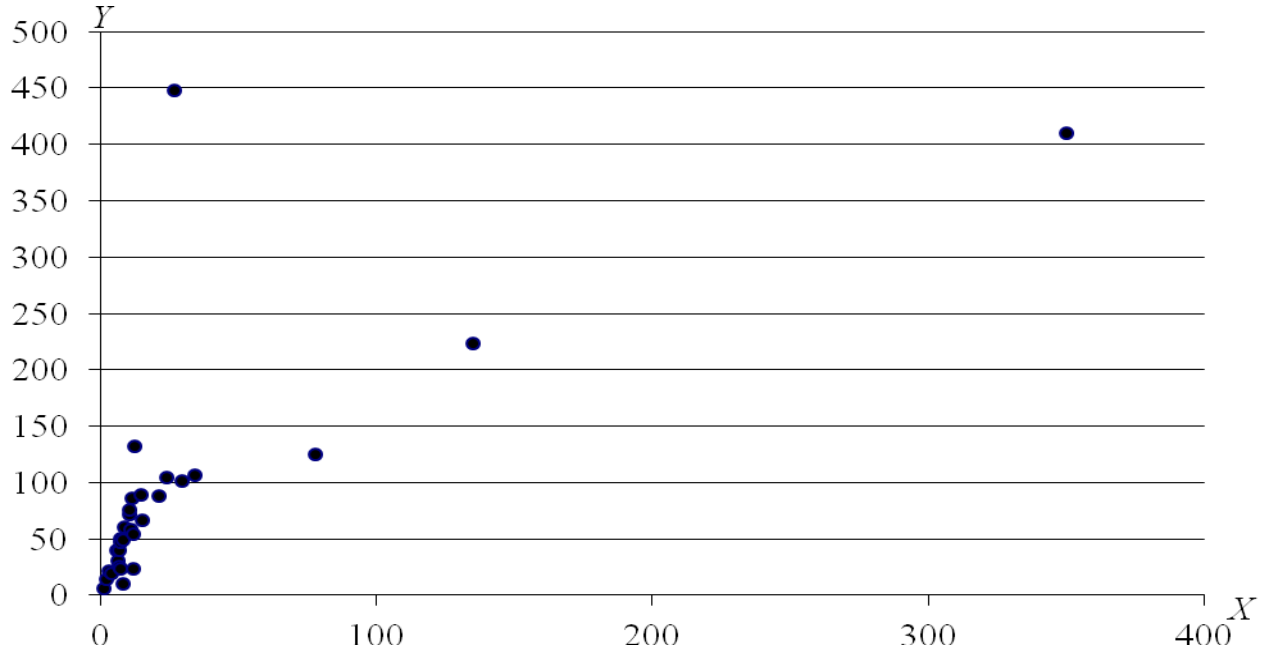

Рисунок 3.1 – Вихідні ЕД для побудови регресійної моделі

Перевіримо вихідні ЕД на наявність викидів за допомогою квадрату відстані Махаланобіса за формулою (2.11) та критерію Пірсона χ^2. Для нормалізації вихідних ЕД використовувалося нормалізуюче перетворення на основі десяткового логарифму за формулою (2.1). Для рівня значущості α=0,005 та m=2 було обчислене значення χ^2 критичне, яке дорівнює 10,60. Жодне значення d_i^2 не перевищує критичного значення. Отже, вихідні ЕД не мають викидів.

Побудуємо лінійну регресійну модель за формулою (2.3). Значення коефіцієнтів лінійного регресійного рівняння b_1 та b_0 знайдемо за допомогою методу найменших квадратів за формулами (2.4) та (2.5) відповідно.

b_1 = 0,7213, b_0 = 0,9634. Лінійна регресійна модель має такий вигляд:

$$Z_y=0,7213Z_x+0,9634+\varepsilon.$$

Далі потрібно перевірити випадкову помилку ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за

формулою (2.10). Для довірчої ймовірності 0,95 критичне значення χ^2 дорівнює 5,99. Розраховане значення χ^2 для ε дорівнює 3,66. Робимо висновок, що ВВ ε відповідає нормальному закону розподілу.

Далі будуємо довірчий інтервал рівняння лінійної регресії за формулою (2.6) та інтервал прогнозування рівняння лінійної регресії за формулою (2.7). Побудоване лінійне рівняння регресії та інтервали разом з нормалізованими ЕД представлені на рис. 3.2.

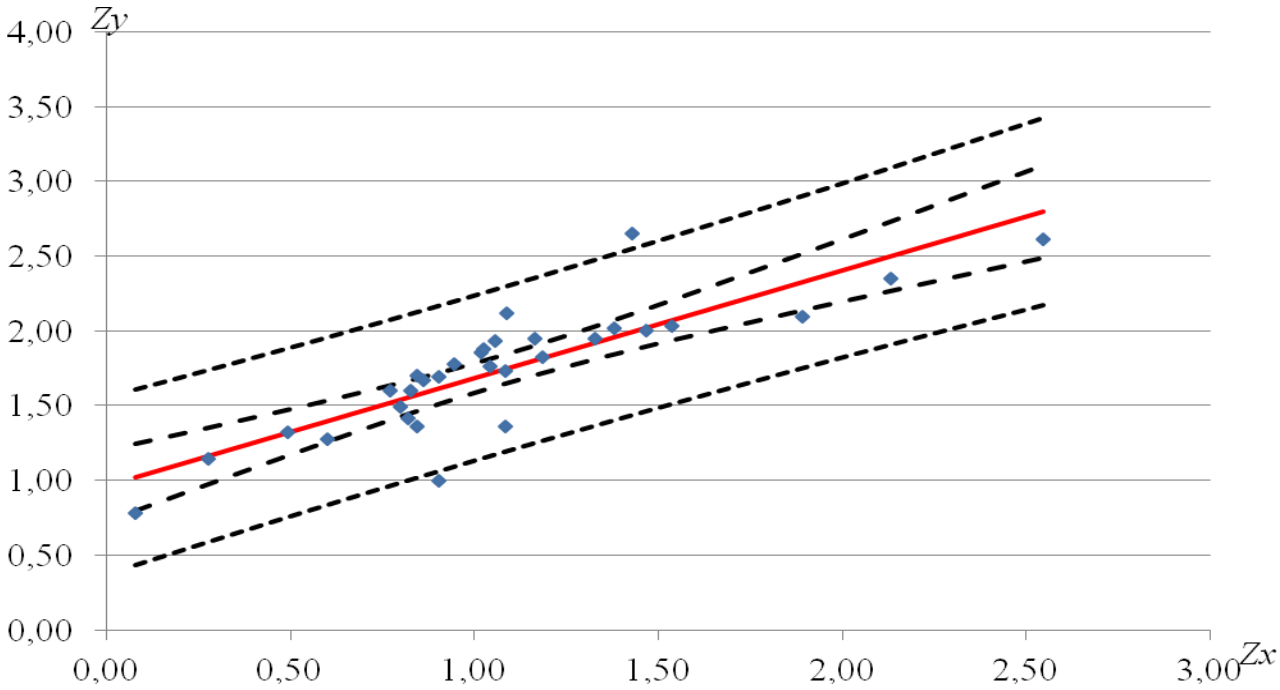

Рисунок 3.2 – Нормалізовані ЕД, лінійне рівняння регресії, довірчий інтервал, інтервал прогнозування

Як можна бачити із рис. 3.2, не всі точки даних знаходяться всередині інтервалу прогнозування. Це може свідчити про те, що не всі викиди були знайдені та видалені на етапі попередньої обробки вихідних ЕД.

Далі на основі вже отриманих даних будуємо нелінійну регресійну модель за формулою (2.9). Нелінійна регресійна модель має такий вигляд:

$$y = x^{0,7213} \cdot 10^{0,9634+\varepsilon}.$$

Далі будуємо довірчий інтервал рівняння нелінійної регресії на основі побудованого довірчого інтервалу лінійної регресій та зворотного перетворення за формулою (2.2). Інтервал прогнозування рівняння нелінійної регресії будуємо аналогічно. Побудоване нелінійне рівняння регресії та інтервали разом з вихідними ЕД представлені на рис. 3.3.

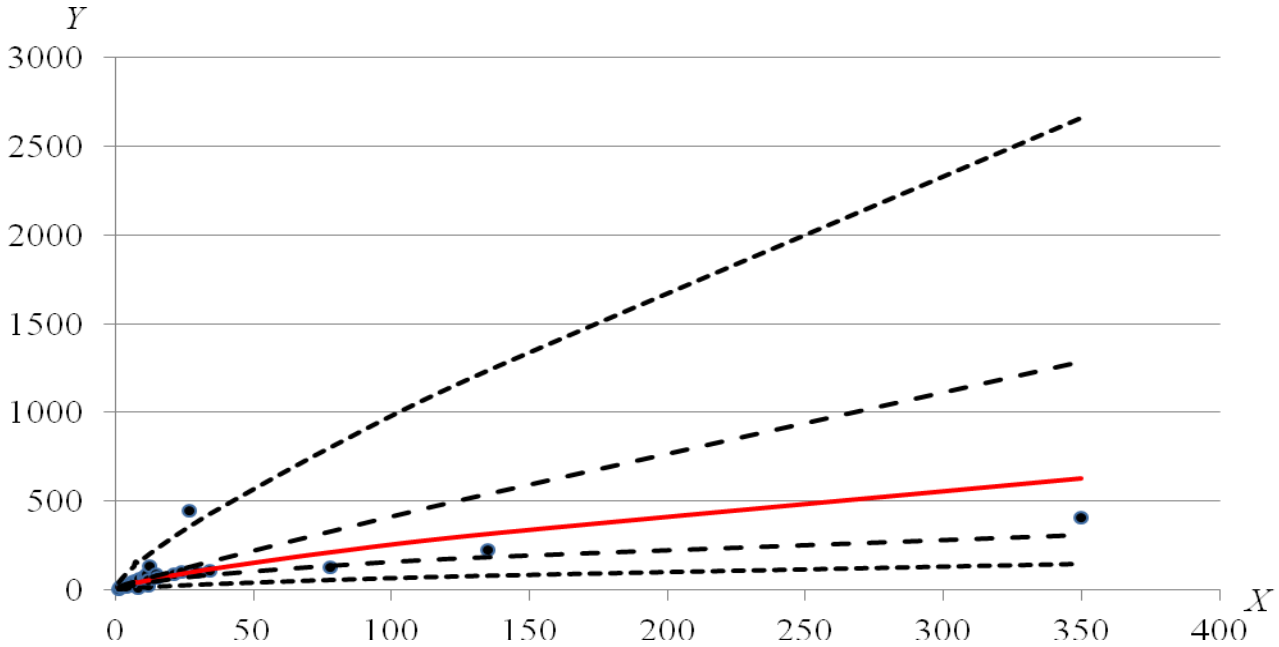

Рисунок 3.3 – Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал, інтервал прогнозування

Як можна бачити із рис. 3.3, не всі точки даних знаходяться всередині інтервалу прогнозування. Це свідчить про те, що не всі викиди були знайдені та видалені на етапі попередньої обробки вихідних ЕД. Видалимо з набору даних 2 точки, які лежать за межами інтервалу прогнозування. Скоригована таким чином вибірка містить 29 наборів даних.

Нормалізуємо ВВ X та ВВ Y за допомогою нормалізуючого перетворення на основі десяткового логарифму за формулою (2.1). Гістограми розподілу нормалізованих ВВ Z_X та Z_Y представлені на рис. 3.4 та 3.5 відповідно.

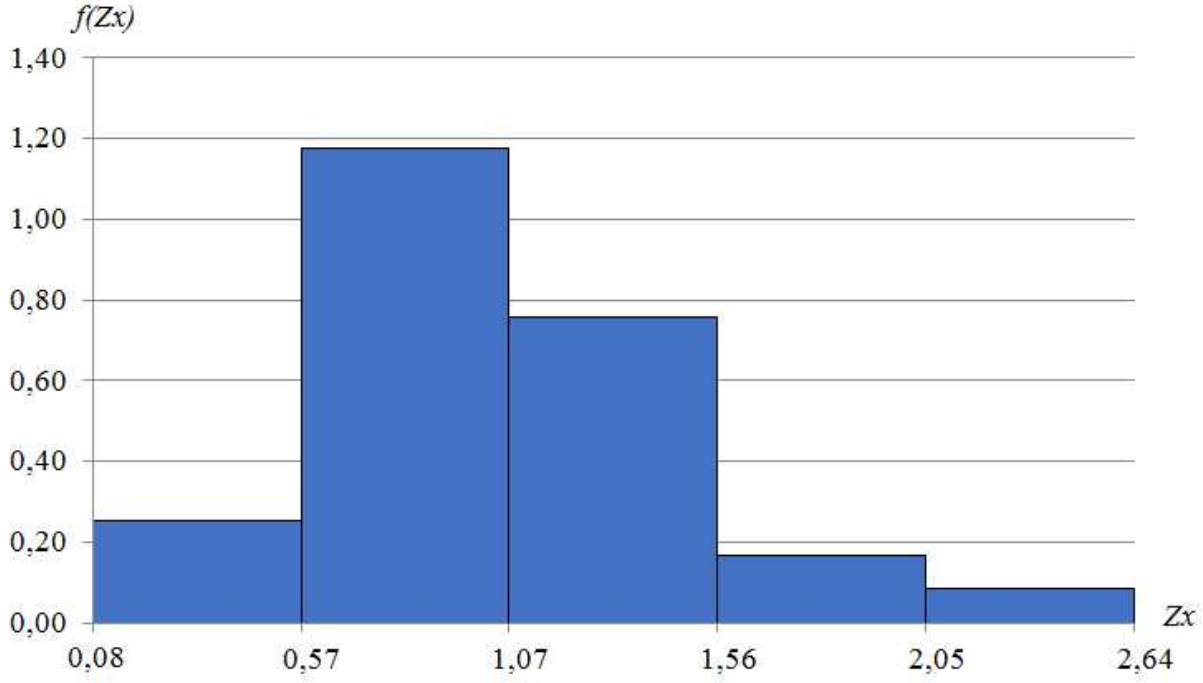

Рисунок 3.4 – Гістограма розподілу нормалізованої ВВ Z_X

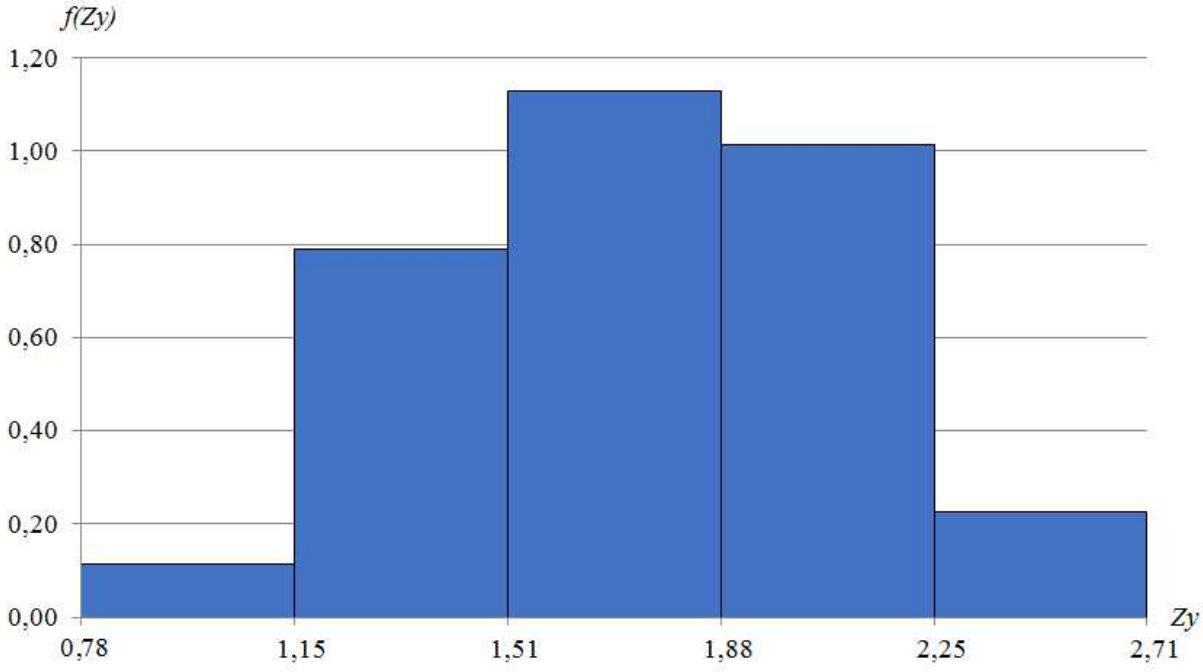

Рисунок 3.5 – Гістограма розподілу нормалізованої ВВ Z_Y

Перевіримо нормалізовані вибірки ВВ Z_X та Z_Y на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона. Для довірчої ймовірності 0,95 χ^2 критичне дорівнює 5,99. χ^2 для ВВ Z_X дорівнює

3,43. χ^2 для ВВ Z_Y дорівнює 0,62. Нормалізовані вибірки ВВ Z_X та Z_Y відповідають нормальному закону розподілу.

Перевіримо скориговану вибірку на наявність викидів, використовуючи квадрат відстані Махаланобіса та критерій Пірсона χ^2. Жодне значення d_i^2 не перевищує значення χ^2 критичне. Отже, скоригована вибірка не має викидів.

Будуємо лінійну регресійну модель, яка має такий вигляд:

$$Z_y=0{,}6741Z_x+1{,}0125+\varepsilon.$$

Перевіряємо випадкову помилку ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона: для довірчої ймовірності 0,95 критичне значення χ^2 дорівнює 5,99, розраховане значення χ^2 дорівнює 1,12. ВВ ε відповідає нормальному закону розподілу. Гістограма розподілу ВВ ε представлена на рис. 3.6.

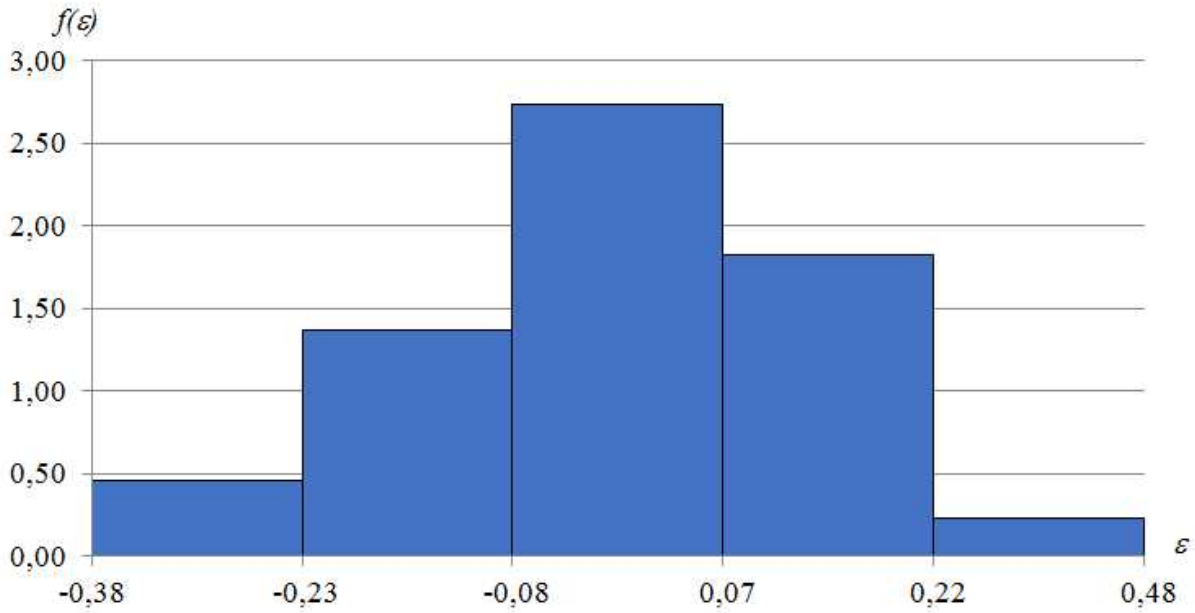

Рисунок 3.6 – Гістограма розподілу ВВ ε

Будуємо довірчий інтервал рівняння лінійної регресії та інтервал прогнозування рівняння лінійної регресії. Побудоване лінійне рівняння регресії та інтервали разом з нормалізованими даними для скоригованої вибірки представлені на рис. 3.7.

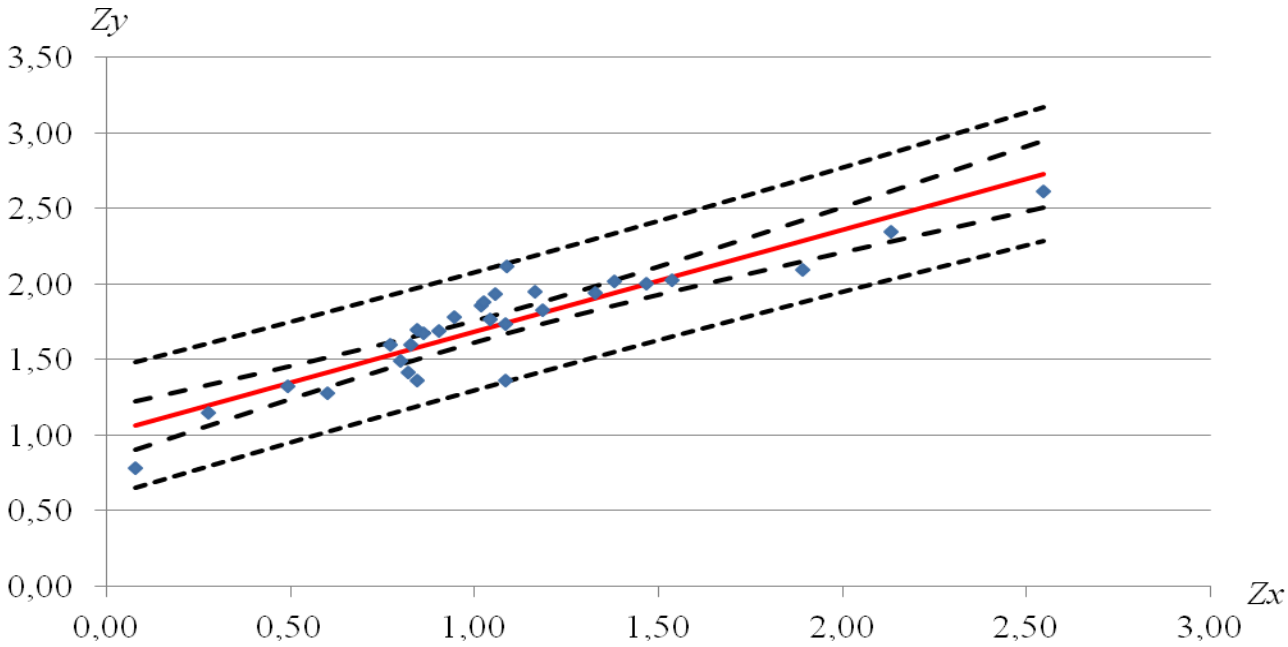

Рисунок 3.7 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал, інтервал прогнозування для скоригованої вибірки

Із рис. 3.7 можна бачити, що всі точки даних знаходяться всередині інтервалу прогнозування. Отже, всі викиди були знайдені та видалені вірно.

Будуємо нелінійну регресійну модель, яка має такий вигляд:

$$y = x^{0,6741} \cdot 10^{1,0125+\varepsilon}.$$

Будуємо довірчий інтервал рівняння нелінійної регресії та інтервал прогнозування рівняння нелінійної регресії. Побудоване нелінійне рівняння регресії та інтервали разом із скоригованими ЕД представлені на рис. 3.8.

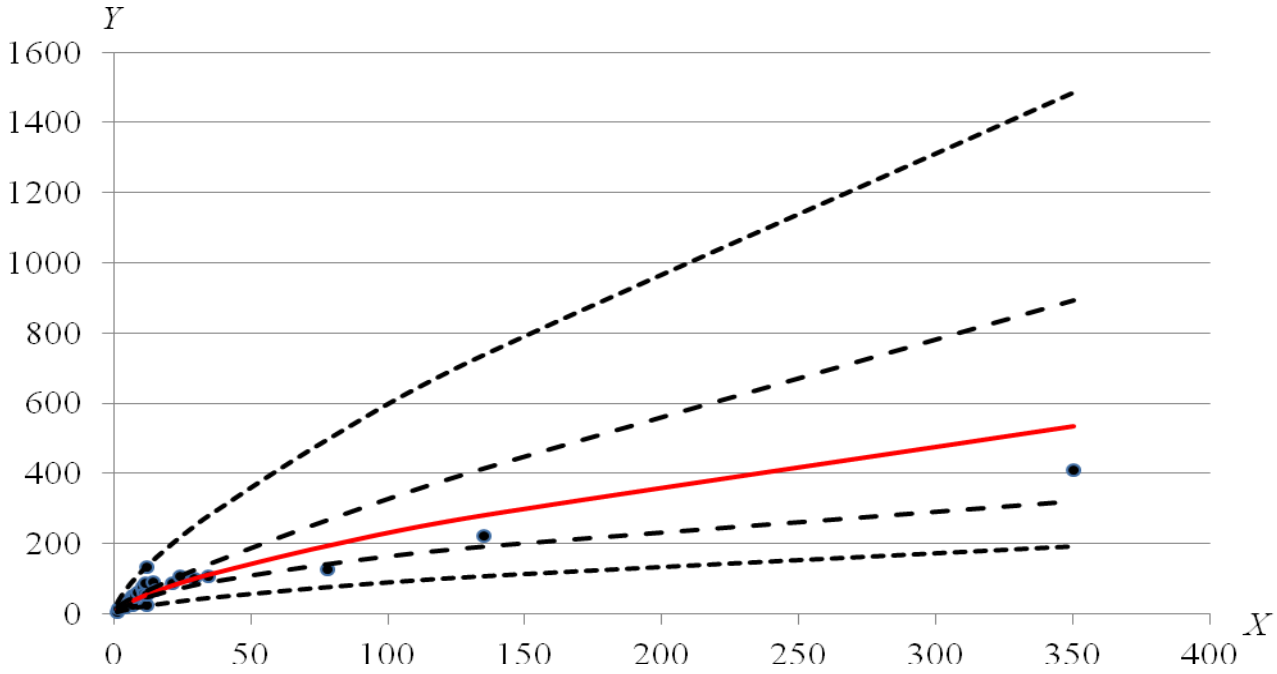

Рисунок 3.8 –ЕД, нелінійне рівняння регресії, довірчий інтервал, інтервал прогнозування для скоригованої вибірки

Із рис. 3.8 можна бачити, що всі точки даних знаходяться всередині інтервалу прогнозування. Отже, скориговані дані не містять викидів.

Побудуємо за скоригованими ЕД лінійну регресійну модель без проведення нормалізації згідно з (2.3) для порівняння з побудованою нелінійною регресійною моделлю. Побудуємо довірчий інтервал рівняння лінійної регресії згідно з (2.6) та інтервал прогнозування рівняння лінійної регресії згідно з (2.7). Лінійна регресійна модель та відповідні інтервали для скоригованих ЕД наведена на рис. 3.9.

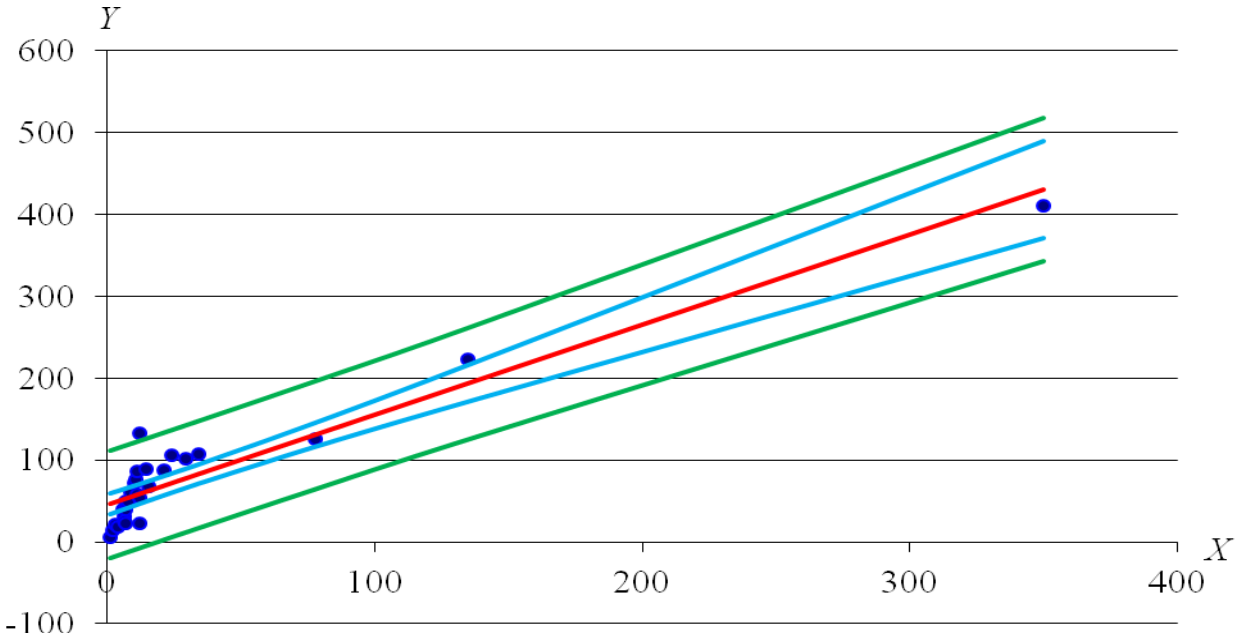

Рисунок 3.9 – ЕД, лінійне рівняння регресії, довірчий інтервал, інтервал прогнозування для скоригованої вибірки

Із рис. 3.9 можна бачити, що нижня границя інтервалу прогнозування частково лежить нижче нуля, що неприпустимо з точки зору предметної галузі, оскільки кількість дефектів ПЗ не можк мати від`ємних значень.

Для більш легкого порівняння зведемо у таблиці розраховані значення границь інтервалів. Границі довірчих інтервалів представлені у табл. 2.2, границі інтервалів прогнозування представлені у табл. 2.3.

Таблиця 2.2 – Розраховані значення границь довірчих інтервалів

Нелінійна регресійна модель			Лінійна регресійна модель		
нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
1	2	3	4	5	6
8,06	16,79	8,73	33,24	59,12	25,88
11,64	21,61	9,97	34,06	59,84	25,78
17,14	28,40	11,26	35,46	61,08	25,61
20,90	32,85	11,95	36,51	62,01	25,50
28,10	41,26	13,16	38,72	63,98	25,26
29,51	42,92	13,41	39,19	64,40	25,21
30,55	44,15	13,61	39,53	64,71	25,17

Продовження табл. 2.2

1	2	3	4	5	6
30,89	44,56	13,67	39,65	64,81	25,16
31,90	45,77	13,87	40,00	65,13	25,13
31,90	45,77	13,87	40,00	65,13	25,13
32,89	46,97	14,08	40,34	65,44	25,09
35,15	49,73	14,58	41,15	66,17	25,01
37,63	52,82	15,19	42,08	67,00	24,93
42,30	58,87	16,57	43,92	68,68	24,76
42,86	59,61	16,76	44,15	68,89	24,74
43,96	61,09	17,14	44,61	69,31	24,70
45,04	62,57	17,53	45,07	69,73	24,66
46,89	65,13	18,24	45,87	70,47	24,60
46,89	65,13	18,24	45,87	70,47	24,60
47,15	65,49	18,35	45,99	70,57	24,59
53,07	74,13	21,06	48,73	73,11	24,38
54,70	76,61	21,91	49,53	73,85	24,33
67,25	97,33	30,07	56,30	80,27	23,97
72,28	106,38	34,10	59,32	83,19	23,87
81,49	124,04	42,55	65,30	89,09	23,78
89,27	139,94	50,67	70,77	94,62	23,85
141,40	266,42	125,03	115,89	145,40	29,51
190,94	413,36	222,43	170,92	215,75	44,82
319,64	892,05	572,41	370,86	488,71	117,85

Із таблиці 2.2 можна бачити, що границі довірчих інтервалів як нелінійної, так і лінійної регресійних моделей, не мають від`ємних значень та ширина довірчого інтервалу для нелінійної регресійної моделі менша, ніж ширина довірчого інтервалу для лінійної регресійної моделі, для 22 точок, або 76% даних.

Таблиця 2.3 – Розраховані значення границь інтервалів прогнозування

Нелінійна регресійна модель			Лінійна регресійна модель		
нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
1	2	3	4	5	6
4,47	30,32	25,86	-19,15	111,52	130,67
6,21	40,49	34,28	-18,37	112,28	130,65
8,79	55,37	46,57	-17,04	113,58	130,62

Продовження табл. 2.3

1	2	3	4	5	6
10,51	65,29	54,78	-16,04	114,56	130,60
13,77	84,19	70,42	-13,92	116,63	130,55
14,41	87,91	73,50	-13,48	117,06	130,54
14,88	90,66	75,78	-13,15	117,39	130,53
15,03	91,56	76,53	-13,03	117,50	130,53
15,49	94,25	78,76	-12,70	117,82	130,53
15,49	94,25	78,76	-12,70	117,82	130,53
15,94	96,91	80,97	-12,37	118,15	130,52
16,97	102,99	86,02	-11,59	118,91	130,50
18,11	109,74	91,63	-10,70	119,78	130,49
20,29	122,73	102,44	-8,93	121,53	130,46
20,55	124,30	103,75	-8,71	121,75	130,45
21,07	127,44	106,37	-8,26	122,18	130,44
21,59	130,54	108,95	-7,82	122,62	130,44
22,47	135,89	113,42	-7,04	123,38	130,42
22,47	135,89	113,42	-7,04	123,38	130,42
22,60	136,65	114,05	-6,93	123,49	130,42
25,49	154,31	128,82	-4,27	126,11	130,38
26,30	159,30	133,00	-3,50	126,88	130,37
32,77	199,74	166,97	3,13	133,44	130,31
35,45	216,86	181,41	6,11	136,40	130,29
40,50	249,58	209,08	12,06	142,33	130,27
44,87	278,44	233,57	17,55	147,84	130,29
75,84	496,69	420,85	64,93	196,37	131,44
107,00	737,60	630,60	125,48	261,18	135,70
192,03	1484,85	1292,82	342,76	516,81	174,05

Із таблиці 2.3 можна бачити, що границі довірчого інтервалу нелінійної регресійної моделі не мають від`ємних значень, тоді, як границі довірчого інтервалу лінійної регресійної моделі мають від`ємні значення. Ширина довірчого інтервалу для нелінійної регресійної моделі менша, ніж ширина довірчого інтервалу для лінійної регресійної моделі, для 21 точки, або 72% даних. Використовувати побудовану лінійну регресійну модель для оцінювання якості ПЗ розпізнавання номерних знаків не бажано, оскільки вона не досить адекватно відображає вихідні ЕД.

4 ПРОЕКТ РОЗРАХУНКОВОЇ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

4.1 Ескізний проект програмного забезпечення

4.1.1 Вибір мови моделювання та побудова діаграми варіантів використання

Універсальна мова моделювання, або UML – це мова позначень, призначена для визначення, візуалізації і документування моделей, зорієнтованих на об’єкти систем ПЗ. UML не є мовою розробки, у конструкціях UML не повідомляється, що робити в яку чергу, і не надається інструкцій щодо побудови системи. UML допомагає наочно представити компонування системи та полегшує співпрацю з її розробниками. UML застосовується для моделювання ПЗ в об`єктно-орієнтованій методології [48]. Графічна нотація UML призначена для візуалізації системи, а специфікації описують її деталі. Оскільки UML є загальноприйнятим стандартом графічного опису ПЗ, використаємо її для моделювання розрахункової програми для оцінювання якості програмного забезпечення для розпізнавання номерних знаків автомобілів.

Для того, щоб зрозуміти як повинна працювати система, використовується опис функціональності системи через її варіанти використання, які також мають назви прецеденти або UseCase. 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 [49].

Діаграма варіантів використання представляє собою граф спеціального вигляду, який є графічною нотацією для представлення конкретних варіантів використання, акторів, або дійових осіб, деяких інтерфейсів і відносин між цими елементами.

Метою розробки діаграми варіантів використання є:

– визначення загальних меж і предметної області;

– формулювання загальних вимог до функціональної поведінки системи;

– розробка початкової концептуальної моделі системи та її подальшої деталізації у формі логічних і фізичних моделей;

– підготовка початкової документації для взаємодії розробників системи з замовниками і користувачами.

В процесі побудови діаграми варіантів використання було виявлено одного актора – Користувача, який виконує всі дії у програмі для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів.

Побудована діаграма варіантів використання розрахункової програми для оцінювання якості програмного забезпечення для розпізнавання номерних знаків автомобілів представлена на рис. 4.1.

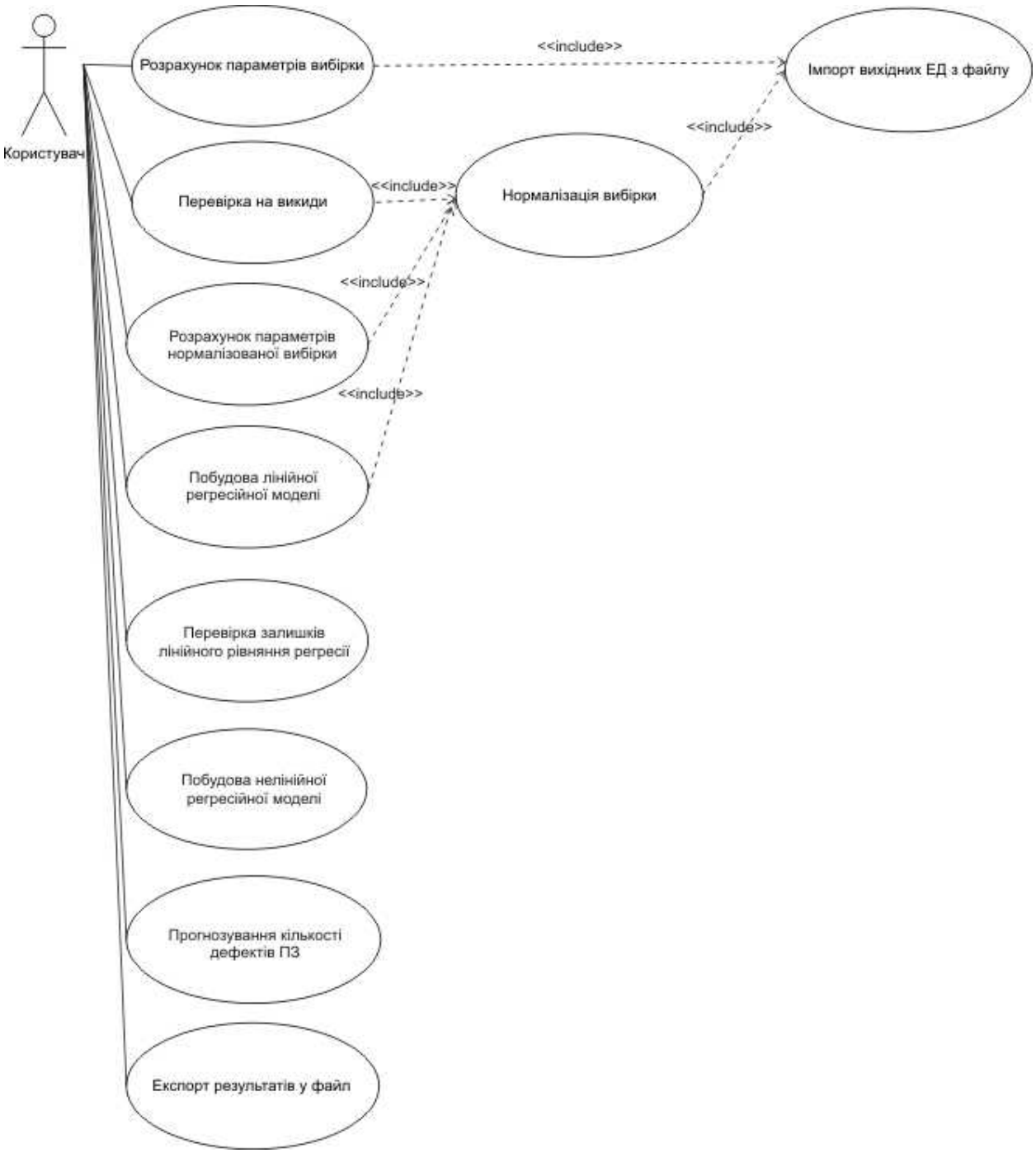

Рисунок 4.1 – Діаграма варіантів використання розрахункової програми

Діаграма варіантів використання є основою для проектування системи. Далі необхідно розробити специфікацію виявлених варіантів використання.

4.1.2 Специфікація варіантів використання

Для кожного варіанту використання, виявленого у попередньому пункті, необхідно розробити специфікацію. Кожна специфікація містить такі складові: короткий опис, передумови, основний потік подій, альтернативний потік подій, постумови. Більш детально варіанти використання представлені в табл. 4.1 – 4.10.

Таблиця 4.1 – Опис варіанту використання "Імпорт вихідних ЕД з файлу"

Складова	Опис
Короткий опис	Імпорт вихідних ЕД з файлу
Передумови	Запуск програми
Основний потік подій	Актор вибирає файл вихідними ЕД і імпортує їх для подальшої обробки
Альтернативний потік подій	Якщо файл має не вірний формат, то програма повідомить про помилку
Постумови	-

Таблиця 4.2 – Опис варіанту використання "Розрахунок параметрів вибірки"

Складова	Опис
Короткий опис	Розрахунок параметрів вибірки
Передумови	Успішний імпорт вихідних ЕД з файлу
Основний потік подій	Програма розраховує основні параметри вибірки
Альтернативний потік подій	-
Постумови	-

Таблиця 4.3 – Опис варіанту використання "Нормалізація вибірки"

Складова	Опис
1	2
Короткий опис	Виконання нормалізації вибірки
Передумови	Успішний імпорт імпорт вихідних ЕД з файлу

Продовження табл. 4.3

1	2
Основний потік подій	Програма розраховує нормалізацію вибірки за допомогою десяткового логарифму
Альтернативний потік подій	-
Постумови	-

Таблиця 4.4 – Опис варіанту використання "Перевірка на викиди"

Складова	Опис
Короткий опис	Перевірка на викиди
Передумови	Успішне виконання нормалізації вибірки
Основний потік подій	Програма розраховує значення квадрату відстані Махаланобіса для всіх даних вибірки
Альтернативний потік подій	-
Постумови	-

Таблиця 4.5 – Опис варіанту використання "Розрахунок параметрів нормалізованої вибірки"

Короткий опис	Розрахунок параметрів нормалізованої вибірки
Передумови	Успішне виконання нормалізації вибірки
Основний потік подій	Програма розраховує основні параметри нормалізованої вибірки
Альтернативний потік подій	-
Постумови	-

Таблиця 4.6 – Опис варіанту використання "Побудова лінійної регресійної моделі"

Складова	Опис
Короткий опис	Побудова лінійної регресійної моделі
Передумови	Успішне виконання нормалізації вибірки
Основний потік подій	Програма будує лінійне рівняння регресії, довірчий інтервал лінійного рівняння регресії, інтервал прогнозування лінійного рівняння регресії
Альтернативний потік подій	-
Постумови	-

Таблиця 4.7 – Опис варіанту використання "Перевірка залишків лінійного рівняння регресії"

Складова	Опис
Короткий опис	Перевірка залишків лінійного рівняння регресії
Передумови	Успішна побудова лінійної регресійної моделі
Основний потік подій	Програма перевіряє залишки лінійного рівняння регресії на відповідність нормальному закону розподілу
Альтернативний потік подій	-
Постумови	-

Таблиця 4.8 – Опис варіанту використання "Побудова нелінійної регресійної моделі"

Складова	Опис
Короткий опис	Побудова нелінійної регресійної моделі
Передумови	Успішна побудова лінійної регресійної моделі
Основний потік подій	Програма будує нелінійне рівняння регресії, довірчий інтервал нелінійного рівняння регресії, інтервал прогнозування нелінійного рівняння регресії
Альтернативний потік подій	-
Постумови	-

Таблиця 4.9 – Опис варіанту використання "Прогнозування кількості дефектів ПЗ"

Складова	Опис
Короткий опис	Прогнозування кількості дефектів ПЗ
Передумови	Успішна побудова нелінійної регресійної моделі
Основний потік подій	Програма розраховує прогнозне значення загальної кількості дефектів ПЗ в залежності від введеного значення кількості рядків коду ПЗ у KLOC
Альтернативний потік подій	-
Постумови	-

Таблиця 4.10 – Опис варіанту використання "Експорт результатів у файл"

Складова	Опис
Короткий опис	Збереження результатів розрахунків у файл
Передумови	Успішна побудова нелінійної регресійної моделі
Основний потік подій	Програма запропонує користувачу зберегти файл з результатами розрахунків
Альтернативний потік подій	-
Постумови	-

Застосування варіантів використання на всіх рівнях діаграми дозволяє не тільки досягти необхідного рівня уніфікації позначень для подання функціональності підсистем і системи в цілому, але і є потужним засобом послідовного уточнення вимог до проектованої системи на основі спуску від пакетів системи до операцій класів [49].

4.1.3 Побудова діаграм діяльності

Діаграми діяльності в UML застосовуються для моделювання процесу виконання операцій. Діаграми діяльності та діаграми станів багато в чому схожі, але є і відмінність: семантика станів, яка використовується для уявлення діяльностей, а не дій [48].

Для описаних у попередньому пункті варіантів використання були побудовані діаграми діяльності, деякі з яких приведені на рис. 4.2 – 4.3.

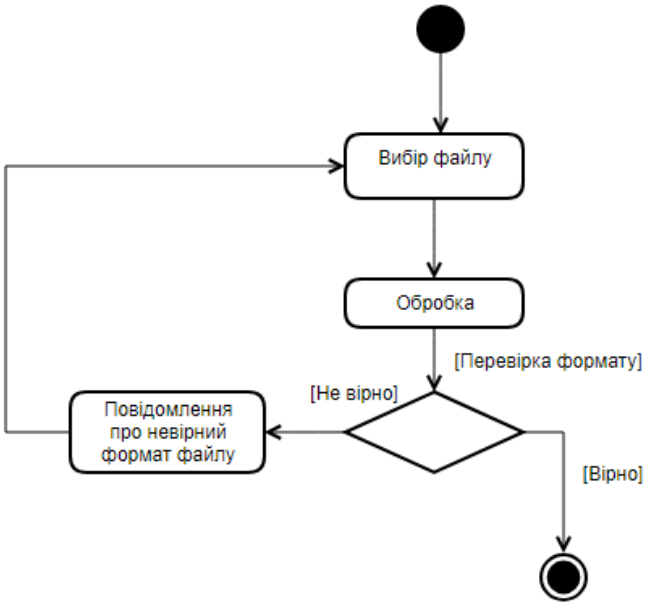

Рисунок 4.2 – Діаграма діяльності варіанту використання «Імпорт вихідних ЕД з файлу»

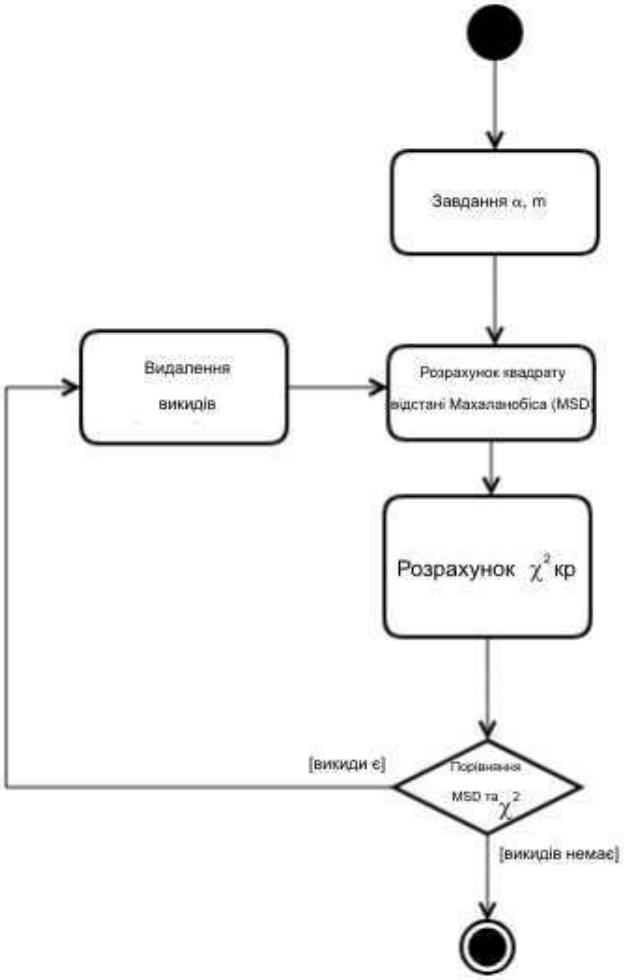

Рисунок 4.3 – Діаграма діяльності варіанту використання «Перевірка на викиди»

Кожна діаграма діяльності має єдиний початковий стан, в якому вона починається, та єдиний кінцевий стан, в якому вона закінчується. Усі дії на діаграмі діяльності розташовуються таким чином, щоб дії слідували зверху вниз.

4.1.4 Спосіб зберігання даних

У випадку даного ПЗ відсутня БД, через те що відсутня необхідність у довгостроковому зберіганні результатів розрахунків. Початкові дані вибірки та результати розрахунків зберігаються у файлу формату CSV(значення, розділені комами).

CSV-файли зазвичай використовуються для обміну табличними даними між системами в звичайному тексті. В основному вони містять рядок заголовка, який надає імена стовпців для даних, але в іншому вони вважаються частково структурованими. Це пов'язано з тим, що CSV-файли з самого початку не можуть представляти ієрархічні або реляційні дані. Зв'язки між даними зазвичай обробляються з використанням декількох CSV-файлів. Зовнішні ключі зберігаються в одному або декількох файлах, однак зв'язки між цими файлами не обробляються самим форматом [50].

Файли в форматі CSV можуть використовувати інші роздільники, крім ком, наприклад символи табуляції або пробілу.

Незважаючи на обмеження, CSV-файли є популярним вибором для обміну даними, так як вони підтримуються широкою низкою бізнес-додатків, споживчих та наукових програм.

Враховуючи переваги і недоліки формату CSV можна сказати що він відмінно підходить для розроблюваного ПЗ адже він має просту структуру і значно спростить процес інтеграції у разі включення ПЗ в великий програмний комплекс.

4.1.5 Розробка прототипу інтерфейсу користувача

Взаємодія між користувачем і комп’ютером (HCI – Human-Computer Interaction) відбувається в інтерфейсі користувача. Основна мета інтерфейсу користувача – це покращення взаємодії між користувачем ПЗ та комп’ютером. ПЗ, з яким взаємодіє користувач, робить комп’ютер більш дружелюбним та сприйнятливими до потреб користувачів.

Саме ефективність використання функцій ПЗ та комфортність роботи користувача визначається у більшості випадків тим, як побудований інтерфейс користувача ПЗ.

З урахуванням всіх вимог до роботи майбутнього ПЗ, було розроблено ескізи форм майбутнього ПЗ, які представлені зображені на рисунках 4.4 – 4.5.

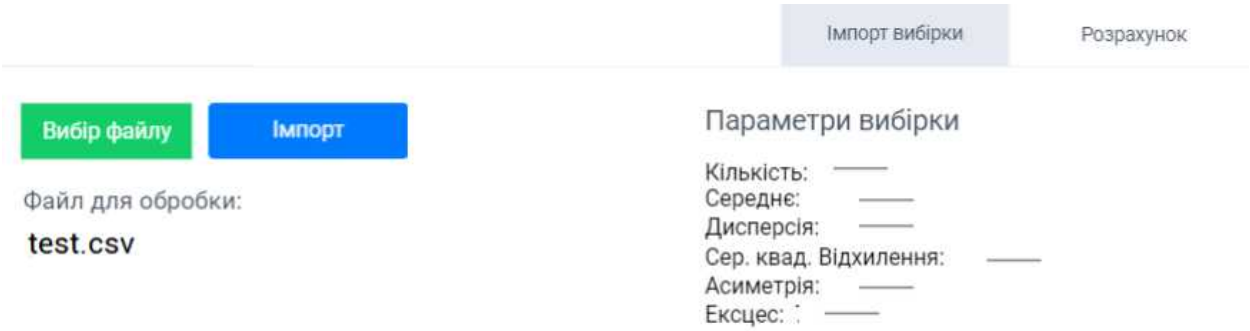

Рисунок 4.4 – Ескіз форми імпорту вихідних ЕД з файлу

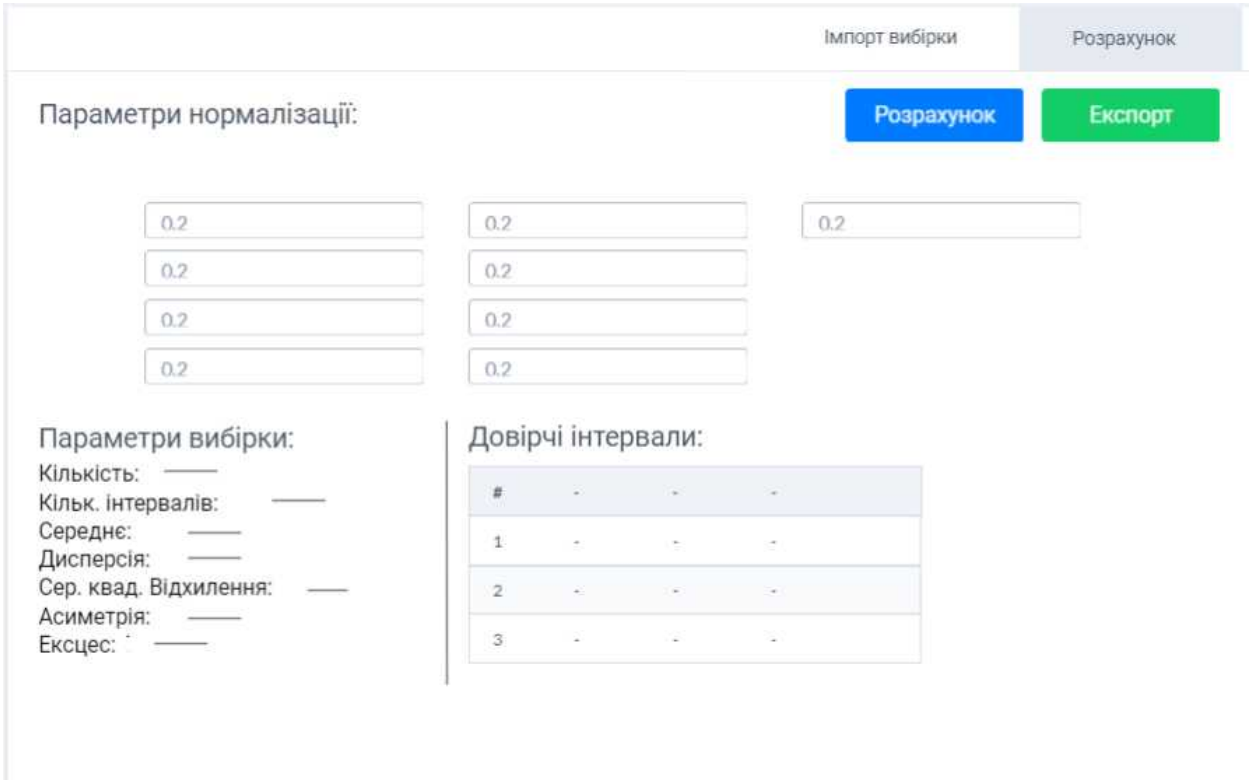

Рисунок 4.5 – Ескіз форми розрахунків

Розроблений прототип інтерфейсу користувача задовольняє поставленим вимогам та забезпечить користувачам програми зручну та комфортну роботу.

4.2 Технічний проект програмного забезпечення

4.2.1 Побудова діаграми класів

Діаграма класів – діаграма на якій представлена сукупність декларативних або статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення, що їх з’єднують [51]. Діаграми класі реалізації деяких варіантів використання наведено на рисунках 4.6 – 4.7.

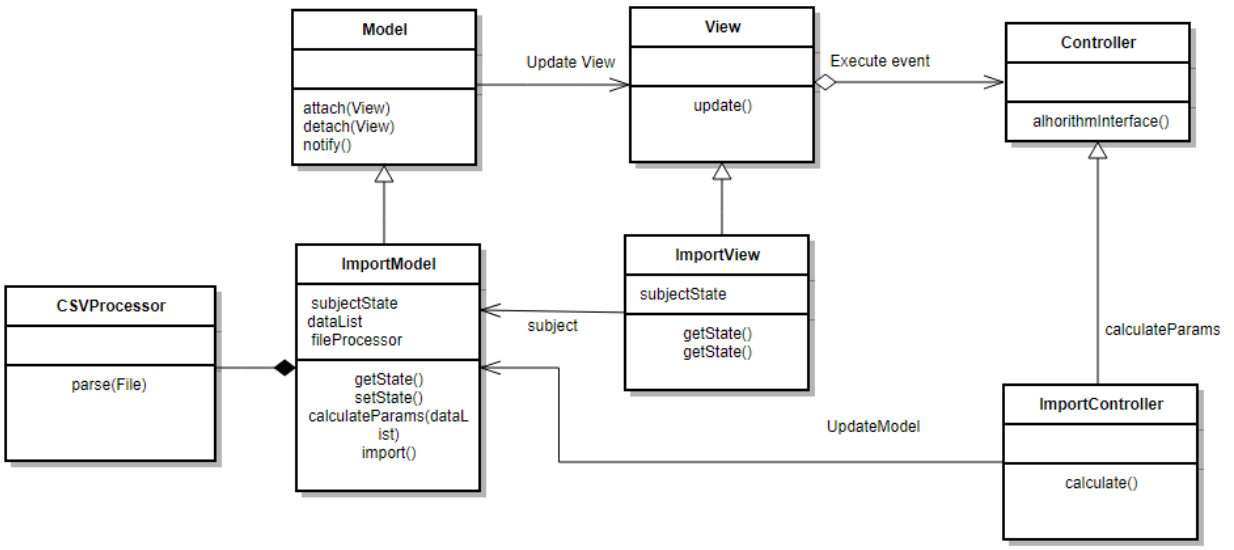

Рисунок 4.6 – Діаграма класів реалізації варіанту використання «Імпорт вихідних ЕД з файлу»

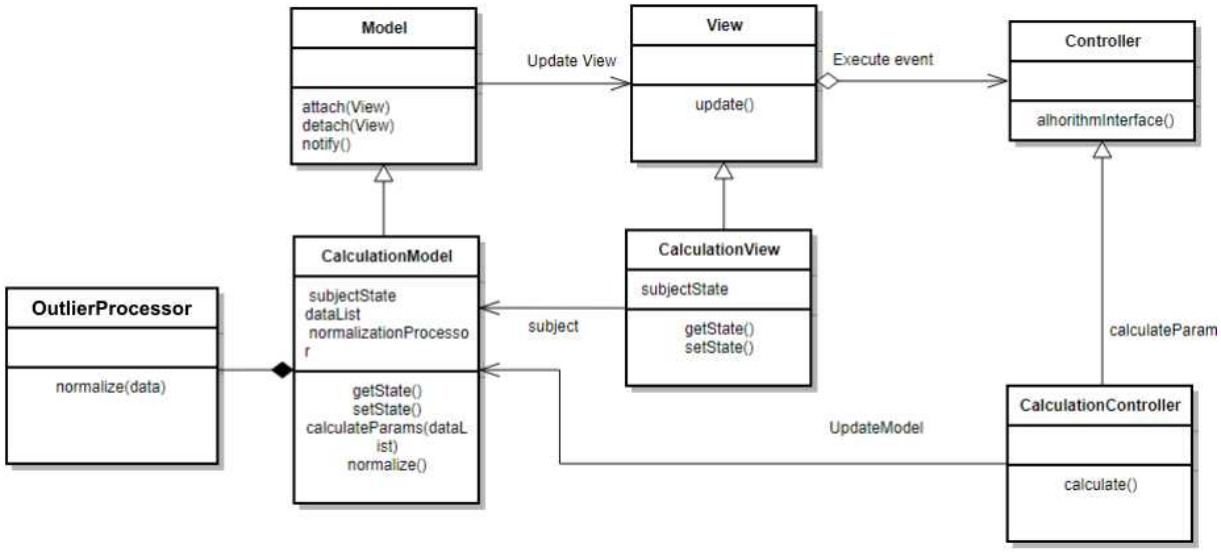

Рисунок 4.7 – Діаграма класів реалізації варіанту використання «Перевірка на викиди»

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.

4.2.2 Побудова діаграми послідовності

Діаграма послідовності – одна з моделей опису поведінки взаємодіючих груп об'єктів в UML. 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 [52]. Діаграми послідовності реалізації деяких варіантів використання наведено на рисунках 4.8 – 4.9.

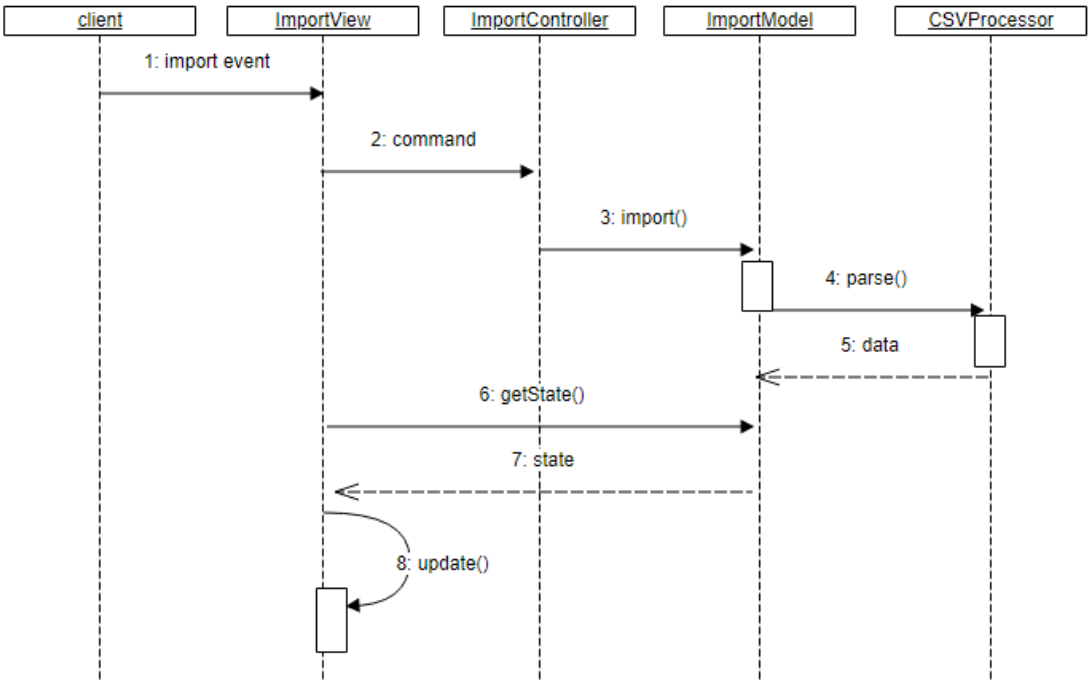

Рисунок 4.8 – Діаграма послідовності реалізації варіанту використання «Імпорт вихвдних ЕД з файлу»

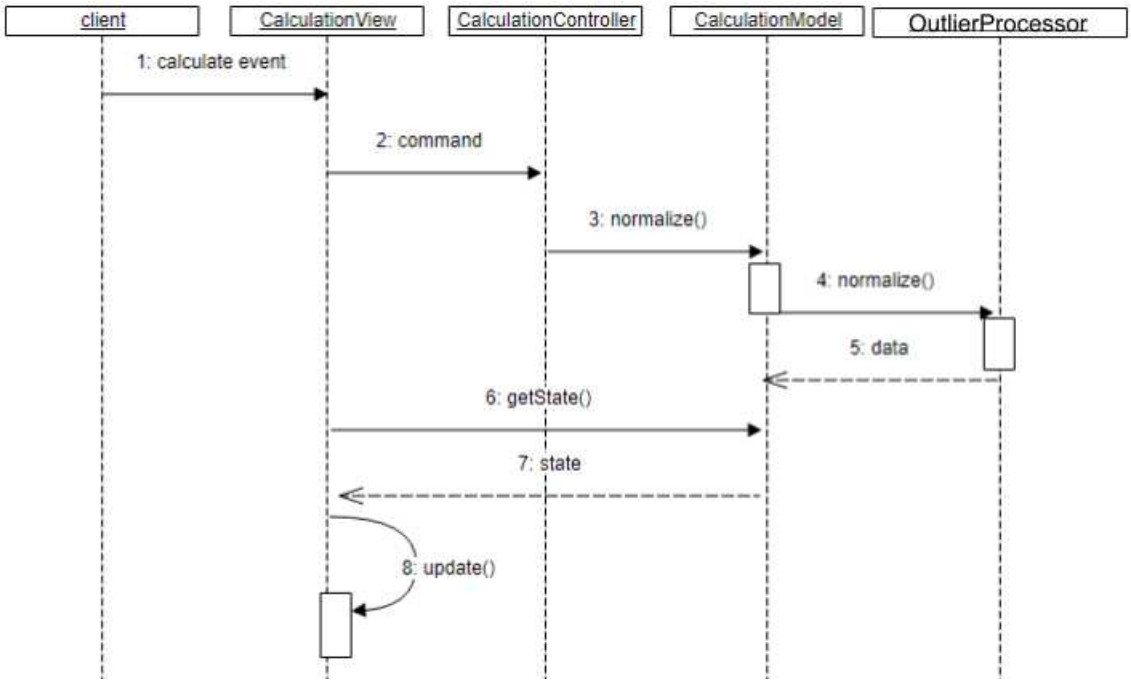

Рисунок 4.9 – Діаграма послідовності реалізації варіанту використання «Перевірка на викиди»

Таким чином, розробивши діаграми послідовності для кожного варіанта використання, маємо опис поведінки взаємодіючих груп об'єктів.

4.3 Робочий проект програмного забезпечення

4.3.1 Обґрунтування вибору мови програмування та засобів розробки

Для розробки ПЗ було обрано об`єктно-орієнтовану мову програмування Java, якою зараз займається компанія «Oracle» [53]. Написані Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Java – лідируюча у світі мова програмування і платформа розробки. Java знижує витрати, скорочує терміни розробки, стимулює інновації та покращує роботу додатків. Завдяки мільйонам розробників, що використовують понад 51 мільярд віртуальних машин Java по всьому світу, ця

програма продовжує залишатися кращою платформою розробки для підприємств та розробників.

До того ж, мова Java призначена для створення програм, які працюють в розподіленому середовищі Internet на базі протоколів TCP/IP. Насправді доступ до ресурсів за допомогою URL відрізняється від доступу к файлу. Крім того в Java наявний засіб передачі повідомлень в межах внутрішнього адресного простору. Це дозволяє забеспечити віддалене виконання процедур. Ці інтерфкйси включені у пакет RMI (remote metod invocation). Цей засіб привносить високий рівень абстракції в програмування для середовища клієнт/сервер.

Також при створенні ПЗ було використано інтегроване середовище розробки (IDE) IntelliJ IDEA [54] від компанії JetBrains [55]. Кожен компонент IntelliJ IDEA створений для того, щоб максимально підвищити продуктивність розробки на Java. Розумний редактор коду у поєднанні з ергономічним дизайном роблять розробку не лише ефективною, а й приємною.

Після індексування вихідного коду IntelliJ IDEA надає безліч можливостей для швидкої та ефективної розробки: розумне автодоповнення, аналіз коду в реальному часі та надійні рефакторинги. Також всі необхідні інструменти вже під рукою, а значить не потрібно шукати та встановлювати плагіни для інтеграції із системами контролю версій та підтримки популярних мов та фреймворків.

4.3.2 Випробування програмного забезпечення

Під час випробування ПЗ було проведено його повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу у конфігурації, яка зазначена у Додатку А – Технічне завдання.

Всі недоліки ПЗ, виявлені під час випробування, були зафіксовані в протоколі та усунуті до моменту впровадження ПЗ у дію.

Програму і методику випробувань наведено у Додатку Д.

Методи випробування

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

Протокол тестування.

1. Проведено перевірку ПЗ на відповідність технічному завданню:

- перевірка роботи функції «Імпорт вихідних ЕД з файлу»

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

- перевірка роботи функції «Розрахунок параметрів вибірки»

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

- перевірка роботи функції «Нормалізація вибірки»

Після імпорту даних, була виконана нормалізацію вибірки.

- перевірка роботи функції «Перевірка на викиди»

Після нормалізації вибірки програма розрахувала квадрат відстані Махаланобіса для всіх точок вибірки, критичне значення χ^2, виконала порівняння цих значень та виділила ті точки, для яких квадрат відстані Махаланобіса більше критичне значення χ^2.

- перевірка роботи функції «Побудова лінійної регресійної моделі»

Після нормалізації вибірки та перевірки на викиди програма виконала побудову лінійної регресійної моделі, а саме: виконала розрахунок коефіцієнтів лінійного рівняння регресії b_1 та b_0, побудувала лінійне рівняння регресії, довірчий інтервал лінійного рівняння регресії та інтервал прогнозування лінійного рівняння регресії.

- перевірка роботи функції «Експорт результатів у файл»

Після проведення розрахунків їх результати було успішно експортовано у файл.

2. Проведено перевірку програми на її програмно-апаратну сумісність:

– виконано тестування на апаратно-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, представлені у технічному завданні, яке наведено у Додатку А;

– 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.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Опис програми

Процес розробки програмних продуктів є досить складним та багатостороннім і недоліки процесу розробки можуть сказатися на якості програмних продуктів, і як наслідок, на їх надійності, не тільки під час розробки та впровадження програмних продуктів, а й під час експлуатації.

Найважливішими метриками для оцінювання якості ПЗ будуть такі метрики, які стосуються різних областей: вимоги до ПЗ, якість ПЗ, ефективність команди тестування, якість роботи QA, зворотний зв'язок.

Найбільш цікавою є друга група, а саме якість ПЗ, яка містить п`ять метрик. Всі вони спираються на один показник – кількість дефектів.

Розроблювана програма виконує оцінювання якості ПЗ розпізнавання номерних знаків автомобілів на основі побудованої удосконаленої математичної моделі. Завдяки цьому удосконаленю можна підвищити достовірність оцінювання якості ПЗ розпізнавання номерних знаків в порівнянні з існуючою моделлю.

У зв’язку з вузькою спеціалізацією питання,майже не існує готового повнофункціонального ПЗ, яке б в повністю вирішувало всі необхідні задачі. Впровадження програми для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів перш за все націлене на зменшення витрат в процесі розробки ПЗ та підвищення ефективності роботи персоналу фірми.

Створення ПЗ вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування. Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення ПЗ

характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

5.2 Розрахунок витрат на створення та експлуатацію програми

Витрати на розробку системи складаються з витрат на заробітну плату 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 15 000,00 грн. Додаткова заробітна плата складає 20% від основної – 3 000,00 грн. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи складає 18 000,00 грн/міс, вартість сучасного комп`ютера складає 19 500,00 грн. (Компьютер Everest Business 3010: Intel Core i3-10100 (3.6 – 4.3 ГГц) / RAM 8 ГБ / SSD 250 ГБ / Intel UHD Graphics 630 / без ОД / LAN / без ОС, монітор 27" HP P27q G4 (8MB11AA) 8-Bit – sRGB 99%). При вартості кіловат-години електроенергії рівної 1,68 грн., розраховується вартість розробки програми. 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	130,00
Флеш пам`ять Transcend JetFlash 700 32GB	169,00
Література	300,00
Непередбачені витрати	50,00
Разом матеріали і комплектуючі вироби	719,00

Вартість програми розрахуємо по формулі:

$$Спр = (Взп + Всф + Взг + Ве) * Т + Вм \quad (5.1)$$

де: Взп – основна і додаткова заробітна плата обслуговуючого персоналу, грн.;

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

Взг – загальногосподарські витрати (10% від основної заробітної плати), грн.;

Ве – витрати на електроенергію;

Т – тривалість розробки, міс.

Вм – витрати на основні і допоміжні матеріали.

Ве при споживанні потужності 500 Вт, тривалості роботи на місяць, рівної 21*8 = 168 годин, вартості кіловат-години електроенергії 1,68 грн.

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

Загальні витрати на розробку системи наведені 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.	18 000,00
Відрахування в соціальні фонди	Грн.	6 840,00
Загальногосподарські витрати	Грн.	1 800,00
Витрати на допоміжні матеріали	Грн.	719,00
Витрати на електроенергію	Грн.	141,12

Відповідно до формули 5.1 вартість програми:

Спр=(18 000,00 + 6 840,00+ 1 800,00+ 141,12)*1 + 719,00=27500,12 грн.

Експлуатаційні витрати для комп`ютера розрахуємо по формулі:

$$Взр = Аоб + Вм + Ве \quad (5.2)$$

де: Аоб – амортизаційні відрахування, грн.

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

Ве – витрати на електроенергію, грн.

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

Аоб = 19 500,00 * 0,6 = 11 700,00 грн.

У масштабах закладу річні витрати на основні і допоміжні матеріали (носії, папір, …) визначаються в розмірі 5% вартості основного устаткування:

Вм = 19 500,00 * 0,05 = 975,00 грн.

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

$$Фм = 253,3 * Тз \quad (5.3)$$

де Тз – це середнє денне завантаження устаткування (близько 7 годин)

253,3 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе:

Фм = 253,3 * 7 = 1773,1 годин

Витрати на електроенергію Ве при 1773,1 годинах роботи устаткування в рік складуть

Ве = 1773,1* 1,68 * 0,5 = 1489,41 грн.

Експлуатаційні витрати для комп`ютера за рік складуть:

Взр = 11 700,00 + 975,00 + 1489,41= 14 164,41 грн.

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

Все = 27 500,12 + 14 164,41 = 41 664,53 грн.

5.3 Економічна ефективність розробки та впровадження програми

Основним показником економічної ефективності функціонування ПЗ є зменшення витрат, підвищення ефективності роботи та удосконалена ступень захисту.

До числа найголовніших факторів, що визначають ефективність роботи в зв'язку з впровадженням ПЗ, відносяться:

– підвищення продуктивності праці;

– підвищення швидкості обробки інформації;

– вивільнення робочого часу;

– умовно-річна економія від підвищення якості захисту інформації.

Розрахуємо пряму економічну ефективність від використання ПЗ, ґрунтуючись на тому, що впровадження програмного продукту зменшує кількість працівників (за експертною оцінкою фахівців установи близько двох осіб), яких потрібно було б задіяти для обробки інформації на старому ПЗ.

Зарплата 2 працівників в рік складає:

$$8000 * 12 * 2 = 192\,000{,}00 \text{ грн.}$$

Економічний ефект першого року розраховується за формулою (5.3):

$$\mathrm{A}_{экон.эф.} = \Delta C_n - E_n * k, \qquad (5.3)$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту (192 000,00 грн.) мінус експлуатаційні витрати (14 164,41 грн.) ;

$\Delta C_n = 192\,000{,}00 - 14\,164{,}41 = 177\,835{,}59$ грн.

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

k – одноразові витрати на впровадження продукту (27 500,12 грн).

$$\mathrm{A}_{экон.эф.} = 177\,835{,}59 - 0{,}6 * 27\,500{,}12 = 161\,335{,}52 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{П} = \frac{27\,500{,}12}{177\,835{,}59} \approx 0{,}15 \text{ (року)}$$

де П – вивільнені кошти після впровадження програмного продукту (177 835,59 грн.);

k – одноразові витрати на впровадження системи (27 500,12 грн.)

Отже строк окупності системи складає приблизно 2 місяці.

6 ОХОРОНА ПРАЦІ

6.1 Основні законодавчі і нормативні документи, які регламентують охорону праці в Україні

До основних законодавчих актів з охорони праці відносяться: Конституція України; Кодекс законів про працю (КЗпП); закони "Про охорону праці" [56], "Про охорону здоров’я", "Про пожежну безпеку", "Про використання ядерної енергії та радіоактивний захист", "Про забезпечення санітарного та епідемічного благополуччя населення". Коло питань щодо охорони праці розглядається в підзаконних актах, указах Президента, постановах Верховної Ради та Кабміну, Цивільному, Кримінальному та Адміністративному кодексах України.

Кожній людині Конституція гарантує право на належні, безпечні і здорові умови праці. Конституцією гарантується захист від незаконного звільнення. Кожна людина має право на достатній життєвий рівень, на охорону здоров’я, медичну допомогу та медичне страхування (ст. 49).

В КЗпП охорона праці відображена в окремому розділі. В окремих статтях цього розділу розглянуто створення здорових і безпечних умов праці;додержання вимог охорони праці при будівництві і експлуатації будівель, споруд та обладнання; заборона введення в експлуатацію підприємств, які не відповідають вимогам охорони праці; заборона передачі у серійне виробництво зразків нових машин (та іншого обладнання), які не відповідають вимогам охорони праці; правила охорони праці обов’язкові для адміністрації, обов’язки адміністрації щодо поліпшення й оздоровлення умов праці.

До найважливіших актів з охорони праці належать: Закон України "Про загальнообов’язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату

працездатності" (23.09.1999 р. № 1105); Порядок розслідування та обліку нещасних випадків, професійних захворювань і аварій на виробництві (25.08.2004 р. №1112); Порядок видачі дозволів Державним комітетом з нагляду за охороною праці та його територіальними органами (15.10.2003 р. № 1631); Порядок проведення атестації робочих місць за умовами праці (01.08.1992 р. №442) і ін.

Спеціальними законодавчими актами є міжгалузеві та галузеві акти про охорону праці: Державні стандарти, Системи стандартів безпеки праці, будівельні норми і правила. Санітарні норми, правила будови електроустановок, норми радіаційної безпеки, правила будови та безпечної експлуатації вантажопідйомних кранів та ін.

6.2 Структура управління питаннями охорони праці на підприємстві

Організація роботи в царині охорони праці полягає у виборі та формуванні такої структури управління охороною праці на підприємстві, яка найповніше відповідала б меті створення безпечних і сприятливих умов праці.

Організація й координація робіт у галузі охорони праці повинні передбачати формування органів управління охороною праці, встановлення обов’язків і порядку взаємодії осіб, які беруть участь в управлінні, а також у прийнятті та реалізації управлінських рішень.

Управління охороною праці на підприємстві здійснює власник підприємства (роботодавець), а у структурних підрозділах – відповідні керівники підрозділів.

Зобов’язання, права та відповідальність посадових осіб за виконання покладених на них функцій щодо питань охорони праці розглядаються в посадових інструкціях, форма яких розроблена Держнаглядохоронпраці. Згідно зі ст. 13 Закону України «Про охорону праці» роботодавець

зобов’язаний створити в кожному структурному підрозділі та на робочому місці умови праці відповідно до вимог нормативних актів, а також забезпечити додержання прав працівників, гарантованих законодавством про охорону праці.

6.3 Перелік шкідливих та небезпечних факторів, які супроводжують роботу програміста

Широке промислове та побутове використання ПК актуалізувало питання охорони праці їхніх користувачів. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98.

Дотримання вимог цих правил може значно знизити наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відеодисплейними матеріалами, зокрема можливість зорових, нервово-емоційних переживань, серцево-судинних захворювань.

Відповідно до встановлених гігієнічно-санітарних вимог (ГОСТ 12.1.005-88, СН 4088-86) роботодавець зобов'язаний забезпечити в приміщеннях з ВДТ оптимальні параметри виробничого середовища, які наведені в табл. 6.1 – 6.4.

Таблиця 6.1 – Норми мікроклімату для приміщень з ВТД

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1 а	22-24	4-6	0,1
	Легка - 1 б	21-23	4-6	0,1
Тепла	Легка - 1 а	23-25	4-6	0,1
	Легка - 1 б	22-24	4-6	0,2

Таблиця 6.2 – Рівні іонізації повітря приміщень при роботі на ВДТ

Рівні	Число іонів в 1 см3 повітря	
	п+	п-
Мінімально необхідні	400	600
Оптимальні	1500-30000	3000-5000
Максимально допустимі	50000	50000

Таблиця 6.3 – Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот

Вид трудової діяльності	Рівні звукового тиску в дБ в октавних смугах із середньогеометричними частотами, Гц								
	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні дБА/дБАекв.
Програмісти ЕОМ	7	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ЕОМ та оператори комп'ютерного набору	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів ЕОМ	91	83	77	73	70	68	66	64	75

Таблиця 6.4 – Допустимі параметри електромагнітних випромінювань і електричного поля

Види поля	Допустимі параметри поля		Допустима поверхнева щільність потоку енергії (інтенсивність потоку енергії), Вт/м2
	за електричною складовою (Е), В/м	за магнітною складовою (Н), А/М	
Напруженість електромагнітного поля, 6 кГц .3 МГц	50	5	
1	2	3	4
3 МГц .30МГц	2	–	
30 МГц .5 ГГц	–	–	10
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру: УФ-С (220 .280 мм)			0,001
УФ-В (280 .320 мм)			0,01
УФ-А (320. 400 мм)			10,0
в інфрачервоній частині спектру: 0,76 . 10,0 мкм			35,0 .70,0
Напруженість електричного поля ВДТ			20 вВ/м

Визначення категорії праці за енерговтратами з урахуванням особливості роботи за комп’ютером наведено у табл. 6.5.

Таблиця 6.5. – Загальні енерговитрати організму

Категорія робіт	Дж/с (Вт)	Ккал/год	Характеристика робіт	Професії (приклади)
Легка – Іа	105–140	90–120	Роботи, які виконуються сидячи та не потребують фізичного напруження	Управлінець, оператор ПК
Легка – Іб	141–175	121–150	Роботи, які виконуються сидячи, стоячи – супроводжуються деяким фізичним напруженням	Інженерно-технічний персонал
Середньої важкості – ІІа	176–232	151–200	Роботи, які пов'язані з постійним ходінням, переміщенням дрібних (до 1кг) виробів або предметів у положенні стоячи або сидячи – потребують певного фізичного напруження	Працівники ремонтних майстерень

6.4 Розрахунок звуко поглинаючого облицювання

Провести розрахунок звукопоглинаючого облицювання поверхонь приміщення з комп’ютерною технікою. Для облицювання поверхонь використано звукопоглинаючий матеріал – плити ПА/С.

В приміщенні працюють програмісти. Шум в приміщенні створюють 3 джерела: 2 ПК та принтер.

Об’єм приміщення (V) становить – 96 м^3.

Для розрахунку звукопоглинаючого облицювання необхідно знати акустичні характеристики приміщення:

В – постійна приміщення, м^2;

А – еквівалентна площа звукопоглинання, м^2;

α – середній коефіцієнт звукопоглинання.

Постійна приміщення (В) акустично необробленого приміщення визначається:

$$B = B_n \cdot \mu,$$

де B_n – постійна приміщення на середньогеометричній частоті 1000 Гц, м^2,

μ – частотний множник.

Значення постійної приміщення B_n обирається з табл. 6.6.

Таблиця 6.6 – Значення постійної приміщення

Приміщення	B_n, $м^2$
З невеликою кількістю людей (цехи заводів)	V/20
З жорсткими меблями і великою кількістю людей або з невеликою кількістю людей та м'якими меблями (лабораторії, кабінети)	V/10
З великою кількістю людей і м'якими меблями (кімнати управління, зали конструкторський бюро, учбові аудиторії)	V/6
Приміщення зі звукопоглинаючим облицюванням стелі і частини стін	V/1,5

Для даного приміщення

$$B_n = \frac{V}{6},$$

де V – об'єм приміщення.

Отже,

$$B_n = \frac{96}{6} = 16.$$

Значення частотного множника μ обирається з табл. 6.7.

Таблиця 6.7 – Значення частотного множника μ

Об'єм приміщення, $м^3$	Частотний множник μ на середньогеометричних частотах октавних смуг, Гц							
	63	125	250	500	1000	2000	4000	8000
<200	0,8	0,75	0,7	0,8	1	1,4	1,8	2,5
200...1000	0,65	0,62	0,64	0,75	1	1,5	2,4	4,2
>1000	0,5	0,5	0,55	0,7	1	1,6	3	6

Значення частотного множника μ залежать від об'єму приміщення. У нашому випадку об'єм приміщення $V = 96\ м^3$, отже $96\ м^3 < 200\ м^3$, тому значення частотного множника взяті для об'єму приміщення $< 200\ м^3$.

За знайденими значеннями постійних приміщення для кожної октавної смуги обчислюється еквівалентна площа звукопоглинання:

$$A=B/(B/S+1),$$

де S – загальна площа огороджуючих поверхонь приміщення, м^2.

Отже,

$$S = 2 \cdot (a \cdot h + b \cdot h + a \cdot b) = 2 \cdot (6 \cdot 4 + 4 \cdot 4 + 6 \cdot 4) = 128\ м^2.$$

Середній коефіцієнт звукопоглинання α в приміщенні до його акустичної обробки обчислюється за формулою:

$$\alpha = \frac{B}{B + S}.$$

Зона відбитого звуку визначається величиною граничного радіусу $r_{гр}$, тобто, відстанню від розповсюджувача шуму, на якій рівень звукового тиску відбитого звуку дорівнює рівню звукового тиску прямого звуку, що розповсюджується.

$$r_{гр} = 0.2 * \sqrt{\frac{B_{n2}}{n}}$$

де B_{n2} – постійна приміщення на середньогеометричній частоті 8000 Гц,

n – кількість однакових розповсюджувачів шуму,

$$B_{n2} = B_n \cdot \mu_{8000}; = 16 \cdot 2{,}5 = 40 м^2$$

=0,2*4,472=0,89 м^2.

Максимальне зниження рівня звукового тиску ΔL (дБ), в кожній октавній смузі при застосуванні звукопоглинаючого облицювання в розрахунковій точці, що розміщена в зоні відбитого звуку обчислюється за формулою:

$$\Delta L = 10 \lg\left(\frac{B'}{B}\right),$$

де B' – постійна приміщення після облаштування в ньому звукопоглинаючих конструкцій, м^2.

$$B' = \frac{A_1 + \Delta A}{1 - \alpha_1}.$$

Еквівалентна площа звукопоглинання поверхнями, не зайнятими звукопоглинаючим облицюванням, м^2:

$$\alpha_1 = \frac{A_1 + \Delta A}{S},$$

де α_1 – середній коефіцієнт звукопоглинання акустично обробленого приміщення,

ΔA – величина сумарного додаткового поглинання, що вноситься конструкцією звукопоглинаючого облицювання або штучними поглиначами.

$$\Delta A = \alpha_{обл} * S_{обл} + A_{шт} * n_{шт}$$

$\alpha_{обл}$ – реверберaційний коефіцієнт звукопоглинаючої конструкції облицювання,

$S_{обл}$ – площа облицьованих поверхонь, м^2 (стіни та стеля),

$A_{шт}$ – еквівалентна площа одного звукопоглинаючого штучного поглинача, м^2;

$n_{шт}$ – кількість штучних поглиначів.

Значення реверберaційного коефіцієнта $\alpha_{обл}$ для різних звукопоглинаючих матеріалів наведено в табл. 6.8.

Таблиця 6.8 – Значення реверберaційного коефіцієнта $\alpha_{обл}$ для різних звукопоглинаючих матеріалів

Марка звукопоглинаючого матеріалу	63	125	250	500	1000	2000	4000	8000
Плита ПА/О	0,02	0,03	0,17	0,68	0,98	0,86	0,45	0,20
Вініпор	0,01	0,15	0,25	0,56	0,85	1	1	1
Плита ПА/С	0,02	0,05	0,21	0,66	0,91	,95	0,89	0,7
Плита «АКМІГРАН»	0,02	0,11	0,3	0,85	0,9	0,78	0,72	0,58
Плита АГП гіпсова	0,03	0,09	0,26	0,54	0,94	0,67	0,4	0,3
Мати із супертонкого скловолокна	0,1	0,25	0,7	0,98	1	1	1	0,95

$$S_{обл} = 2 \cdot (6 \cdot 4 + 4 \cdot 4) + 6 \cdot 4 = 2 \cdot 40 + 24 = 80 + 24 = 104 \ м^2.$$

В приміщенні штучні поглиначі не використовувались. Результати обчислень наведені у табл. 6.9.

Таблиця 6.9 – Результати обчислень максимального зниження рівня звукового тиску

Середньогеометрична астота, Гц.	63	125	250	500	1000	2000	4000	8000
μ, Гц	0.80	0.75	0.70	0.80	1	1.40	1.80	2.50
В, $м^2$	9.6	9	8.4	9.6	12	16.8	21.6	30
А, $м^2$	8.82	8.31	7.79	8.82	10.8	14.54	18	23.48
α	0.076	0.071	0.067	0.076	0.091	0.119	0.143	0.179
$\alpha_{обл}$	0.02	0.05	0.21	0.66	0.91	0.95	0.89	0.70
ΔA	1.68	4.20	17.64	55.44	76.44	79.80	74.76	58.80
A_1	1.82	1.70	1.60	1.82	2.18	2.86	3.43	4.30
α_1	0.032	0.055	0.178	0.530	0.728	0.765	0.724	0.584

Використання звукопоглинаючого облицювання доцільне, коли в зоні відбитого звуку потрібно знизити рівень звуку не більше ніж на 10..12 дБ., а на робочих місцях – на 4..5 дБ.

7 ОХОРОНА НАВКОЛИЩНЬОГО СЕРЕДОВИЩА

7.1 Сутність охорони навколишнього середовища

Охорона навколишнього середовища (ОНС) 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 [57].

Дві основні проблеми, які вирішуються у процесі ОНС, – це:

– забезпечення чистоти та її підтримання, що стосується природних систем;

– організація раціонального природокористування.

Природокористування можна вважати раціональним, якщо воно охоплює ряд певних заходів для захисту та відтворення природних ресурсів. Це сприяє гармонічній взаємодії людини та природи.

Ці завдання вирішуються такими шляхами:

– ресурси розподіляють оптимально на конкретні цілі для господарства;

– використовують ресурсозберігаючі технології;

– проводяться заходи для поповнення природних багатств.

Забезпечення та підтримка чистоти природних екологічних систем (водного середовища, повітряного басейну, ґрунтових покровів тощо) є ще одним важливим завданням ОНС. У результаті цього населення отримує екологічно чисту воду, їжу, повітря. Це сприяє збереженню високих показників рівня здоров'я суспільства та його довголіттю.

Будь-яка економічна діяльність забруднює природнє середовище. Як результат – дефіцит та забрудненні повітряні ресурси, території, вода. Сьогодні забруднення досягло надзвичайного рівня і набуло загрозливого характеру для оточуючого середовища та безпосередньо людини.

Значні обсяги викидів та відходів є основною з причин забруднення

природного оточення. До них відносять: не використані у виробництві матеріали, що не підлягають подальшій переробці, або продукти, що відслужили свій термін споживання, різні пакувальні матеріали, всілякі відвали та терикони породи тощо.

Величезна кількість відходів є результатом значного збільшення обсягів виробництва. За підрахунками фахівців, за останнє століття світове промислове виробництво збільшилося більш як у 50 разів, причому 4/5 цього приросту припадає на кінець XX століття. Як слушно зазначають американські економісти, збільшення валового національного продукту є одночасно збільшенням "валового національного сміття". Проблема утилізації відходів стала сьогодні проблемою глобального масштабу.

Не менш значною причиною забруднення є широке використання забруднюючих технологій, які для багатьох підприємств є вигіднішими, ніж екологічно чисті, в силу більшої дешевизни виробництва продукції і менших витрат товарообігу.

Завданням держави є розробка на основі наукових досліджень програми заходів у галузі охорони навколишнього середовища і формування достатніх фінансових коштів на екологічні потреби й покриття екологічних витрат. В Україні функції з охорони природи і контролю за дотриманням екологічних нормативів здійснює Міністерство екології, а також Державна екологічна інспекція.

У загальному випадку до екологічних витрат належать:

– економічна оцінка негативних наслідків забруднення і порушення компонентів цілісного природного середовища (економічні збитки, шкода);

– поточні і капітальні витрати на здійснення природоохоронних заходів;

– платежі за викиди забруднюючих речовин і розміщення відходів у навколишньому середовищі;

– платежі за використання природних ресурсів;

– штрафи за екологічні правопорушення і платежі на відшкодування заподіяних збитків;

– система компенсаційних виплат населенню, що мешкає в несприятливих екологічних умовах.

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.

7.2 Забруднення навколишнього середовища

Ніхто з учених так і не придумав можливості сконструювати повністю безпечну для навколишнього середовища техніку. Стара техніка нерідко стає причиною забруднення навколишнього середовища. Відправлена на звалище вийшла з ладу електротехніка цілком здатна дійсно завдати нашій планеті великої шкоди. Електротехніка в наш час не купується на довгі роки, тому що на зміну техніці майже відразу приходять нові моделі техніки. Рідко буває, що обладнання в компаніях в наші дні взагалі ремонтується – зазвичай замість негайно купується нове обладнання. Не секрет, що якщо зламане оснащення міститься в неправильних умовах, то воно загрожує завдати шкоди чистоті довкілля. Без сумніву, правильне поводження з непотрібною технікою – це гарантія того, що екології електронне сміття не завдає шкоди. Необхідно усвідомлювати, як захистити навколишнє середовище.

Всі, хто застосовують лампи денного світла для освітлення офісів, магазинів, кафе та ін. приміщень, зазвичай не підозрюють про можливі проблеми, які виникнуть під час перевірок. Люмінесцентні лампи містять небезпечні для здоров'я людини хімічні речовини. У кожної люмінесцентної лампи знаходиться від 20 до 500 мг ртуті (в залежності від типу і технології), тому вони відносяться до першого, тобто найвищого класу небезпеки. Також не вірно утилізована оргтехніка несе в собі небезпеку навколишньому середовищі.

7.3 Розробка заходів щодо зменшення забруднення навколишнього середовища

Компанії використовують дуже багато ноутбуків, клавіатур, мишок, батарейок, бумаги та т.п.

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 відпрацьоване свій термін виріб поширюється на міське звалище.

Завдяки впливу вологи, що містяться в електронних компонентах метали (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 з природоохоронних, так і з економічних міркувань.

ВИСНОВКИ

Мета, яка була поставлена в кваліфікаційній роботі, а саме: підвищення достовірності оцінювання якості ПЗ з розпізнавання номерних знаків автомобілів, досягнута в повному обсязі. Усі завдання, поставлені при підготовці кваліфікаційної роботи, були успішно виконані.

Було проведено аналіз: сучасних методів та алгоритмів, математичних моделей та існуючого ПЗ для розпізнавання номерних знаків автомобілів; існуючих метрик та моделей для оцінювання якості ПЗ. Була обґрунтована необхідність дослвдження.

Удосконалено математичну модель для оцінювання якості ПЗ для розпізнавання номерних знаків автомобілів за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дозволило підвищити достовірність оцінювання якості ПЗ розпізнавання номерних знаків в порівнянні з існуючою моделлю

Розроблена програма для оцінювання якості ПЗ розпізнавання номерних знаків на основі удосконаленої математичної моделі, яка показала повну працездатність та ефективність, та дозволила автоматизувати та скоротити час відповідних розрахунків.

Виконано спецрозділи з охорони праці, охорони навколишнього середовища та економічний розділи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Как построить алгоритм распознавания номеров с точностью более 95%. URL: https://security-news.today/kak-postroit-algoritm-raspoznavaniya-nomerov-s-tochnostyu-bolee-95/ (дата звернення: 03.09.2021).

2. Некоторые аспекты автоматического распознавания автомобильных номеров. URL: https://old.nordavind.ru/node/110 (дата звернення: 03.09.2021).

3. Распознаватель всех параметров автомобиля. URL: https://number-ok.com/ (дата звернення: 03.09.2021).

4. ПО "Авто Орион Про". URL: https://bolid.ru/production/orion/po-orion/po-arm/orion_avto.html (дата звернення: 03.09.2021).

5. Smart Engines представила технологию распознавания автомобильных номеров для мобильных устройств. URL: https://smartengines.ru/news44/ (дата звернення: 03.09.2021).

6. Nomeroff Net. Automatic numberplate recognition system. URL: https://nomeroff.net.ua/ (дата звернення: 03.09.2021).

7. Распознавание номеров. Практическое пособие. Часть 1. URL: https://habr.com/ru/post/432444/ (дата звернення: 03.09.2021).

8. Распознавание номеров. Как мы получили 97% точности для Украинских номеров. Часть 2. URL: https://habr.com/ru/post/439330/ (дата звернення: 03.09.2021).

9. Nomeroff Net. Automatic numberplate recognition system. URL: https://github.com/ria-com/nomeroff-net (дата звернення: 03.09.2021).

10. Відомості про транспортні засоби та їх власників. URL: https://data.gov.ua/dataset/06779371-308f-42d7-895e-5a39833375f0 (дата звернення: 03.09.2021).

11. Метрики качества программного обеспечения. URL: https://coderlessons.com/tutorials/kachestvo-programmnogo-

obespecheniia/izuchite-upravlenie-kachestvom-programmnogo-obespecheniia/metriki-kachestva-programmnogo-obespecheniia (дата звернення: 07.09.2021).

12. ISO 9001: 1994 Quality systems – Model for quality assurance in design, development, production, installation and servicing.

13. ISO 8402: 1994 Quality management and quality assurance.

14. 1061-1998 IEEE Standard for Software QualityMetrics Methodology.

15. ГОСТ 28806-90. Качество программных средств. Термины и определения.

16. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.

17. Сборник действующих международных стандартов ИСО серии 9000: Т.1, 2, 3. М.: ВНИИКИ, 1998.

18. Поморова О.В., Говорущенко Т.О., Тарасек С.Я. Аналіз та опрацювання метрик якості програмного забезпечення на етапі проектування. *Вісник Хмельницького національного університету*. 2010. №1. С. 54-62.

19. Моисеев С.И., Черная Ю.В., Паршина Е.В. Модель оценки качества программного обеспечения, основанная на методе Раша оценки латентных переменных. *Вестник ВГУ, Серия: Системный анализ и информационные технологии*. 2016. № 1 С. 102-109.

20. Клюев Е.И., Гриненко Е.А. Подход к оценке качества программных средств. *Інженерія програмного забезпечення*. 2014. № 3 (19). С. 5-14.

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

22. Полтєв О.С., Макарова Л.М., Фаріонова Т.А. Математична модель для оцінювання якості програмного забезпечення розпізнавання номерних знаків. *Матеріали IV Всеукраїнської науково-практичної інтернет-*

конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць / Під редакцією Г.О. Райко. Херсон: Видавництво ФОП Вишемирський В.С. 2021. с. 256-257.

23. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.:Техносфера, 2005. 1072 с.

24. OpenCV. URL: https://opencv.org/ (дата звернення: 07.09.2021).

25. Яне Б. Цифровая обработка изображений. Пер. с англ. А.М. Измайловой. М.: Техносфера, 2007. 584 с.

26. Пару слов о распознавании образов. URL: https://habr.com/ru/post/208090/ (дата звернення: 07.09.2021).

27. Метод k-средних (K-Means). URL: https://www.helenkapatsa.ru/mietod-k-sriednikh/ (дата звернення: 07.09.2021).

28. Алгоритм AdaBoost. URL: https://habr.com/ru/company/otus/blog/503888/ (дата звернення: 07.09.2021).

29. Краткий обзор алгоритма машинного обучения. Метод Опорных Векторов (SVM). URL: https://habr.com/ru/post/428503/ (дата звернення: 07.09.2021).

30. P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004.

31. Работа алгоритма Viola Jones. URL: http://habrahabr.ru/post/134857/http://habrahabr.ru/post/134857/ (дата звернення: 07.09.2021).

32. R. Lienhart, A. Kuranov, V. Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. MRL Technical Report, 2002.

33. Цветков А.А., Шорох Д.К., Зубарева М.Г. Алгоритмы распознавания объектов. *Технические науки: проблемы и перспективы: материалы IV Междунар. науч. конф.* (г. Санкт-Петербург, июль 2016 г.). Санкт-Петербург: Свое издательство, 2016. С. 20-28.

34. NVIDIA DIGITS. Interactive Deep Learning GPU Training System. URL: https://developer.nvidia.com/digits (дата звернення: 07.09.2021).

35. Техническое зрение. URL: http://wiki.technicalvision.ru/index.php/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0 (дата звернення: 07.09.2021).

36. ISO/IEC Standard No. 9126: Software engineering – Product quality; Parts 1–4. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 2001-2004.

36. Тестирование ПО. Самые важные метрики QA. URL: https://doitsmartly.ru/all-articles/sw-testing/133-the-most-important-metrics-in-qa.html (дата звернення: 11.09.2021).

37. Albrecht J., Gaffney J.E. Software function, source lines of codes, and development effort prediction: a software science validation. IEEE Trans Software Eng. SE-9. 1983. P. 639–648.

38. Function Point Counting Practices Manual, Release 4.2, IFPUG, 2004.

39. Електронний конспект лекцій з дисципліни «Емпіричні методи програмної інженерії». URL: http://tc.kpi.ua/content/kurs/ (дата звернення: 11.09.2021).

40. Memon, Mashooque Ahmed et al. A Regression Analysis Based Model for Defect Learning and Prediction in Software Development. *Mehran University Research Journal of Engineering and Technology*, [S.l.], v. 40, n. 3, p. 617- 629, july 2021.

41. Muhammad Dhiauddin Mohamed Suffian, Suhaimi Ibrahim. A Prediction Model for System Testing Defects using Regression Analysis. *International Journal of Soft Computing And Software Engineering (JSCSE)*. Vol.2, o.7, 2012. p. 55-68.

42. Приходько С.Б., Макарова Л.Н. Доверительный интервал нелинейной регрессии времени восстановления работоспособности

устройств терминальной сети. *Восточно-Европейский журнал передовых технологий*. 2014. № 3(4). С. 26-31.

43. Вентцель Е.С. Теория вероятностей: Учеб. для вузов. М.: Высш. шк., 1999. 576 с.

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

45. Prykhodko S., Prykhodko N., Makarova L., Pugachenko K. Multivariate outlier detection technique based on normalizing transformations for non-Gaussian data. *«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.

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

47. Магнус Я.Р., Катышев П.К., Пересецкий А.А. Эконометрика. Начальный курс. М.: Дело, 2004. 576 с

48. Уніфікована мова моделювання. URL: https://bibliofond.ru/view.aspx?id=446563#1 (дата звернення: 07.11.2021).

49. Діаграма варіантів використання. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28086 (дата звернення: 07.11.2021).

50. Файл формата CSV — что это? URL: https://filesreview.com/ru/info/csv (дата звернення: 07.11.2021).

51. Діаграма класів. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 (дата звернення: 07.11.2021).

52. Діаграма послідовності. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?r=16538 (дата звернення: 07.11.2021).

53. Java и вы. Загрузите сегодня. URL: https://www.java.com/ru/ (дата звернення: 11.11.2021).

54. IntelliJ IDEA: Функциональная и эргономичная IDE для профессиональной разработки. URL: https://www.jetbrains.com/ru-ru/idea/ (дата звернення: 11.11.2021).

55. JetBrains. Все необходимые инструменты для разработчиков и команд. URL: https://www.jetbrains.com/ru-ru/ (дата звернення: 11.11.2021).

56. Про охорону праці: Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text (дата звернення: 15.11.2021).

57. Про охорону навколишнього природного середовища: Закон України від 26.06.91р. № 1268-XII. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення: 15.11.2021).

Додаток А Технічне завдання

Вступ

Назва розроблюваного проекту: «Програма для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів». Область застосування – підприємство, яке займається розробкою ПЗ.

1. Підстави для розробки

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

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням ПЗ є спрощення бізнес-процесів підприємства, яке займається розробкою ПЗ, покращення та легкість обробки масивів статистичних даних, підвищення достовірності та полегшення оцінювання якості ПЗ розпізнавання номерних знаків автомобілів.

2.2 Функціональне призначення

Функціональним призначенням ПЗ є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою нормалізуючого перетворення на основі десяткового логарифму;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

– побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, а саме загальної кількості дефектів, знайдених у ПЗ, в залежності від кількості рядків коду ПЗ, KLOC.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.3.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1) Загальні функції:

– можливість завантажити файл з вихідними емпіричними даними у форматі CSV;

– можливість змінити файл з вихідними емпіричними даними;

– розрахунок параметрів для вихідних вибірок емпіричних даних;

– перевірка вихідних вибірок емпіричних даних на нормальність закону розподілу;

– нормалізація вихідних вибірок емпіричними даними за допомогою нормалізуючого перетворення на основі десяткового логарифму;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка нормалізованих вибірок на нормальність закону розподілу;

– перевірка вибірок даних на наявність викидів за допомогою квадрату відстані Махаланобіса;

– побудова лінійного рівняння регресії;

– перевірка залишків лінійного рівняння регресії на нормальність закону розподілу;

– побудова довірчого інтервалу лінійного рівняння регресії;

– побудова інтервалу прогнозування лінійного рівняння регресії;

– побудова нелінійного рівняння регресії;

– побудова довірчого інтервалу нелінійного рівняння регресії;

– побудова інтервалу прогнозування нелінійного рівняння регресії;

– оцінювання загальної кількості дефектів, знайдених у ПЗ, в залежності від кількості рядків коду ПЗ, KLOC;

– експорт результатів у файл у форматі CSV.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатура або миша). Дані вводяться через завантаження файлу у вікні програми та за допомогою ручного вводу.

Виведення даних здійснюється за допомогою пристрою виведення даних – монітору комп’ютера.

Вхідні дані:

- файл у форматі CSV;
- параметри для перевірки вибірок даних на нормальність закону розподілу (α);
- параметри для перевірки вибірок даних на наявність викидів за допомогою квадрату відстані Махаланобіса (α, m).

Вихідні дані:

1) Розраховані параметри для вихідної вибірки:

– кількість значень вибірки;

– вибіркове середнє;

– дисперсія;

– середньоквадратичне відхилення;

– асиметрія;

– квадрат асиметрії;

– ексцес;

– мінімальне значення;

– максимальне значення;

– обчислене значення χ^2.

2) Розраховані параметри для перевірки вибірок даних на нормальність закону розподілу:

– χ^2 критичне.

3) Розраховані параметри для нормалізованої вибірки:

– кількість значень вибірки;

– вибіркове середнє;

– дисперсія;

– середньоквадратичне відхилення;

– асиметрія;

– квадрат асиметрії;

– ексцес;

– мінімальне значення;

– максимальне значення;

– обчислене значення χ^2.

4) Розраховані параметри для перевірки вибірок даних на наявність викидів за допомогою квадрату відстані Махаланобіса:

– квадрат відстані Махаланобіса d_i^2 для кожної точки вибірки;

– значення χ^2 критичне.

5) Графік для побудованого лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього виводиться на монітор користувача. Параметри лінійної регресії записуються у вихідний файл.

6) Графік для побудованого нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього виводиться на монітор користувача. Параметри нелінійної регресії записуються у вихідний файл.

7) Значення оцінки загальної кількості дефектів, знайдених у ПЗ, в залежності від кількості рядків коду ПЗ, KLOC.

3.2 Вимоги до надійності

Програмний продукт повинен нормально функціонувати при безперебійній роботі ПК. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Відмови програми можливі внаслідок некоректних дій користувача при взаємодії з програмою. Щоб зменшити кількість відмов програми за вказаною вище причиною слід забезпечити зрозумілість програми, через надання підказок кінцевому користувачеві у разі невірних дій.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навички в користуванні комп'ютером. Комп'ютер призначений для роботи в закритому опалювальному приміщені при наступних умовах навколишнього середовища:

- температура повітря від +10^{o}C до +30^{o}C;
- 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 користувача повинна задовольняти вказаним мінімальним вимогам для коректного запуску програми:

– CPU: Intel Pentium 4405Y (1,2 ГГц);
– RAM: 1 GB;
– 500 Mb вільного місця на диску.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування системи необхідно використовувати: ОС Windows версії не нижче 7.

3.6 Вимоги до маркування та пакування

Додаток для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, буде упаковано в електронний архів і мати найменування «quality_lpr.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

Програмна документація повинна містити:

– технічне завдання;
– текст програми;
– опис програми;
– інструкцію користувача;
– програму і методику випробувань ПЗ.

5. Техніко-економічні показники

– Не розраховувалися.

6. Стадії та етапи розробки

Стадії та етапи розробки представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Термін виконання робіт	
		Початок етапу	Кінець етапу
1. Технічне завдання	1.1Обгрунтування необхідності розробки програми	06.09.21	11.09.21
	1.2 Розробка технічного завдання	11.09.21	16.09.21
	1.3 Затвердження технічного завдання	16.09.21	28.09.21
2. Ескізний проект	2.1 Розробка ескізного проекту	28.09.21	08.10.21
	2.2 Затвердження ескізного проекту	08.10.21	15.10.21
3. Технічний проект	3.1 Розробка технічного проекту	15.10.21	21.10.21
	3.2 Затвердження технічного проекту	21.10.21	01.11.21
4. Робочий проект	4.1 Розробка робочого проекту	01.11.21	07.11.21
	4.2 Затвердження робочого проекту	07.11.21	15.11.21
	4.3 Розробка документації	15.11.21	20.11.21

7. Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини ПЗ для оцінювання якості ПЗ розпізнавання номерних знаків автомобілі, відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом готового програного проекту документують за допомогою протоколу проведення випробувань. У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації – замовника та організації – розробника. Розробник повинен протягом не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б Текст програми

Файл ImportController.java

```
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();
          }
```

Файл ImportModel.java

```
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;
    }
  }
}
```

Файл ImportView.java

```
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;
          }
        }
```

Файл Application.java

```
        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 В Опис програми

Програма для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, яка розроблялась у рамках кваліфікаційної роботи, спрямована на використання на підприємствах, які займаються розробкою ПЗ.

Дана програма розроблена з метою спрощення бізнес-процесів підприємства, яке займається розробкою ПЗ, покращення та легкість обробки масивів статистичних даних, підвищення достовірності та полегшення оцінювання якості ПЗ розпізнавання номерних знаків автомобілів.

Функціональним призначенням програми є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою нормалізуючого перетворення на основі десяткового логарифму;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

– побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, а саме загальної кількості дефектів, знайдених у ПЗ, в залежності від кількості рядків коду ПЗ, KLOC.

Реалізація програми була виконана об`єктно-орієнтованою мовою програмування Java у інтегрованому середовищі розробки (IDE) IntelliJ IDEA.

При розробці програми значну увагу було приділено користувацькому інтерфейсу. Важливим завданням було створити його якомога простішим у використанні і в той самий час багатофункціональним і зручним. Всі графічні елементи і кольорові гами було використано відповідно до загальноприйнятих.

Було проведено повне функціональне тестування, а також навантажувальне тестування. Усі виявлені недоліки програми були зафіксовані в протоколах і усунуті розробником до моменту впровадження програми у дію. Наступні випробування були успішними та показали повну готовність програми до використання.

Додаток Г Інструкція користувача

Програма автоматизованої обробки інформації для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, яка розроблялась у рамках кваліфікаційної роботи, запакована в архів з назвою quality_lpr.rar.

Після розпакування архіву необхідно зайти в папку QUALITY_LPR та вікрити програму Quality_Lpr. З’явиться початкове вікно програми, зображене на рис. Г.1.

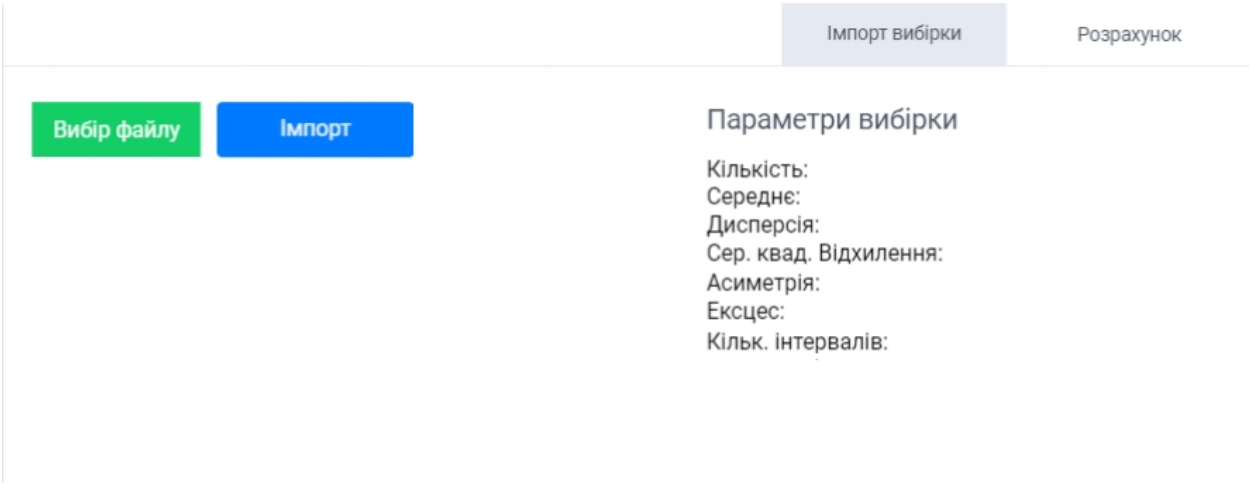

Рисунок Г.1 – Початкове вікно програми

Для того щоб вибрати файл для обробки необхідно натиснути на кнопку “Вибір файлу” (рис. Г.1), після чого у стандартному діалозі ОС для вибору файлу вибрати файл. Вікно вибору файлу зображене на рис. Г.2.

Для того щоб імпортувати обраний файл необхідно натиснути на кнопку “Імпорт”, після чого файл буде імпортовано, розраховано основні параметри вибірки, і досліджено на відповідність закону розподілу (рис. Г.3).

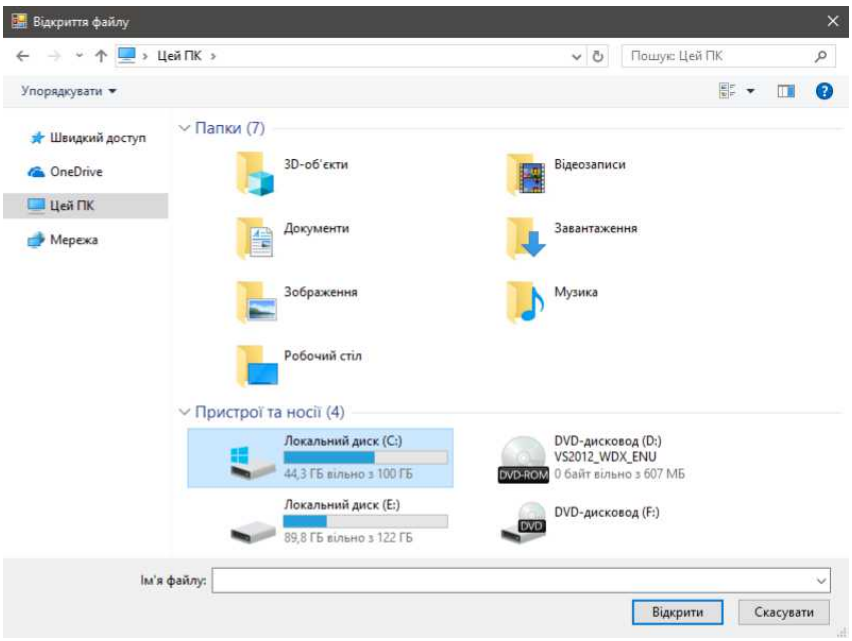

Рисунок Г.2 – Вибір файлу з вихідними параметрами

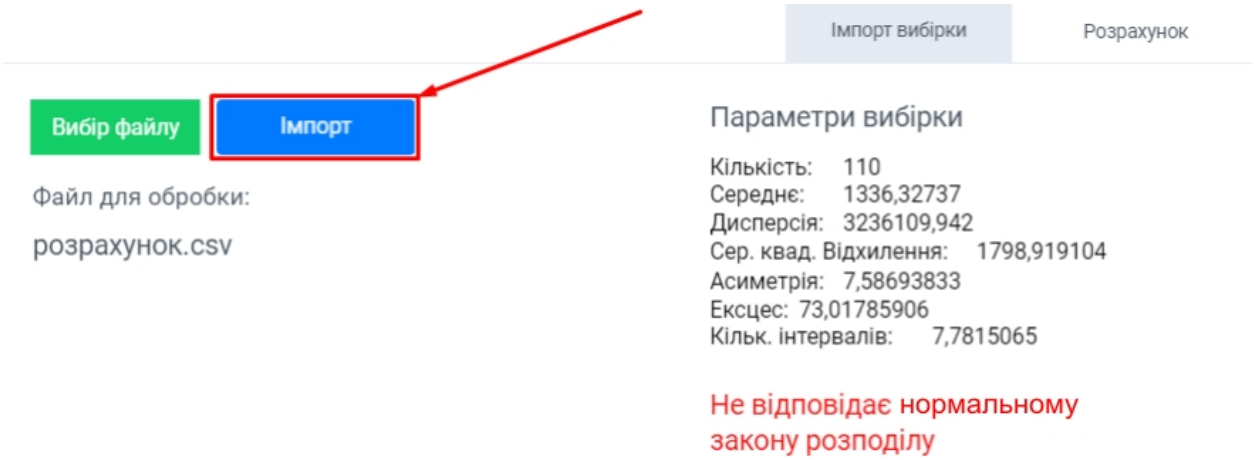

Рисунок Г.3 – Імпорт файлу

Для того щоб переходити між вкладками імпорту і розрахунку необхідно натиснути на відповідну вкладку у правому верхньому кутку головної форми програми (рис. Г.4).

Рисунок Г.4 – Вкладки “Імпорт вибірки” та “Розрахунок”

Для того щоб виконати нормалізацію та розрахунки параметрів нормалізованої вибірки, а також інтервалів лінійного рівняння, необхідно на формі розрахунку натиснути на кнопку “Розрахувати” (рис. Г.5).

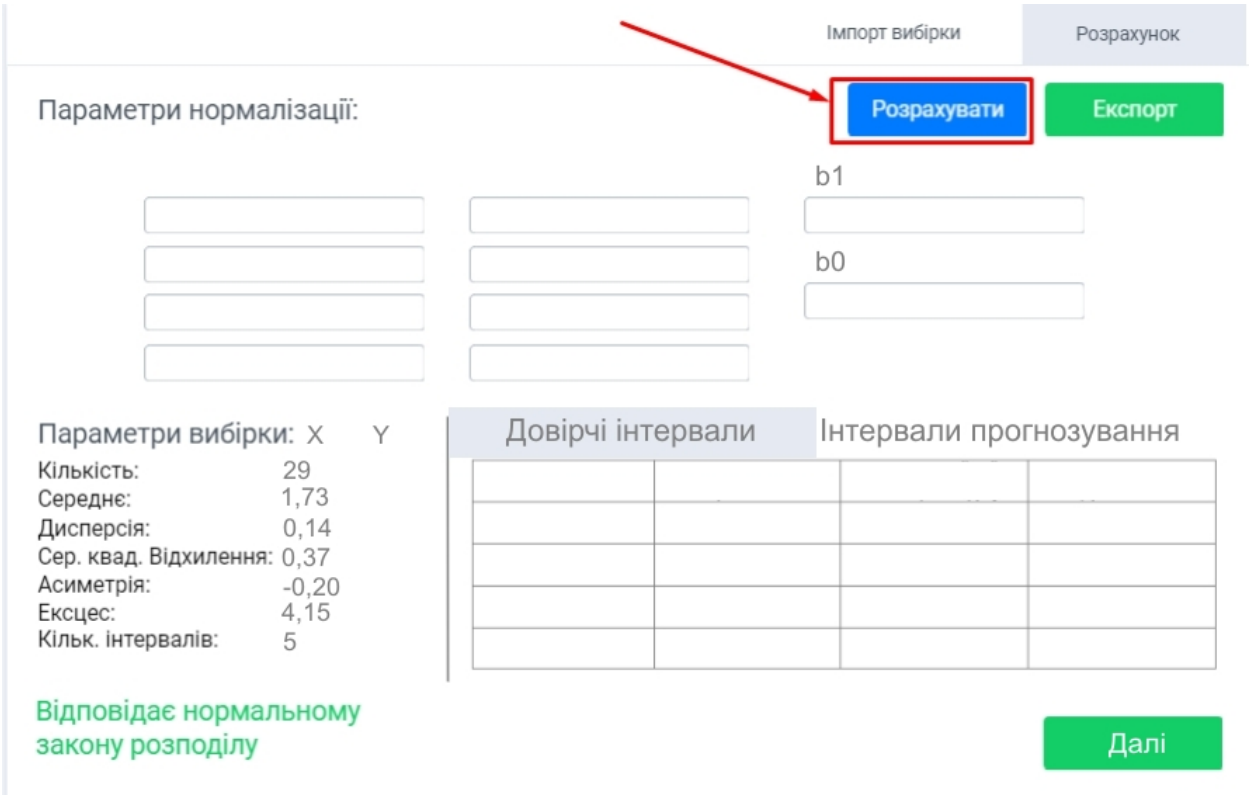

Рисунок Г.5 – Нормалізація та розрахунок параметрів

Для отримання нелінійної моделі потрібно натиснути на кнопку "Далі" у правому нижньому куті форми.

Для того щоб виконати експорт отриманих результатів необхідно на формі розрахунку натиснути на кнопку “Експорт”, після чого у стандартному діалозі ОС вказати ім`я файлу.

Додаток Д Програма і методика випробувань ПЗ

Програма для оцінювання якості ПЗ розпізнавання номерних знаків автомобілів, яка розроблялась у рамках кваліфікаційної роботи, пройшла ряд випробувань. Усі виявлені недоліки ПЗ були зафіксовані й усунуті до моменту впровадження ПЗ в дію. Випробування ПЗ проводилося на цільовому обладнанні в тій конфігурації, яка заявлена в технічному завданні (див. Додаток А).

Нижче наведено список випробувань, які проводились для виявлення недоліків програми та описано послідовність дій для кожного з них, показано результати реагування ПЗ.

1. Випробування на завантаження некоректного формату файлу.

У головному вікні програми натиснули кнопку «Вибір файлу». Зліва було наведено допустимі параметри. Було вибрано картинку з недопустимим розширенням .jpg. У результаті отримали повідомлення системи про недопустимий формат файлу та можливість вибрати інший файл. Випробування пройдено успішно.

2. Випробування на завантаження коректного формату файлу.

У головному вікні програми натиснули кнопку «Вибір файлу». Зліва було наведено допустимі параметри. Було вибрано фацл з розширенням .csv. У результаті файл успішно завантажився та нижче було виведено його назву. Випробування пройдено успішно.

3. Випробування на розрахунок параметрів без завантаження файлу.

У головному вікні програми не вибрали жодного файлу. Натиснули кнопку «Розрахунок». У результаті отримали повідомлення системи про необхідність вибрати файл. Випробування пройдено успішно.

4. Випробування на розрахунок параметрів після коректного завантаження файлу.

Після коректного завантаження файлу натиснули «Розрахунок».У відповідних текстових блоках з`явились розраховані параметри, нижче – можливість перевірити нормальність розподілу та перехід до нормалізації. Випробування пройдено успішно.

5. Випробування на побудову графіку без попереднього розрахунку параметрів лінійного рівняння регресії.

У вікні «Рівняння регресії» натискаємо кнопку побудувати графік без попереднього розрахунку параметрів. У результаті нічого не відбувається, а курсор при наведенні на кнопку стає забороненим, а кнопка неклікабельною. Випробування пройдено успішно.

6. Випробування на побудову графіку після розрахунку параметрів лінійного рівняння регресії.

У вікні «Рівняння регресії» натискаємо кнопку побудувати графік після розрахунку параметрів. У результаті графік успішно побудований. Випробування пройдено успішно.

Отже, усі наведені перевірки тестування показали очікуваний результат реагування програми на дії користувача. Можна зробити висновок, що ПЗ працює правильно, не допускаючи некоректних дій з боку користувача та у разі виникнення проблем інформує користувача про можливі шляхи їх вирішення.