МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

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

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, та її програмна реалізація

Виконав: студент групи 6151м

_____ Коваленко В.В.

(підпис, ПІБ)

Керівник роботи:

Доцент, к.т.н.

(посада, науковий ступень вчене звання)

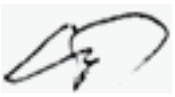
_____ Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»

Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
« 27 » жовтня 2025 р.

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

Студенту Коваленку Вадиму Володимировичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, та її програмна реалізація

Керівник роботи доцент, к.т.н. Пухалевич Андрій Володимирович

Затверджені наказом ректора № 1222-УЧ від « 15 » 10 2025 р.

2. Термін подання роботи: 08.12.2025 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; Розділ з розробки 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 1. Титульний слайд; 2. Актуальність теми; 3. Мета та завдання дослідження; 4. Об'єкт та Предмет дослідження; 5. Наукова новизна та Практичне значення одержаних результатів; 6. Апробація результатів досліджень та Публікації; 7.Аналіз існуючих підходів та обґрунтування вибору метрик для оцінювання якості застосунків з відкритим кодом на Java; 8. Розробка нелінійних регресійних моделей для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate; 9. Рівняння нелінійних регресійних моделей; 10-11. Аналіз вимог до програмного забезпечення; 12. Розробка архітектури програмного забезпечення; 13. Розробка проєкту програмного забезпечення; 14. Статична модель даних; 15. Динамічна модель даних; 16. Вибір інструментів розробки; 17. Результати розробки програмного забезпечення; 18. Випробування програмного забезпечення; 19. Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 27.10.2025 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	28.10.2025	ДП*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	30.10.2025	ДП*
3.	Підготовка розділу (ів) за результатами власних досліджень	01.11.2025	ДП*
4.	Підготовка розділу з розробки програмного забезпечення	26.11.2025	
5.	Підготовка організаційно-економічного розділу	28.11.2025	
6.	Підготовка розділу з охорони праці	02.12.2025	
7.	Підготовка розділу з охорони навколишнього середовища	03.12.2025	
8.	Підготовка розділу ВИСНОВКИ	04.12.2025	
9.	Оформлення списку використаних джерел та додатків	05.12.2025	
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.)	08.12.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	12.12.2025	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	15.12.2025	
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	19.12.2025	

* - за результатами (розділ звіту) дослідницької практики (ДП), яка була з 01.09.2025 по 26.10.2025 р.

Студент _____ Коваленко В.В.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

РЕФЕРАТ

Коваленко Вадим Володимирович

Нелінійна регресійна модель для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, та її програмна реалізація

Кваліфікаційна (магістерська) робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2025 р.

Обсяг роботи: 122 стор., 12 табл., 18 рис., 28 використаних джерел, 5 додатків.

Актуальність теми: Дослідження присвячене актуальній проблемі оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. З огляду на розповсюдженість використання мови програмування Java та ORM-технологій, запропонований в даній роботі підхід до оцінювання якості на основі нелінійних регресійних моделей і метрик RFC, CBO та WMC сприяє підвищенню достовірності аналізу якості та вдосконаленню існуючих методів оцінювання застосунків.

Мета та завдання дослідження: Метою дослідження є підвищення достовірності оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Основними завданнями є аналіз існуючих підходів та моделей оцінювання якості застосунків, визначення та обґрунтування вибору метрик для застосунків з відкритим кодом на Java, побудова трьох нелінійних моделей для оцінювання якості, розробка програмного забезпечення на основі побудованих моделей та проведення тестування для перевірки надійності та достовірності.

Об'єкт дослідження: Об’єктом дослідження є процес оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Предмет дослідження: Предметом дослідження є три нелінійні регресійні моделі: RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Методи дослідження: Статистичні методи, методи теорії ймовірностей, регресійний аналіз, об’єктно-орієнтоване програмування.

Наукова новизна: Вдосконалено нелінійні регресійні моделі для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate за рахунок врахування особливостей взаємозв’язків між метриками RFC, CBO та WMC таких застосунків.

Практична значущість: Програмне забезпечення, яке реалізовує побудовані моделі, дозволить швидко та достовірно оцінювати якість застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Апробація результатів досліджень: Результати досліджень пройшли апробацію на VI Міжнародній науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи», яка проводилась 16-17 листопада 2025 року, організована Національним університетом кораблебудування імені адмірала Макарова.

Публікації: За результатами досліджень опубліковано одну наукову працю, а саме матеріали інтернет-конференції.

Ключові слова: Оцінювання якості застосунків, нелінійна регресійна модель, Java, фреймворк Hibernate.

ABSTRACT

Kovalenko Vadym

A nonlinear regression model for assessing the quality of open source Java applications created using the Hibernate framework, and its software implementation

Qualification (Master's) Thesis for obtaining a master's degree in the specialty 121 «Software Engineering». The Admiral Makarov National University of Shipbuilding. Mykolaiv, 2025.

Volume of work: 122 pages, 12 tables, 18 figures, 28 references, 5 appendices.

Relevance of the topic: The research is devoted to the topical problem of assessing the quality of open source Java applications developed using the Hibernate framework. Given the widespread use of the Java programming language and ORM technologies, the approach to quality assessment based on nonlinear regression models and the RFC, CBO, and WMC metrics proposed in this work contributes to improving the reliability of quality analysis and refining existing application assessment methods.

Purpose and objectives of the study: The purpose of the study is to improve the reliability of quality assessment of open source Java applications developed using the Hibernate framework. The main objectives are to analyze existing approaches and models for application quality assessment; to identify and justify the selection of metrics for open source Java applications; to construct three nonlinear models for quality assessment; to develop software based on the constructed models; and to conduct testing to verify their reliability and accuracy.

Object of the study: The object of the study is the process of assessing the quality of open source Java applications developed using the Hibernate framework.

Subject of the study: The subject of the study is three nonlinear regression models: RFC(CBO, WMC), CBO(WMC, RFC), and WMC(RFC, CBO) for

assessing the quality of open source Java applications developed using the Hibernate framework.

Methods of the study: statistical methods, probability theory methods, regression analysis, object-oriented programming.

Scientific novelty of the obtained results: Nonlinear regression models for evaluating open source Java applications created using the Hibernate framework have been improved by taking into account the peculiarities of the relationships between the RFC, CBO, and WMC metrics of such applications.

Practical significance of the obtained results: The software that implements the built models will enable quick and reliable assessment of the quality of open source Java applications created using the Hibernate framework.

Testing of the research results: The research results were tested at the VI International Scientific and Practical Internet Conference 'Information Technologies: Models, Algorithms, Systems', held on 16-17 November 2025, organised by the Admiral Makarov National University of Shipbuilding.

Publications: Based on the research results, one scientific publication was produced, namely the materials of the Internet conference.

Keywords: Application quality assessment, nonlinear regression model, Java, Hibernate framework.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CBO – Coupling Between Object Classes

RFC – Response for Class

WMC – Weighted Methods per Class

ВСТУП

Актуальність теми. У сучасному світі Java продовжує утримувати провідні позиції серед мов програмування завдяки своїй інноваційності, адаптивності та продуктивності. Згідно з цьогорічним звітом компанії Azul, майже 70% із понад двох тисяч опитаних IT-фахівців у всьому світі зазначили, що більше половини їхніх застосунків створено за допомогою Java або працюють на Java Virtual Machine (JVM) [1]. Враховуючи широку популярність Java, а також розповсюджене використання фреймворку Hibernate – одного з найпоширеніших інструментів для роботи з ORM (Object-Relational Mapping) – постає потреба у створенні ефективних засобів оцінювання якості програмного забезпечення, зокрема забезпечення з відкритим вихідним кодом.

Оцінювання якості програмного забезпечення є складним завданням інженерії програмного забезпечення, оскільки вимагає урахування багатьох факторів, що визначають надійність, ефективність та підтримуваність коду. Для розв’язання цієї задачі можуть використовуватися нелінійні регресійні моделі, побудовані на основі метрик RFC, CBO та WMC [2–4]. Одним із найрезультативніших підходів до оцінювання якості застосунків з відкритим кодом є метод [5], який передбачає обчислення меж довірчих та прогнозних інтервалів нелінійних регресій для показників RFC, CBO та WMC на рівні застосунку для тривимірних точок даних. Під час порівняльного аналізу з існуючими моделями [6], було виявлено значні відмінності у значеннях інтервалів показників RFC, CBO та WMC.

Оскільки порівнювані моделі побудовані для інших інтервалів, вони можуть призводити до зниження достовірності оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Таким чином, побудова та впровадження нелінійних регресійних моделей для такого оцінювання дозволить удосконалити існуючі підходи, що

сприятиме підвищенню надійності та продуктивності розроблюваних застосунків з відкритим кодом.

Мета дослідження. Метою дослідження є підвищення достовірності оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Для досягнення мети дослідження слід виконати наступні завдання:

1. Провести критичний аналіз існуючих підходів та моделей оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, а також визначити підходи до використання метрик RFC, CBO та WMC.

2. Обґрунтувати вибір метрик RFC, CBO та WMC для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, визначивши вплив основних параметрів на якість програмного забезпечення.

3. Зібрати дані метрик RFC, CBO та WMC із застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Виконати обробку даних, включно з видаленням викидів.

4. Розробити три нелінійні регресійні моделі RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) із застосуванням одновимірного нормалізуючого перетворення методом Бокса-Кокса для кожного параметру для оцінювання якості застосунків на Java, що створюються з використанням фреймворку Hibernate.

5. Використати метод найменших квадратів для оцінки параметрів трьох побудованих моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) та оцінити їх якість на основі таких показників, як R^2, MMRE, та PRED(0.25).

6. Протестувати три розроблені моделі RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) на вибірці застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate за допомогою побудованих довірчих інтервалів та інтервалів прогнозування.

Об'єкт дослідження. Об’єктом дослідження є процес оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Предмет дослідження. Предметом дослідження є три нелінійні регресійні моделі RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Методи дослідження. Статистичні методи, методи теорії ймовірностей, регресійний аналіз, об’єктно-орієнтоване програмування.

Наукова новизна. Вдосконалено нелінійні регресійні моделі для оцінювання застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate за рахунок врахування особливостей взаємозв’язків між метриками RFC, CBO та WMC таких застосунків.

Особистий внесок здобувача. Всі результати, викладені в роботі, отримані автором самостійно. У спільній публікації [7], підготовленій разом із науковим керівником, автору належить: проведення аналізу існуючих моделей оцінювання якості, збір та попередня обробка даних, розробка нелінійних регресійних моделей, отримання параметрів моделей та аналіз її якості.

Апробація результатів досліджень. Результати досліджень пройшли апробацію на VI Міжнародній науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи», яка проводилась 16-17 листопада 2025 року, організована Національним університетом кораблебудування імені адмірала Макарова.

Публікації. За результатами досліджень опубліковано одну наукову працю, а саме матеріали інтернет-конференції.

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТРИК ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA

1.1 Аналіз існуючих підходів та моделей оцінювання якості застосунків з відкритим кодом на Java

Оцінювання якості застосунків з відкритим кодом є необхідним та важливим кроком для забезпечення надійності, стабільності, безпеки та довготривалої підтримки застосунків. Забезпечується оцінювання якості застосунків шляхом аналізу програмного коду та, як наслідок, збір певних метрик на основі яких, базуються різні підходи та моделі оцінювання.

Серед найрозповсюдженіших метрик слід виокремити:

– Цикломатична складність (Cyclomatic Complexity) описує кількість незалежних маршрутів виконання програми, характеризуючи складність її логічної структури та час, необхідний для виконання.

– LOC (Lines of Code) відображає загальний обсяг коду, проте не враховує його структурні особливості.

– RFC (Response For Class) демонструє кількість методів, які можуть бути викликані через даний клас; надто велике значення свідчить про перевантаження функціональністю.

– CBO (Coupling Between Object Classes) показує ступінь взаємозалежності між класами: чим вищий цей показник, тим складніше підтримувати та модифікувати код.

− WMC (Weighted Methods per Class) оцінює сумарну складність методів у класі, що дозволяє зробити висновки щодо можливості їх подальшого супроводу та надійності.

− NOC (Number of Children) та DIT (Depth of Inheritance Tree), характеризують ієрархічну структуру класів, їх здатність до розширення та глибину наслідування.

У сукупності ці метрики забезпечують цілісне уявлення про якість програмного забезпечення, допомагаючи виявити потенційні ризики, надмірну складність і напрямки для вдосконалення коду.

Однією з перших і найвідоміших моделей оцінювання якості програмного забезпечення є модель Джима МакКолла (General Electric Model, 1977), створена для ВПС США [8]. Модель спрямована на подолання розриву між користувачами та розробниками, поєднуючи їхні потреби та пріоритети й має три ключові аспекти визначення якості програмного продукту: продукт перегляду (здатність зазнавати змін), продукт з перехідною економікою (пристосовність до нових умов), і продукт діяльності (його експлуатаційних характеристик).

Другим важливим етапом у розвитку моделей оцінювання якості стала модель Баррі Боема (1978), яка вдосконалила підхід МакКолла, усунувши його недоліки. Вона також має ієрархічну структуру, де якість визначається через сукупність характеристик різних рівнів. На верхньому рівні модель охоплює три ключові аспекти – зручність використання, ремонтопридатність і переносимість програмного забезпечення.

На проміжному рівні Боєм визначив сім основних характеристик: переносимість, надійність, ефективність, зручність у використанні, тестованість, зрозумілість і гнучкість, які комплексно формують загальну оцінку якості програмного забезпечення.

Головна відмінність цієї моделі від підходу МакКолла полягає в тому, що Боєм зосередився не лише на точних вимірюваннях окремих характеристик, а й на їх взаємозв’язках, приділяючи особливу увагу обслуговуванню та підтримці програмного забезпечення.

У 1987 році компанія Hewlett-Packard представила модель якості FURPS (Functionality, Usability, Reliability, Performance, Supportability) – промислову інтерпретацію підходів МакКолла та Боема [9]. Вона включає п’ять основних

атрибутів: функціональність, зручність використання, надійність, продуктивність і можливість супроводу, які деталізуються 27 підпоказниками нижчого рівня.

Сучасна версія FURPS+ розширена додатковими вимогами: проєктними, реалізаційними, інтерфейсними та фізичними, що забезпечує створення повної аналітичної документації та якісно задокументованого проєкту.

Попри велику кількість різноманітних показників, модель має низку обмежень, пов’язаних з відсутністю формалізованих метрик, суб’єктивністю інтерпретацій вимог та недостатньою придатністю для складних застосунків, що обмежує її застосування як універсального інструмента, зокрема для оцінювання якості застосунків з відкритим кодом на Java, що розробляються з використанням фреймворку Hibernate.

Для оцінювання якості застосовують також лінійні регресійні моделі, що базуються на простих математичних зв’язках між метриками, але часто вони не здатні врахувати нелінійні залежності між метриками RFC, CBO та WMC і виявляються недостатньо точними для складних застосунків з відкритим кодом на Java. У сукупності ці показники формують основу для оцінювання якості коду, проте їх взаємодія має нелінійний характер, який неможливо коректно відобразити за допомогою звичайних лінійних підходів.

З метою підвищення точності оцінювання якості застосунків з відкритим кодом на Java, в [5] було запропоновано використовувати нелінійні регресійні моделі, які враховують взаємозалежність між трьома основними метриками коду. Моделі представлені трьома рівняннями:

– RFC = f(CBO, WMC);
– CBO = f(WMC, RFC);
– WMC = f(RFC, CBO).

Завдяки цьому досягається комплексний опис структури програмного продукту, де кожна метрика виступає одночасно і залежною, і незалежною змінною, що дозволяє точніше виявляти вплив окремих характеристик на загальну якість застосунків.

Метод побудови моделей складається з шести етапів:

1. Перевірка нормальності розподілу даних тестом Мардіа.

2. Нормалізація даних – застосовується одновимірне нормалізуюче перетворення методом Бокса-Кокса для кожного параметру RFC, CBO та WMC, що забезпечує порівнюваність метрик.

3. Розрахунок відстані Махаланобіса для кожного застосунку, що дозволяє виявити аномальні спостереження.

4. Перевірка статистики розрахованої відстані Махаланобіса: якщо значення перевищує критичний поріг, застосунок виключається з подальшого аналізу.

5. Побудова довірчих та прогнозних інтервалів для кожної з трьох нелінійних регресій.

6. Класифікація рівня якості застосунку відносно побудованих інтервалів [5]: якщо фактичні параметри застосунку розташовані між межами довірчого та прогнозного інтервалів – якість висока, якщо дані знаходяться всередині меж довірчого інтервалу – якість середня, а якщо значення виходять за верхню межу прогнозного інтервалу – якість низька.

Запропонований метод дозволяє більш глибоко аналізувати взаємозв’язки між метриками та надає інструмент для комплексного оцінювання їхньої архітектурної якості. Використання нелінійної регресії у поєднанні з довірчими та прогнозними інтервалами забезпечує більш точне моделювання реальних залежностей між характеристиками програмного коду, підвищуючи достовірність і практичну цінність отриманих результатів.

1.2 Особливості розробки застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Розробка застосунків з відкритим кодом на Java із використанням фреймворку Hibernate має свої особливості, які впливають на зручність, якість

та подальшу підтримку проєкту. Hibernate вважається одним із найпопулярніших інструментів для роботи з базами даних у Java, оскільки дозволяє взаємодіяти з ними через об’єкти, а не через складні SQL-запити. Це спрощує процес розроблення і робить код більш зрозумілим навіть для нових учасників команди. Крім того, Hibernate сумісний із більшістю відомих СКБД, таких як MySQL, PostgreSQL та Oracle, що робить його універсальним рішенням для проєктів різного масштабу.

У більшості випадків під час створення застосунків використовується багаторівнева архітектура, де розділяються рівні даних, логіки та інтерфейсу користувача. Такий підхід дозволяє зробити застосунок більш гнучким та полегшує супровід і спільну роботу кількох розробників над одним проєктом.

Досить часто Hibernate застосовують разом із фреймворком Spring. Це поєднання дає змогу створювати сучасні Java-застосунки з повним набором інструментів для конфігурації, роботи з даними та безпеки. Наприклад, Spring Boot спрощує налаштування застосунку, Spring Data полегшує роботу з репозиторіями, а Spring Security відповідає за авторизацію та захист даних.

Під час командної розробки важливо стежити за змінами у структурі бази даних. Для цього зазвичай використовують такі інструменти, як Flyway або Liquibase – вони допомагають уникнути конфліктів і забезпечують узгодженість між різними середовищами.

Щоб підвищити швидкодію, у Hibernate застосовують кешування другого рівня. Завдяки цьому кількість звернень до бази даних зменшується, і програма працює швидше. Для кешу найчастіше використовують такі рішення, як Ehcache, Infinispan або Redis. Також важливо правильно налаштовувати способи завантаження даних (Eager або Lazy), адже це безпосередньо впливає на швидкість роботи застосунку.

Якість коду у відкритих Java-проєктах має велике значення, адже ним користуються і його змінюють багато людей. Щоб підтримувати єдині стандарти, використовують інструменти на кшталт SonarQube або PMD, а для тестування – JUnit і Mockito. Так можна швидко виявити помилки ще до того, як вони потраплять у готову збірку. Для складніших перевірок часто

застосовують Testcontainers – це дає можливість протестувати роботу застосунку в умовах, максимально близьких до реальних.

Оскільки відкриті проєкти зазвичай мають декілька учасників, велике значення має зручна організація командної роботи. Для цього використовують GitHub або GitLab, які поєднують у собі контроль версій, управління завданнями та автоматизацію тестування. Збірка проєкту зазвичай здійснюється за допомогою Maven або Gradle, а процес розгортання – через Jenkins чи GitHub Actions.

Безпека теж не залишається поза увагою. Hibernate використовує параметризовані запити, що знижують ризик SQL-ін’єкцій [10]. Якщо ж поєднати його зі Spring Security, можна створити повноцінне підґрунтя для ефективного захисту даних і контролю доступу користувачів.

Отже, розробка відкритих Java-застосунків із використанням фреймворку Hibernate є складним та багатоетапним процесом, що передбачає дотримання сучасних вимог до якості, надійності та зручності використання застосунків з можливістю легко масштабувати та удосконалювати їх у майбутньому.

1.3 Обґрунтування вибору нелінійних регресійних моделей для оцінювання RFC, CBO та WMC застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Для оцінювання якості програмних застосунків було обрано вже наявний, модифікований метод, що ґрунтується на побудові трьох нелінійних регресійних моделей для метрик RFC, CBO та WMC на рівні застосунків та формування на їх основі довірчих та прогнозованих інтервалів. Модифікація цього методу зумовлена необхідністю розглядати кожну з зазначених метрик як окрему незалежну змінну в межах відповідних регресійних моделей: RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO). Для їх побудови

можна використовувати вже зазначене раніше одновимірне нормалізуюче перетворення Бокса-Кокса.

Використання ж лінійної регресійної моделі в даній роботі не є доцільною, через розповсюджену недостовірність результатів для специфічних метрик RFC, CBO та WMC. Так відбувається через декілька причин:

Нелінійна природа взаємозв'язків між метриками. У проєктах на Java, особливо при використанні Hibernate, складність класу (WMC) та ступінь зв'язності між ними (CBO) мають нелінійний вплив на значення RFC. Наприклад, зростання кількості зв'язків між класами не завжди пропорційно збільшує кількість відповідей на повідомлення (RFC) – іноді цей вплив має експоненційний або логарифмічний характер.

Метрики мають широкий діапазон значень. Для Java-застосунків, що використовують Hibernate, спостерігається значна варіація значень метрик:

– RFC: [1,6000; 13,6222];
– CBO: [2,5714; 9,0952];
– WMC: [2,1905; 16,2500].

Такий розкид даних ускладнює побудову стабільної лінійної моделі, оскільки високі значення окремих показників призводять до зміщення регресійних коефіцієнтів.

Для вирішення цих проблем буде застосована нормалізація на основі одновимірних перетворень Бокса-Кокса, яка дозволить стабілізувати дисперсію, зменшити асиметрію та ексцес розподілу та зробити залежності між змінними більш лінійними у нових координатах. Це також забезпечить кращу кореляцію між метриками після трансформації, що має позитивно вплинути на якість регресійних моделей.

З метою підвищення точності оцінювання буде розроблено три окремі нелінійні регресійні моделі, які відображатимуть взаємозалежність між метриками:

RFC(CBO, WMC) – модель, що описує, як на кількість відповідей класу впливають зв'язність із іншими класами та кількість методів.

CBO(WMC, RFC) – модель, що відображає ступінь взаємозалежності класу з іншими на основі його функціонального навантаження.

WMC(RFC, CBO) – модель, яка дозволяє оцінити складність класу на основі його активності та зв’язності.

Існуюча модель для оцінювання якості програмного забезпечення з відкритим кодом на Java [6] не враховує всіх особливостей застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, оскільки має такі значення діапазонів метрик:

– RFC: [4,7940; 37,2778];
– CBO: [3,4550; 22,4167];
– WMC: [4,9770; 71,8368].

Таким чином, відмінність значень діапазонів метрик підкреслює необхідність розробки вдосконалених нелінійних регресійних моделей для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate задля підвищення достовірності оцінювання.

2 РОЗРОБКА НЕЛІНІЙНИХ РЕГРЕСІЙНИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ HIBERNATE

2.1 Збір та попередня обробка даних метрик застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Для побудови трьох нелінійних регресійних моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO), що оцінюватимуть відповідні метрики RFC, CBO та WMC для застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, було зібрано дані з 70 проєктів аналогічної тематики, доступних на GitHub. Необхідні для побудови моделей метрики RFC, CBO та WMC були зібрані завдяки програмному інструменту з відкритим кодом CK [11]. Для кожного проєкту були розраховані середні арифметичні значення кожної з метрик відносно загальної кількості класів, базуючись на яких будуються три нелінійні регресійні моделі, де спочатку метрика RFC обирається як випадкова залежна величина Y, а CBO та WMC є незалежними змінними X_1 та X_2 відповідно, а далі незалежні змінні змінюються відповідно до обраної залежної. Тобто: WMC – Y, RFC – X_1, CBO – X_2 та CBO – Y, WMC – X_1, RFC – X_2.

Набір застосунків та зібраних з них метрик RFC, CBO та WMC наведено в таблиці 2.1.

Таблиця 2.1 – Набір застосунків та метрик RFC, CBO та WMC

№	Застосунок	RFC	CBO	WMC
1	2	3	4	5
1	adixchen/demo-restWS-spring-jersey-jpa2-hibernate	5,4000	6,5000	6,2000
2	ajanata/PretendYoureXyzzy	9,6290	6,5403	8,6210

Продовження таблиці 2.1

1	2	3	4	5
3	akshu20791/spring-hibernate-security-react-ant-design-polling-app	6,1087	6,2391	5,9783
4	anakiou/multitenancy	3,0833	4,5833	3,0833
5	bcssp10/multi-tenant	4,4000	5,4000	2,5000
6	bdot/struts-hibernate	2,8571	2,5714	5,5714
7	benssj5/Projet-JEE-Gestion-des-Comptes-Spring-JPA-Hibernate	2,1250	5,5625	4,6250
8	BhartiKhankure/Vehicle_insurance_management_system	8,0000	6,5909	14,2727
9	brishi19791/HealthCare-WebApplication	5,0882	7,4118	7,1765
10	callistaenterprise/blog-multitenancy	5,6842	6,1053	2,6842
11	cassiomolin/jersey-jwt	7,6000	5,6000	5,2571
12	Cepr0/sb-multitenant-db-demo	6,1333	4,1333	4,2000
13	chrisssss/springboot-hibernatespatial-postgis-demo	6,6667	6,0000	13,0000
14	codedecode25/HibernateDemo	7,3333	6,0000	4,8333
15	coderzc/biubiu	7,5424	4,3390	10,5932
16	CoffeeLatte007/sharebook-jersey-jwt-spring-hibernate	5,7692	5,1923	6,3846
17	colinbut/sales-order-system	3,0476	9,0952	2,1905
18	cr2121/spring	1,6000	3,0000	5,6000
19	cyela/Spring-Web-App	8,5333	5,8667	14,4000
20	dadoonet/hsearch-es-demo	1,7500	4,2500	4,7500
21	daliasheasha/HibernateApp	7,0000	6,6667	11,6667
22	desperatecat/ShoppingPortal_Vue_Springboot_Project	6,3208	7,5849	8,2830
23	devdilip/todo-springboot-mysql-hibernate-rest	2,8000	4,4000	5,2000
24	diegoneuralo/maven	3,5000	8,1667	2,8333
25	discospiff/JavaFullStackEnterpriseWeb	6,4815	5,3704	6,4444
26	dmitrykhramov/Online-Bank-Simulator	4,8378	6,5135	7,0811
27	dobermai/Hibernate-Flyway-Integration	2,6667	6,0000	3,0000
28	F4bwDP6a6W/spring-mvc-REST	4,7500	3,8750	9,7500
29	foysal-mahmud/E-BookShop----Spring-boot	5,7931	6,4310	7,8621
30	frenmanoj/bookstore	3,1667	4,6667	6,6667
31	Fruzenshtein/security-spr	3,0556	5,3333	2,7778
32	IcedSoul/Shopping	6,1842	4,5526	8,6842
33	jamesward/spring_hibernate_hstore_demo	6,4000	5,5000	3,7000
34	jasenkoh/spring-boot-jersey-hibernate	2,6667	5,5833	4,4167
35	Java-Techie-jt/Payment-Resource	3,1667	5,1667	5,0000
36	jjoe64/shiroexample	8,0000	5,9091	6,4545
37	Joryun/Shop	5,4648	5,3521	7,2254
38	Karszt/sda-zdjava124	10,0930	4,5814	4,2326

Продовження таблиці 2.1

1	2	3	4	5
39	khozema-nullwala/online-shopping	7,8788	6,9091	10,8182
40	lastfm/musicbrainz-data	4,2368	6,4474	5,6842
41	licyun/SpringMVCgbook	7,1667	4,7500	9,5417
42	michaelisvy/hibernate-4-spring-3.1-samples	4,8000	7,7000	4,2000
43	MiteshPuthran/Shopping-Cart-Spring-MVC-Hibernate-Application	7,4444	7,9259	7,4815
44	mohamedYoussfi/eboutique-jpa-hibernate-spring	4,8333	7,0556	10,0000
45	MoonSulong/Ecommerce	4,8649	5,9459	5,8108
46	OKaluzny/spring-boot-rest-api-postgresql	4,2500	8,0000	2,2500
47	OKaluzny/springboot-rest-api-angularjs-https	5,5000	6,2000	5,0000
48	oregami/dropwizard-guice-jpa-seed	6,1163	6,2791	6,2326
49	pfac/demo-spring-data-jpa-hibernate-h2	4,6667	4,0000	6,5000
50	quarkiverse/quarkus-hibernate-types	2,9444	5,6111	5,1111
51	radoslaw-piwowarczyk/PetClinic-Spring-Thymeleaf-Hibernate	3,3333	4,8974	2,6923
52	rajbhatta/springboot_restapi_hibernate	2,2143	4,0000	4,0000
53	ralscha/eds-starter6-jpa	13,6222	7,5111	8,8000
54	RameshMF/springboot-thymeleaf-crud-pagination-sorting-webapp	3,8333	5,1667	4,6667
55	rampatra/springboot-hibernate	9,2857	8,0000	9,5714
56	rid17pawar/HospitalManagement	7,0189	8,0189	6,0943
57	royvanrijn/graalvm-native-microservice	6,0769	3,6923	5,5385
58	rutujaBacchuwar/MuzixAppHibernate	4,8571	5,8571	5,0000
59	scbushan05/book-api	3,6667	5,3333	4,0000
60	scottescue/dropwizard-entitymanager	12,6471	6,9412	11,0588
61	soulcode-acad/goservice	5,0488	5,9756	7,0244
62	sroysf/webapp-springmvc-jpa-hibernate	4,2857	5,8571	4,1429
63	stevegocoding/SpringMVC-SpringDataJpa-Hibernate-Demo	2,0000	3,4000	4,6000
64	swapnildipankar/java_jersey_spring_hibernate_maven	8,2000	5,9500	11,9500
65	ThxGodNovember/AprilDragonSpring	13,3056	5,1111	16,2500
66	tolitius/money-making-project	2,5000	2,6667	3,6667
67	vasylmalik/project-hibernate-1	6,6923	4,3846	7,3846
68	yannbriancon/spring-hibernate-query-utils	5,8750	2,6250	6,3750
69	ykameshrao/spring-mvc-angular-js-hibernate-bootstrap-java-single-page-jwt-auth-rest-api-webapp-framework	5,3235	5,3088	5,1029
70	ZhuXS/Spring-Shiro-Spark	7,4000	6,1400	9,0400

Щоб перевірити нормальність багатовимірних даних в даній роботі використовується тест Мардіа [12], що ґрунтується на визначенні параметрів багатовимірної асиметрії та багатовимірного ексцесу. Оцінка асиметрії виконується шляхом розрахунку відстаней Махаланобіса для кожного з проєктів [13]:

$$d_i^2 = \left(\mathbf{X}_i - \overline{\mathbf{X}}\right)^T \mathbf{S}_N^{-1} \left(\mathbf{X}_i - \overline{\mathbf{X}}\right), \tag{2.1}$$

де $\mathbf{X}$ – вектор випадкових величин, $\mathbf{X} = \{Y, X_1, X_2\}^T$;

$\overline{\mathbf{X}}$ – це вектор вибіркових середніх, $\overline{\mathbf{X}} = \{\overline{Y}, \overline{X}_1, \overline{X}_2\}^T$;

N – кількість точок даних;

$\mathbf{S}_N^{-1}$ – обернена коваріаційна матриця, $\mathbf{S}_N^{-1} = \frac{1}{\mathbf{S}_N}$;

$\mathbf{S}_N$ – зміщена вибіркова коваріаційна матриця:

$$\mathbf{S}_N = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{X}_i - \overline{\mathbf{X}}\right)\left(\mathbf{X}_i - \overline{\mathbf{X}}\right)^T. \tag{2.2}$$

Формули для обчислення багатовимірної асиметрії та ексцесу за методом Мардіа виглядають наступним чином [14]:

$$\beta_1 = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\left(\mathbf{X}_i - \overline{\mathbf{X}}\right)^T \mathbf{S}_N^{-1} \left(\mathbf{X}_i - \overline{\mathbf{X}}\right)\right]^3, \tag{2.3}$$

$$\beta_2 = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\mathbf{X}_i - \overline{\mathbf{X}}\right)^T \mathbf{S}_N^{-1} \left(\mathbf{X}_i - \overline{\mathbf{X}}\right)\right]^2, \tag{2.4}$$

де β_1 – багатовимірна асиметрія; β_2 – багатовимірний ексцес.

Щоб перевірити багатовимірну асиметрію β_1 (2.3) необхідно обчислити тестову статистику [14]:

$$\frac{N}{6}\beta_1, \tag{2.5}$$

яка наближається до розподілу χ^2 зі ступенями свободи $k(k+1)(k+2)/6$ та рівнем значущості $\alpha = 0{,}005$.

Для багатовимірного ексцесу β_2 (2.4) застосовується тестова статистика z_2, що відповідає $1-\alpha$-квантилю нормального розподілу із середнім значенням $k(k+2)$ і дисперсією $8k(k+2)/N$.

Перевірка нормальності виконується завдяки порівнянню значення $\frac{N}{6}\beta_1$ із квантилем $\chi^2_{k(k+1)(k+2)/6,\alpha}$ та значення β_2 із статистикою z_2.

Розподіл вважається не нормальним у тому випадку, коли хоча б одна з тестових статистик перевищує відповідний критичний рівень.

Внаслідок розрахунків були отримані наступні значення:

$\beta_1 = 2{,}8425$; $\frac{N}{6}\beta_1 = 33{,}1619$; $\chi^2_{k(k+1)(k+2)/6,\alpha} = 25{,}1882$;

$\beta_2 = 17{,}4883$; $z_2 = 18{,}3726$.

Відповідно до наведених вище результатів, було виявлено перевищення критичного рівня за асиметрією: 33,1619 > 25,1882, що свідчить про відсутність нормального розподілу даних, навіть при нижчому за критичний рівень ексцесі: 17,4883 < 18,3726. Так як отримане значення статистики за асиметрією не є прийнятним для побудови нелінійних моделей, слід спочатку провести нормалізацію даних [15].

Перетворення Бокса-Кокса можна записати наступним чином:

$$Z_j = \boldsymbol{\Psi}(\lambda_j) = \begin{cases} \left(X_j^{\lambda_j} - 1\right)/\lambda_j, & if \quad \lambda_j \neq 0; \\ \ln(X_j), & if \quad \lambda_j = 0. \end{cases} \tag{2.6}$$

де Z_j – це випадкова змінна з розподілом Гаусу (нормалізована змінна);

$\boldsymbol{\Psi}$ – вектор, $\boldsymbol{\Psi} = \{\Psi_Y, \Psi_{X_1}, \Psi_{X_2}\}^T$ перетворення $\mathbf{T} = \boldsymbol{\Psi}(\mathbf{P})$, $\mathbf{T} = \{\mathrm{Z}_Y, \mathrm{Z}_{X_1}, \mathrm{Z}_{X_2}\}^T$ – гаусівський випадковий вектор, $\boldsymbol{P} = \{\mathrm{Y}, \mathrm{X}_1, \mathrm{X}_2\}^T$ – негаусівський випадковий вектор;

λ_j – параметр перетворення Бокса-Кокса;

X_j – компонента вектору випадкових величин $\mathbf{X}$, $\mathbf{X} = \{Y, X_1, X_2\}^T$;

$j = Y, X_1, X_2$.

Розраховані значення лямбд для метрик RFC, CBO та WMC дорівнюють 0,2485, 1,0492 та 0,0370 відповідно.

Нормалізовані методом Бокса-Кокса значення метрик RFC, CBO та WMC наведено в таблиці 2.2.

Таблиця 2.2 – Набір нормалізованих метрик RFC, CBO та WMC

№	Застосунок	Z_{RFC}	Z_{CBO}	Z_{WMC}
1	2	3	4	5
1	adixchen/demo-restWS-spring-jersey-jpa2-hibernate	2,0947	5,8396	1,8875
2	ajanata/PretendYoureXyzzy	3,0404	5,8838	2,2423
3	akshu20791/spring-hibernate-security-react-ant-design-polling-app	2,2851	5,5538	1,8486
4	anakiou/multitenancy	1,2993	3,7550	1,1498
5	bcssp10/multi-tenant	1,7911	4,6388	0,9320
6	bdot/struts-hibernate	1,1994	1,6143	1,7733
7	benssj5/Projet-JEE-Gestion-des-Comptes-Spring-JPA-Hibernate	0,8290	4,8155	1,5757
8	BhartiKhankure/Vehicle_insurance_management_system	2,7225	5,9393	2,7933
9	brishi19791/HealthCare-WebApplication	2,0049	6,8426	2,0444
10	callistaenterprise/blog-multitenancy	2,1732	5,4075	1,0056
11	cassiomolin/jersey-jwt	2,6370	4,8563	1,7115
12	Cepr0/sb-multitenant-db-demo	2,2914	3,2712	1,4738
13	chrisssssss/springboot-hibernatespatial-postgis-demo	2,4236	5,2924	2,6905
14	codedecode25/HibernateDemo	2,5781	5,2924	1,6223
15	coderzc/biubiu	2,6244	3,4920	2,4662
16	CoffeeLatte007/sharebook-jersey-jwt-spring-hibernate	2,1961	4,4134	1,9189
17	colinbut/sales-order-system	1,2839	8,7100	0,7956
18	cr2121/spring	0,4985	2,0650	1,7788
19	cyela/Spring-Web-App	2,8315	5,1469	2,8031
20	dadoonet/hsearch-es-demo	0,6004	3,3964	1,6039
21	daliasheasha/HibernateApp	2,5023	6,0225	2,5717
22	desperatecat/ShoppingPortal_Vue_Springboot_Project	2,3388	7,0337	2,1990
23	devdilip/todo-springboot-mysql-hibernate-rest	1,1733	3,5576	1,6999
24	diegoneuralo/maven	1,4696	7,6777	1,0617
25	discospiff/JavaFullStackEnterpriseWeb	2,3786	4,6067	1,9289
26	dmitrykhramov/Online-Bank-Simulator	1,9298	5,8544	2,0300
27	dobermai/Hibernate-Flyway-Integration	1,1107	5,2924	1,1212
28	F4bwDP6a6W/spring-mvc-REST	1,9028	2,9947	2,3759
29	foysal-mahmud/E-BookShop----Spring-boot	2,2025	5,7639	2,1427
30	frenmanoj/bookstore	1,3347	3,8449	1,9652

Продовження таблиці 2.2

1	2	3	4	5
31	Fruzenshtein/security-spr	1,2874	4,5664	1,0412
32	IcedSoul/Shopping	2,3044	3,7219	2,2502
33	jamesward/spring_hibernate_hstore_demo	2,3585	4,7475	1,3405
34	jasenkoh/spring-boot-jersey-hibernate	1,1107	4,8381	1,5269
35	Java-Techie-jt/Payment-Resource	1,3347	4,3856	1,6583
36	jjoe64/shiroexample	2,7225	5,1932	1,9305
37	Joryun/Shop	2,1129	4,5868	2,0517
38	Karszt/sda-zdjava124	3,1235	3,7529	1,4820
39	khozema-nullwala/online-shopping	2,6969	6,2888	2,4892
40	lastfm/musicbrainz-data	1,7367	5,7819	1,7947
41	licyun/SpringMVCgbook	2,5405	3,9348	2,3524
42	michaelisvy/hibernate-4-spring-3.1-samples	1,9182	7,1609	1,4738
43	MiteshPuthran/Shopping-Cart-Spring-MVC-Hibernate-Application	2,6029	7,4109	2,0892
44	mohamedYoussfi/eboutique-jpa-hibernate-spring	1,9284	6,4500	2,4034
45	MoonSulong/Ecommerce	1,9381	5,2333	1,8182
46	OKaluzny/spring-boot-rest-api-postgresql	1,7412	7,4929	0,8232
47	OKaluzny/springboot-rest-api-angularjs-https	2,1227	5,5110	1,6583
48	oregami/dropwizard-guice-jpa-seed	2,2870	5,5976	1,8931
49	pfac/demo-spring-data-jpa-hibernate-h2	1,8768	3,1284	1,9381
50	quarkiverse/quarkus-hibernate-types	1,2387	4,8684	1,6816
51	radoslaw-piwowarczyk/PetClinic-Spring-Thymeleaf-Hibernate	1,4034	4,0940	1,0087
52	rajbhatta/springboot_restapi_hibernate	0,8789	3,1284	1,4224
53	ralscha/eds-starter6-jpa	3,6764	6,9522	2,2646
54	RameshMF/springboot-thymeleaf-crud-pagination-sorting-webapp	1,5952	4,3856	1,5852
55	rampatra/springboot-hibernate	2,9770	7,4929	2,3558
56	rid17pawar/HospitalManagement	2,5066	7,5139	1,8691
57	royvanrijn/graalvm-native-microservice	2,2769	2,7996	1,7670
58	rutujaBacchuwar/MuzixAppHibernate	1,9357	5,1364	1,6583
59	scbushan05/book-api	1,5335	4,5664	1,4224
60	scottescue/dropwizard-entitymanager	3,5356	6,3241	2,5132
61	soulcode-acad/goservice	1,9933	5,2658	2,0213
62	sroysf/webapp-springmvc-jpa-hibernate	1,7532	5,1364	1,4594
63	stevegocoding/SpringMVC-SpringDataJpa-Hibernate-Demo	0,7564	2,4885	1,5699
64	swapnildipankar/java_jersey_spring_hibernate_maven	2,7640	5,2378	2,5980
65	ThxGodNovember/AprilDragonSpring	3,6316	4,3254	2,9368
66	tolitius/money-making-project	1,0290	1,7142	1,3310
67	vasylmalik/project-hibernate-1	2,4298	3,5410	2,0751

Продовження таблиці 2.2

1	2	3	4	5
68	yannbriancon/spring-hibernate-query-utils	2,2242	1,6704	1,9173
69	ykameshrao/spring-mvc-angular-js-hibernate-bootstrap-java-single-page-jwt-auth-rest-api-webapp-framework	2,0730	4,5398	1,6799
70	ZhuXS/Spring-Shiro-Spark	2,5930	5,4454	2,2937

Після одновимірної нормалізації для кожного параметру методом Бокса-Кокса, було отримано наступні результати:

$\beta_1 = 1{,}2705; \frac{N}{6}\beta_1 = 14{,}8224; \chi^2_{k(k+1)(k+2)/6,\alpha} = 25{,}1882;$

$\beta_2 = 14{,}9362; z_2 = 18{,}3726.$

Відповідно до розрахованих значень асиметрії та ексцесу, внаслідок нормалізації дані мають нормальний розподіл та проходять тест Мардіа.

Далі необхідно здійснити пошук та видалення викидів із вхідних даних, за їх наявності. Для цього слід виконати наступні кроки:

1. Розрахувати коваріаційну матрицю $\mathbf{S}_N$ (2.2) для трьох змінних, використовуючи нормалізовані дані.

2. Знайти зворотну коваріаційну матрицю $\mathbf{S}_N^{-1}$.

3. Обчислити квадрат відстані Махаланобіса для нормалізованих даних для кожного спостереження за формулою (2.1).

4. Для кожного спостереження обчислити тестову статистику (T_S) та завдяки їй оцінити відхилення від середнього значення з урахуванням коваріаційної структури даних [16]:

$$T_{S_i} = \frac{(N-k)Nd_i^2}{(N^2-1)k}. \quad (2.7)$$

5. Обчислити квантиль розподілу Фішера $F_{\text{крит}}$ як [17]:

$$F_{\text{крит}} = F(\alpha, 3, N-3), \quad (2.8)$$

де α – рівень значущості (0,005), 3 – кількість ступенів вільності, а N – розмір вибірки.

6. У випадку, якщо $T_{S_i}(2.7) > F_{\text{крит}}(2.8)$, то i-те спостереження вважається викидом та видаляється з вибірки.

7. Кроки 1-7 повторюються до тих пір, поки всі викиди не будуть видалені.

В таблиці 2.3 наведено квадрати відстаней Махаланобіса D^2 та тестову статистику T_S розраховані на нормалізованих даних для кожного застосунку.

Таблиця 2.3 – Набір розрахованих значень D^2 та T_S

№	Застосунок	D^2	T_S
1	2	3	4
1	adixchen/demo-restWS-spring-jersey-jpa2-hibernate	0,4132	0,1319
2	ajanata/PretendYoureXyzzy	2,0872	0,6661
3	akshu20791/spring-hibernate-security-react-ant-design-polling-app	0,2595	0,0828
4	anakiou/multitenancy	2,4220	0,7729
5	bcssp10/multi-tenant	4,6694	1,4901
6	bdot/struts-hibernate	5,1275	1,6362
7	benssj5/Projet-JEE-Gestion-des-Comptes-Spring-JPA-Hibernate	3,9544	1,2619
8	BhartiKhankure/Vehicle_insurance_management_system	4,3289	1,3814
9	brishi19791/HealthCare-WebApplication	2,6347	0,8408
10	callistaenterprise/blog-multitenancy	5,4184	1,7291
11	cassiomolin/jersey-jwt	2,1383	0,6824
12	Cepr0/sb-multitenant-db-demo	4,2724	1,3634
13	chrisssssss/springboot-hibernatespatial-postgis-demo	3,5894	1,1454
14	codedecode25/HibernateDemo	2,0722	0,6613
15	coderzc/biubiu	2,9531	0,9424
16	CoffeeLatte007/sharebook-jersey-jwt-spring-hibernate	0,2665	0,0850
17	colinbut/sales-order-system	11,9993	3,8291
18	cr2121/spring	8,0505	2,5690
19	cyela/Spring-Web-App	3,8210	1,2193
20	dadoonet/hsearch-es-demo	5,2143	1,6640
21	daliasheasha/HibernateApp	2,9967	0,9563
22	desperatecat/ShoppingPortal_Vue_Springboot_Project	2,7237	0,8692
23	devdilip/todo-springboot-mysql-hibernate-rest	2,0274	0,6470
24	diegoneuralo/maven	6,4701	2,0647
25	discospiff/JavaFullStackEnterpriseWeb	0,4809	0,1535
26	dmitrykhramov/Online-Bank-Simulator	1,1347	0,3621
27	dobermai/Hibernate-Flyway-Integration	2,7284	0,8707
28	F4bwDP6a6W/spring-mvc-REST	3,3609	1,0725
29	foysal-mahmud/E-BookShop----Spring-boot	0,8208	0,2619

Продовження таблиці 2.2

1	2	3	4
30	frenmanoj/bookstore	2,2626	0,7220
31	Fruzenshtein/security-spr	2,6170	0,8351
32	IcedSoul/Shopping	1,4550	0,4643
33	jamesward/spring_hibernate_hstore_demo	3,4089	1,0878
34	jasenkoh/spring-boot-jersey-hibernate	2,0326	0,6486
35	Java-Techie-jt/Payment-Resource	1,1016	0,3515
36	jjoe64/shiroexample	1,3584	0,4335
37	Joryun/Shop	0,2619	0,0836
38	Karszt/sda-zdjava124	10,7937	3,4444
39	khozema-nullwala/online-shopping	2,4342	0,7768
40	lastfm/musicbrainz-data	0,9122	0,2911
41	licyun/SpringMVCgbook	1,7862	0,5700
42	michaelisvy/hibernate-4-spring-3.1-samples	2,9108	0,9289
43	MiteshPuthran/Shopping-Cart-Spring-MVC-Hibernate-Application	2,8421	0,9069
44	mohamedYoussfi/eboutique-jpa-hibernate-spring	4,7762	1,5242
45	MoonSulong/Ecommerce	0,1010	0,0322
46	OKaluzny/spring-boot-rest-api-postgresql	7,6590	2,4441
47	OKaluzny/springboot-rest-api-angularjs-https	0,3944	0,1259
48	oregami/dropwizard-guice-jpa-seed	0,2472	0,0789
49	pfac/demo-spring-data-jpa-hibernate-h2	1,4941	0,4768
50	quarkiverse/quarkus-hibernate-types	1,7803	0,5681
51	radoslaw-piwowarczyk/PetClinic-Spring-Thymeleaf-Hibernate	3,1774	1,0139
52	rajbhatta/springboot_restapi_hibernate	3,1412	1,0024
53	ralscha/eds-starter6-jpa	6,2172	1,9840
54	RameshMF/springboot-thymeleaf-crud-pagination-sorting-webapp	0,4501	0,1436
55	rampatra/springboot-hibernate	3,8141	1,2171
56	rid17pawar/HospitalManagement	2,9197	0,9317
57	royvanrijn/graalvm-native-microservice	3,4958	1,1155
58	rutujaBacchuwar/MuzixAppHibernate	0,1592	0,0508
59	scbushan05/book-api	0,7754	0,2474
60	scottescue/dropwizard-entitymanager	4,6320	1,4781
61	soulcode-acad/goservice	0,4141	0,1322
62	sroysf/webapp-springmvc-jpa-hibernate	0,6192	0,1976
63	stevegocoding/SpringMVC-SpringDataJpa-Hibernate-Demo	4,6776	1,4927
64	swapnildipankar/java_jersey_spring_hibernate_maven	2,3791	0,7592
65	ThxGodNovember/AprilDragonSpring	7,4511	2,3778
66	tolitius/money-making-project	5,3563	1,7093
67	vasylmalik/project-hibernate-1	1,7598	0,5616
68	yannbriancon/spring-hibernate-query-utils	6,1547	1,9641
69	ykameshrao/spring-mvc-angular-js-hibernate-bootstrap-java-single-page-jwt-auth-rest-api-webapp-framework	0,3746	0,1195
70	ZhuXS/Spring-Shiro-Spark	0,9660	0,3083

Окрім квадратів відстаней Махаланобіса D^2 та тестової статистики T_S було розраховано квантиль розподілу Фішера $F_{\text{крит}}$ за формулою (2.8), яке дорівнює 4,6793. Також розрахований квадрат відстані Махаланобіса не має перевищувати критичне значення $\chi^2_{0,005,10} = 12{,}8382$. Як видно з таблиці 2.3, жодне із значень D_i^2 та T_{S_i} не перевищують відповідні значення $\chi^2_{0,005,10}$ та $F_{\text{крит}}$, а отже можна стверджувати про відсутність викидів у нормалізованих вхідних даних.

2.2 Побудова моделей для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

В ході попередньої обробки даних та визначення відсутності викидів у вибірці було залишено всі 70 застосунків. Побудова нелінійних моделей буде виконана з використанням нормалізуючих перетворень (2.6).

Для реалізації цього етапу, першочергово необхідно побудувати три лінійні моделі для нормалізованих даних за загальною формулою:

$$\hat{Z}_y = b_0 + b_1 Z_{x_1} + b_2 Z_{x_2} + \varepsilon, \qquad (2.9)$$

де $\hat{Z}_y$ – нормалізована залежна випадкова величина (RFC, CBO або WMC);

b_0, b_1 та b_2 – оцінки параметрів лінійного рівняння регресії;

Z_{x_1}, Z_{x_2} – нормалізовані незалежні змінні X_1 та X_2;

ε – випадкова величина, з розподілом Гауса, з нульовим математичним сподіванням та оцінкою середньоквадратичного відхилення σ_ε^2.

В першій моделі RFC(CBO, WMC) незалежним змінним X_1 та X_2 відповідають метрики CBO та WMC.

В другій моделі WMC(RFC, CBO) незалежним змінним X_1 та X_2 відповідають метрики RFC та CBO.

В третій моделі CBO(WMC, RFC) незалежним змінним X_1 та X_2 відповідають метрики WMC та RFC.

Наступним кроком, на основі лінійних моделей (2.9) необхідно побудувати нелінійні регресійні моделі:

$$\hat{Y} = \left(\left[b_0 + b_1 \cdot \frac{X_1^{\lambda_{X_1}} - 1}{\lambda_{X_1}} + b_2 \cdot \frac{X_2^{\lambda_{X_2}} - 1}{\lambda_{X_2}} + \varepsilon \right] \cdot \lambda_Y + 1 \right)^{\frac{1}{\lambda_Y}}, \qquad (2.10)$$

де $\hat{Y}$ – оцінка умовного математичного сподівання Y;

λ_Y, λ_{X_1} та λ_{X_2} – оптимальні лямбди змінних Y, X_1 та X_2.

Після побудови нелінійних регресійних моделей (2.10) маємо все необхідне для побудови інтервалів передбачення множинної нелінійної регресії:

$$\Psi_Y^{-1} \left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y} \left\{ 1 + \frac{1}{N} + (\mathbf{Z}_X^+)^T [(\mathbf{Z}_X^+)^T \mathbf{Z}_X^+]^{-1} (\mathbf{Z}_X^+) \right\}^{\frac{1}{2}} \right), \qquad (2.11)$$

де Ψ_Y – перша компонента вектору $\boldsymbol{\Psi}$, $\boldsymbol{\Psi} = \{\Psi_Y, \Psi_{X_1}, \Psi_{X_2}\}^T$ зворотного перетворення;

$\hat{Z}_Y$ – передбачуваний результат відповідно до лінійного рівняння регресії (2.9) для нормалізованих даних;

$t_{\alpha/2,\nu}$ – квантиль розподілу Стьюдента з рівнем значущості $\alpha/2$ та $\nu = N - 3$ ступенями вільності;

$\mathbf{Z}_X^+$ – матриця нормалізованих регресорів зі значеннями $Z_{X_{1_i}} - \overline{Z}_{X_1}, Z_{X_{2_i}} - \overline{Z}_{X_2}$;

$\overline{Z}_{X_j} = \frac{1}{N} \sum_{i=1}^{N} Z_{X_i}$, $j = 1, 2$ – середні значення нормалізованих незалежних змінних;

$S_{Z_Y}^2 = \frac{1}{\nu} \sum_{i=1}^{N} \left(Z_{Y_i} - \hat{Z}_{Y_i} \right)^2$ – залишкова дисперсія, $\nu = N - k - 1$;

$(\mathbf{Z}_X^+)^T \mathbf{Z}_X^+$ – 2×2 матриця:

$$(\mathbf{Z}_X^+)^T \mathbf{Z}_X^+ = \begin{pmatrix} S_{Z_1 Z_1} & S_{Z_1 Z_2} \\ S_{Z_1 Z_2} & S_{Z_2 Z_2} \end{pmatrix},$$

де $S_{Z_q Z_r} = \sum_{i=1}^{N} [Z_{q_i} - \overline{Z}_q]\,[Z_{r_i} - \overline{Z}_r]$, $q, r = 1, 2$.

Оцінка параметрів всіх моделей здійснюється методом найменших квадратів [16]:

$$\hat{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y, \tag{2.12}$$

де $\hat{b}$ – оцінка найменших квадратів, $\mathbf{X}$ – змінна матричного регресора $\mathbf{X}$, T – транспонування матриці, а y – вектор значення змінної відгуку.

Шляхом зворотного нормалізуючого перетворення отримаємо нелінійні регресійні моделі (2.10) з такими значеннями коефіцієнтів:

RFC(CBO, WMC): $b_0 = -0{,}2599, b_1 = 0{,}1469, b_2 = 0{,}8532$.

CBO(WMC, RFC): $b_0 = 4{,}1336, b_1 = -0{,}7971, b_2 = 1{,}1138$.

WMC(RFC, CBO): $b_0 = 1{,}1486, b_1 = 0{,}4837, b_2 = -0{,}0596$.

Таким чином, отримаємо кінцевий вигляд нелінійних регресійних моделей:

$$\begin{aligned}
\hat{Y}_{RFC} &= \left(\left[-0{,}2599 + 0{,}1469 \cdot \frac{X_{CBO}^{1{,}0492} - 1}{1{,}0492} + 0{,}8532 \cdot \frac{X_{WMC}^{0{,}0369} - 1}{0{,}0369} + \varepsilon\right] \cdot 0{,}2485 + 1\right)^{\frac{1}{0{,}2485}} \\
\hat{Y}_{CBO} &= \left(\left[4{,}1336 - 0{,}7971 \cdot \frac{X_{WMC}^{0{,}0369} - 1}{0{,}0369} + 1{,}1138 \cdot \frac{X_{RFC}^{0{,}2485} - 1}{0{,}2485} + \varepsilon\right] \cdot 1{,}0492 + 1\right)^{\frac{1}{1{,}0492}} \\
\hat{Y}_{WMC} &= \left(\left[1{,}1486 + 0{,}4837 \cdot \frac{X_{RFC}^{0{,}2485} - 1}{0{,}2485} - 0{,}0596 \cdot \frac{X_{CBO}^{1{,}0492} - 1}{1{,}0492} + \varepsilon\right] \cdot 0{,}0369 + 1\right)^{\frac{1}{0{,}0369}}
\end{aligned} \tag{2.13}$$

Моделі побудовано в межах метрик RFC[1,6000; 13,6222], CBO[2,5714; 9,0952] та WMC[2,1905; 16,2500].

Наступним кроком виконуємо розрахунок показників якості для побудованих нелінійних регресійних моделей. Аналіз їх точності та перевірка адекватності здійснюватиметься за допомогою критеріїв якості R^2, $MMRE$ та $PRED(0.25)$.

Результати оцінювання моделі RFC(CBO, WMC):

$R^2 = 0{,}4772$, $MMRE = 0{,}2777$, $PRED(0.25) = 0{,}5571(39/70)$.

Результати оцінювання моделі CBO(WMC, RFC):

$R^2 = 0{,}1682$, $MMRE = 0{,}2011$, $PRED(0.25) = 0{,}7571(53/70)$.

Результати оцінювання моделі WMC(RFC, CBO):

$R^2 = 0{,}4090$, $MMRE = 0{,}3085$, $PRED(0.25) = 0{,}5714(40/70)$.

Відповідно до отриманих результатів було визначено, що усі три побудовані нелінійні моделі демонструють прийнятну точність прогнозування загальної вибірки проєктів за критеріями R^2, $MMRE$ та $PRED(0.25)$.

Модель RFC(CBO, WMC) характеризується середньою точністю прогнозування ($R^2 = 0{,}4777$) і помірними значеннями $MMRE = 0{,}2777$ та $PRED(0.25) = 0{,}5571$. Це вказує на здатність модель відтворювати основні закономірності в даних, хоча й без забезпечення високої точності у всіх випадках.

Модель CBO(WMC, RFC) демонструє найнижчий рівень пояснювальної здатності серед усіх моделей ($R^2 = 0{,}1666$), проте має найкращі значення $MMRE = 0{,}2011$ та $PRED(0.25) = 0{,}7571$. Це свідчить про те, що модель, попри слабку лінійну залежність, є найбільш стабільною у точковому прогнозуванні та забезпечує високу частку точних передбачень.

Модель WMC(RFC, CBO) забезпечує середню якість прогнозування ($R^2 = 0{,}3978$) при $MMRE = 0{,}3085$ та $PRED(0.25) = 0{,}5714$. Отримані результати вказують на те, що модель загалом є придатною для використання, але її точність є нижчою за інші моделі.

Отже, серед трьох моделей найкращий прогнозний потенціал за більшістю критеріїв оцінювання якості демонструє модель CBO(WMC, RFC), в той час як моделі RFC та WMC мають задовільний, але помірний рівень точності.

Для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, пропонується використовувати довірчі та прогнозні інтервали, побудовані на трьох регресійних моделях: RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) [16]. Такий підхід дає змогу отримувати не лише точкові оцінки кожної з цих метрик, але й визначати діапазони їх можливих значень із заданою ймовірністю, що підвищує надійність та інформативність оцінювання. Інтервали передбачення для нелінійних моделей, сформованих із

застосуванням нормалізації даних методом Бокса-Кокса, визначаються відповідно до формули (2.11).

Формула довірчого інтервалу для нелінійних регресій розраховується аналогічно до інтервалу передбачення, але без доданка «1» під знаком кореня у фігурних дужках:

$$\Psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{\frac{1}{N} + (\mathbf{Z}_X^+)^T[(\mathbf{Z}_X^+)^T\mathbf{Z}_X^+]^{-1}(\mathbf{Z}_X^+)\right\}^{1/2}\right).$$

Значення квантилю розподілу Стьюдента становить 1,9960, яке є однаковим для всіх побудованих моделей.

У таблицях 2.4, 2.5 та 2.6 наведено значення Y, передбачені значення $\hat{Y}$ та розраховані довірчі та прогнозні інтервали моделей RFC, CBO та WMC для кожного застосунку.

Таблиця 2.4 – Передбачені значення та інтервали для моделі RFC

№	Y	$\hat{Y}$	Довірчий нижня	Довірчий верхня	Прогнозний нижня	Прогнозний верхня
1	2	3	4	5	6	7
1	5,4000	5,8148	5,2961	6,3707	2,7995	10,7917
2	9,6290	7,0659	6,3415	7,8507	3,5206	12,7935
3	6,1087	5,5376	5,0736	6,0327	2,6430	10,3412
4	3,0833	3,0216	2,5342	3,5765	1,2420	6,2562
5	4,4000	2,8951	2,3668	3,5075	1,1698	6,0630
6	2,8571	3,5532	2,8481	4,3821	1,4958	7,2397
7	2,1250	4,4021	3,9942	4,8404	2,0013	8,5075
8	8,0000	9,3866	7,9926	10,9558	4,8527	16,5428
9	5,0882	6,9447	6,1196	7,8508	3,4362	12,6380
10	5,6842	3,3060	2,7437	3,9508	1,3837	6,7624
11	7,6000	4,7854	4,3880	5,2093	2,2176	9,1253
12	6,1333	3,5217	3,0351	4,0647	1,5095	7,0880
13	6,6667	8,4411	7,2894	9,7245	4,3031	15,0309
14	7,3333	4,7464	4,3184	5,2055	2,1932	9,0691
15	7,5424	6,3949	5,4995	7,3956	3,0978	11,8161
16	5,7692	5,1587	4,7221	5,6250	2,4271	9,7324
17	3,0476	4,1239	3,0112	5,5203	1,7288	8,4260
18	1,6000	3,7406	3,0825	4,4991	1,6062	7,5194

Продовження таблиці 2.4

1	2	3	4	5	6	7
19	8,5333	8,8187	7,5181	10,2817	4,5132	15,6582
20	1,7500	3,8669	3,3889	4,3940	1,6986	7,6536
21	7,0000	8,4705	7,3706	9,6895	4,3292	15,0550
22	6,3208	7,6573	6,6743	8,7453	3,8465	13,7825
23	2,8000	4,1672	3,6982	4,6796	1,8652	8,1416
24	3,5000	4,3472	3,4425	5,4198	1,9029	8,6246
25	6,4815	5,2865	4,8543	5,7470	2,5007	9,9354
26	4,8378	6,2936	5,7177	6,9120	3,0754	11,5566
27	2,6667	3,5114	2,9807	4,1102	1,4987	7,0879
28	4,7500	5,8105	4,9447	6,7856	2,7563	10,8984
29	5,7931	6,6304	6,0033	7,3055	3,2702	12,0938
30	3,1667	5,0095	4,5084	5,5514	2,3364	9,5095
31	3,0556	3,0830	2,5748	3,6632	1,2718	6,3678
32	6,1842	5,8090	5,1266	6,5575	2,7797	10,8281
33	6,4000	3,7948	3,3426	4,2914	1,6617	7,5269
34	2,6667	4,2867	3,8705	4,7357	1,9363	8,3220
35	3,1667	4,4245	4,0207	4,8581	2,0142	8,5431
36	8,0000	5,5994	5,1528	6,0744	2,6801	10,4354
37	5,4648	5,6517	5,1645	6,1726	2,7073	10,5270
38	10,0930	3,7262	3,2755	4,2219	1,6244	7,4135
39	7,8788	8,3155	7,2597	9,4828	4,2398	14,8052
40	4,2368	5,4927	5,0020	6,0187	2,6149	10,2762
41	7,1667	6,2663	5,5174	7,0892	3,0410	11,5638
42	4,8000	5,2409	4,4553	6,1262	2,4357	9,9733
43	7,4444	7,4818	6,4501	8,6328	3,7325	13,5346
44	4,8333	8,0750	7,0751	9,1775	4,0998	14,4218
45	4,8649	5,2782	4,8549	5,7287	2,4966	9,9203
46	4,2500	3,6919	2,8269	4,7422	1,5421	7,5620
47	5,5000	4,9520	4,5040	5,4327	2,3082	9,4043
48	6,1163	5,7002	5,2197	6,2131	2,7360	10,6021
49	4,6667	4,5922	4,0136	5,2314	2,0916	8,8637
50	2,9444	4,7088	4,3103	5,1345	2,1743	9,0020
51	3,3333	2,8630	2,3588	3,4441	1,1563	5,9975
52	2,2143	3,3577	2,8597	3,9181	1,4198	6,8206
53	13,6222	7,8622	6,8523	8,9802	3,9679	14,1034
54	3,8333	4,2368	3,8279	4,6777	1,9092	8,2389
55	9,2857	8,6294	7,3574	10,0600	4,4002	15,3633
56	7,0189	6,7286	5,7705	7,8017	3,2881	12,3551
57	6,0769	4,0114	3,4432	4,6473	1,7680	7,9211
58	4,8571	4,7715	4,3592	5,2125	2,2086	9,1060
59	3,6667	3,9141	3,4822	4,3852	1,7290	7,7171

Продовження таблиці 2.4

1	2	3	4	5	6	7
60	12,6471	8,4424	7,3502	9,6524	4,3129	15,0098
61	5,0488	5,9280	5,4377	6,4508	2,8680	10,9636
62	4,2857	4,2457	3,8021	4,7270	1,9111	8,2622
63	2,0000	3,4376	2,8722	4,0828	1,4542	6,9808
64	8,2000	8,0176	6,9948	9,1488	4,0613	14,3433
65	13,3056	8,7849	7,3217	10,4576	4,4621	15,6848
66	2,5000	2,7021	2,1084	3,4134	1,0583	5,7765
67	6,6923	5,1766	4,5883	5,8198	2,4234	9,8005
68	5,8750	3,9055	3,1517	4,7873	1,6820	7,8350
69	5,3235	4,5515	4,1531	4,9780	2,0858	8,7477
70	7,4000	6,9772	6,2727	7,7396	3,4695	12,6512

Таблиця 2.5 – Передбачені значення та інтервали для моделі СВО

№	Y	$\hat{Y}$	Довірчий нижня	Довірчий верхня	Прогнозний нижня	Прогнозний верхня
1	2	3	4	5	6	7
1	6,5000	5,6971	5,3852	6,0082	3,0507	8,2821
2	6,5403	6,4024	5,8611	6,9414	3,7377	9,0120
3	6,2391	5,9201	5,5815	6,2578	3,2769	8,5044
4	4,5833	5,4234	4,8851	5,9591	2,7306	8,0490
5	5,4000	6,0853	5,3671	6,7994	3,3693	8,7408
6	2,5714	4,8622	4,3186	5,4028	2,1479	7,4995
7	5,5625	4,6259	3,9658	5,2815	1,8749	7,2933
8	6,5909	5,6761	4,9948	6,3535	2,9579	8,3291
9	7,4118	5,4904	5,1333	5,8463	2,8314	8,0847
10	6,1053	6,4205	5,6417	7,1947	3,6962	9,0873
11	5,6000	6,3784	5,8655	6,8893	3,7189	8,9828
12	4,1333	6,1999	5,6964	6,7014	3,5374	8,8054
13	6,0000	5,4456	4,7924	6,0949	2,7272	8,0957
14	6,0000	6,3835	5,8506	6,9143	3,7200	8,9917
15	4,3390	5,8152	5,3134	6,3149	3,1421	8,4269
16	5,1923	5,7778	5,4594	6,0953	3,1330	8,3621
17	9,0952	5,6672	4,9357	6,3940	2,9352	8,3331
18	3,0000	4,1331	3,2216	5,0347	1,2834	6,8800
19	5,8667	5,7804	5,0990	6,4579	3,0654	8,4316
20	4,2500	4,3686	3,5831	5,1472	1,5712	7,0734
21	6,6667	5,6131	5,0482	6,1752	2,9208	8,2407
22	7,5849	5,7188	5,3346	6,1017	3,0631	8,3128
23	4,4000	4,8894	4,3581	5,4179	2,1788	7,5238
24	8,1667	5,6623	5,0802	6,2415	2,9677	8,2926

Продовження таблиці 2.5

1	2	3	4	5	6	7
25	5,3704	5,9569	5,6084	6,3044	3,3134	8,5418
26	6,5135	5,4240	5,0565	5,7902	2,7613	8,0209
27	6,0000	5,2509	4,6774	5,8213	2,5447	7,8868
28	3,8750	5,1422	4,5674	5,7138	2,4318	7,7804
29	6,4310	5,6206	5,2517	5,9883	2,9641	8,2142
30	4,6667	4,8601	4,2964	5,4205	2,1414	7,5014
31	5,3333	5,4908	4,9002	6,0783	2,7889	8,1260
32	4,5526	5,6461	5,2387	6,0521	2,9846	8,2448
33	5,5000	6,3653	5,7521	6,9757	3,6836	8,9909
34	5,5833	4,9524	4,4302	5,4719	2,2461	7,5838
35	5,1667	5,0862	4,6340	5,5364	2,3982	7,7024
36	5,9091	6,3059	5,8417	6,7685	3,6537	8,9026
37	5,3521	5,5955	5,2518	5,9382	2,9418	8,1862
38	4,5814	7,0385	6,1661	7,9057	4,3000	9,7243
39	6,9091	5,8725	5,3574	6,3853	3,1985	8,4856
40	6,4474	5,3987	5,0539	5,7423	2,7384	7,9931
41	4,7500	5,8128	5,3643	6,2595	3,1495	8,4150
42	7,7000	5,8193	5,4172	6,2201	3,1639	8,4142
43	7,9259	6,0685	5,6699	6,4659	3,4209	8,6586
44	7,0556	5,1483	4,5635	5,7298	2,4359	7,7885
45	5,9459	5,5876	5,2738	5,9006	2,9375	8,1748
46	8,0000	6,1138	5,3298	6,8929	3,3798	8,7867
47	6,2000	5,8933	5,5353	6,2502	3,2467	8,4806
48	6,2791	5,8895	5,5571	6,2211	3,2463	8,4736
49	4,0000	5,4370	5,0909	5,7820	2,7778	8,0309
50	5,6111	4,9702	4,4731	5,4648	2,2695	7,5965
51	4,8974	5,6334	5,0240	6,2395	2,9316	8,2701
52	4,0000	4,7906	4,1825	5,3949	2,0592	7,4428
53	7,5111	7,0313	6,2136	7,8443	4,3099	9,7004
54	5,1667	5,4074	5,0373	5,7761	2,7438	8,0050
55	8,0000	6,2553	5,7303	6,7781	3,5900	8,8640
56	8,0189	6,1310	5,7322	6,5286	3,4851	8,7200
57	3,6923	5,9713	5,6141	6,3274	3,3270	8,5571
58	5,8571	5,7023	5,3705	6,0332	3,0535	8,2896
59	5,3333	5,4635	5,0503	5,8751	2,7952	8,0664
60	6,9412	6,7084	5,9793	7,4335	4,0055	9,3572
61	5,9756	5,4953	5,1449	5,8447	2,8375	8,0887
62	5,8571	5,6612	5,2692	6,0520	3,0026	8,2574
63	3,4000	4,5552	3,8606	5,2446	1,7921	7,2324
64	5,9500	5,8614	5,2918	6,4283	3,1756	8,4856
65	5,1111	6,4980	5,6661	7,3246	3,7596	9,1790

Продовження таблиці 2.5

1	2	3	4	5	6	7
66	2,6667	5,0126	4,4640	5,5584	2,3031	7,6480
67	4,3846	5,9023	5,5428	6,2607	3,2557	8,4896
68	2,6250	5,8077	5,4865	6,1281	3,1635	8,3918
69	5,3088	5,8268	5,4861	6,1665	3,1806	8,4130
70	6,1400	5,9092	5,4785	6,3383	3,2518	8,5069

Таблиця 2.6 – Передбачені значення та інтервали для параметру WMC

№	Y	$\hat{Y}$	Довірчий нижня	Довірчий верхня	Прогнозний нижня	Прогнозний верхня
1	2	3	4	5	6	7
1	6,2000	5,7872	5,2281	6,4036	2,7597	11,8991
2	8,6210	8,8337	7,6043	10,2534	4,2245	18,1138
3	5,9783	6,4092	5,8263	7,0479	3,0674	13,1325
4	3,0833	4,5285	3,9677	5,1652	2,1344	9,4143
5	2,5000	5,3924	4,9180	5,9108	2,5698	11,0941
6	5,5714	4,8805	3,9411	6,0337	2,2613	10,3121
7	4,6250	3,4352	2,8614	4,1190	1,5897	7,2665
8	14,2727	7,6387	6,7494	8,6403	3,6571	15,6466
9	7,1765	5,2528	4,5344	6,0802	2,4787	10,9080
10	2,6842	6,1429	5,6089	6,7257	2,9383	12,5938
11	5,2571	7,8048	6,9380	8,7755	3,7427	15,9626
12	4,2000	7,2955	6,3092	8,4295	3,4745	15,0185
13	13,0000	6,9221	6,2722	7,6366	3,3183	14,1616
14	4,8333	7,4203	6,6552	8,2697	3,5585	15,1752
15	10,5932	8,3676	7,1310	9,8094	3,9871	17,2168
16	6,3846	6,5599	5,9546	7,2241	3,1409	13,4358
17	2,1905	3,3960	2,5122	4,5754	1,5132	7,4445
18	5,6000	3,4486	2,7186	4,3655	1,5717	7,4008
19	14,4000	8,3796	7,3449	9,5540	4,0160	17,1474
20	4,7500	3,3509	2,7343	4,1003	1,5414	7,1283
21	11,6667	6,8870	6,1529	7,7051	3,2941	14,1196
22	8,2830	6,0472	5,2151	7,0064	2,8639	12,5149
23	5,2000	4,3225	3,7369	4,9961	2,0296	9,0187
24	2,8333	3,9231	3,1253	4,9152	1,7999	8,3662
25	6,4444	7,0463	6,3607	7,8028	3,3777	14,4164
26	7,0811	5,3654	4,8205	5,9694	2,5512	11,0617
27	3,0000	3,8073	3,2448	4,4631	1,7762	7,9922
28	9,7500	6,2179	5,3921	7,1647	2,9502	12,8454
29	7,8621	6,1024	5,5288	6,7330	2,9153	12,5252
30	6,6667	4,5792	4,0289	5,2015	2,1606	9,5103

Продовження таблиці 2.6

1	2	3	4	5	6	7
31	2,7778	4,3024	3,7834	4,8895	2,0263	8,9505
32	8,6842	7,1573	6,3093	8,1145	3,4193	14,6905
33	3,7000	6,9285	6,2788	7,6428	3,3215	14,1741
34	4,4167	3,9066	3,3617	4,5361	1,8275	8,1795
35	5,0000	4,4418	3,9257	5,0229	2,0956	9,2252
36	6,4545	7,9601	7,0514	8,9810	3,8169	16,2810
37	7,2254	6,2572	5,7164	6,8471	2,9946	12,8212
38	4,2326	10,3024	8,4510	12,5414	4,8893	21,2791
39	10,8182	7,4070	6,5107	8,4216	3,5400	15,1972
40	5,6842	4,9339	4,3960	5,5349	2,3377	10,2067
41	9,5417	7,8648	6,8718	8,9952	3,7607	16,1288
42	4,2000	4,9600	4,1944	5,8594	2,3266	10,3586
43	7,4815	6,6718	5,6469	7,8745	3,1551	13,8262
44	10,0000	5,1860	4,5460	5,9124	2,4536	10,7427
45	5,8108	5,5760	5,0939	6,1021	2,6603	11,4596
46	2,2500	4,4901	3,6895	5,4568	2,0857	9,4642
47	5,0000	5,9695	5,4415	6,5466	2,8525	12,2498
48	6,2326	6,3992	5,8125	7,0427	3,0621	13,1140
49	6,5000	6,0996	5,3254	6,9816	2,8964	12,5916
50	5,1111	4,1361	3,6072	4,7393	1,9430	8,6258
51	2,6923	4,6595	4,1349	5,2479	2,2028	9,6592
52	4,0000	3,8688	3,2347	4,6217	1,7978	8,1516
53	8,8000	11,0535	8,8632	13,7604	5,2209	22,9333
54	4,6667	5,0030	4,5111	5,5464	2,3756	10,3282
55	9,5714	7,8580	6,5513	9,4140	3,7187	16,2738
56	6,0943	6,3517	5,3542	7,5270	2,9970	13,1912
57	5,5385	7,4399	6,2927	8,7871	3,5280	15,3782
58	5,0000	5,6003	5,1225	6,1209	2,6726	11,5064
59	4,0000	4,8151	4,3260	5,3571	2,2828	9,9550
60	11,0588	10,7473	8,7887	13,1228	5,1017	22,1929
61	7,0244	5,7072	5,2181	6,2402	2,7248	11,7209
62	4,1429	5,1545	4,6775	5,6781	2,4517	10,6236
63	4,6000	3,7918	3,0901	4,6457	1,7495	8,0436
64	11,9500	8,0895	7,1416	9,1580	3,8783	16,5483
65	16,2500	12,5014	9,8661	15,8082	5,8939	25,9828
66	3,6667	4,4895	3,6198	5,5588	2,0746	9,5099
67	7,3846	7,6482	6,6339	8,8111	3,6492	15,7169
68	6,3750	7,7341	6,1766	9,6665	3,6151	16,2058
69	5,1029	6,1617	5,6296	6,7421	2,9477	12,6304
70	9,0400	7,4070	6,6353	8,2648	3,5513	15,1515

Отже, в результаті було успішно побудовано таблицю довірчих та прогнозних інтервалів трьох нелінійних регресійних моделей, які в подальшому будуть використані для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

2.3 Оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Оцінювання якості застосунків з відкритим кодом є необхідним та важливим кроком для забезпечення надійності, стабільності та довготривалої підтримки застосунків. Відповідно до побудованих таблиць 2.4-2.6 з довірчими та прогнозними інтервалами для кожної з моделей RFC, CBO та WMC було визначено чіткі критерії для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Проєкт має низьку якість у випадку, якщо фактичне значення залежної змінної Y перевищує верхню межу довірчого інтервалу, що свідчить про надлишкову складність, яка має здебільшого негативний вплив на підтримку та масштабування проєкту.

Проєкт має середню якість у випадку, якщо фактичне значення залежної змінної Y знаходиться в межах довірчого інтервалу, що свідчить про помірну складність коду при заданих незалежних змінних X_1 та X_2.

В усіх інших випадках проєкт має високу якість. В такому випадку вважається, що складність коду є нижчою за очікувану.

У таблиці 2.7 наведено результати оцінювання якості застосунків, що включають назви проєктів, рівні якості за кожною з метрик та загальну оцінку якості застосунків, визначену за мінімальними значеннями серед усіх метрик.

Таблиця 2.7 – Результати оцінювання якості застосунків

№	Застосунок	Якість за моделлю RFC	Якість за моделлю CBO	Якість за моделлю WMC	Загальна якість проєкту
1	2	3	4	5	6
1	adixchen/demo-restWS-spring-jersey-jpa2-hibernate	середня	низька	середня	низька
2	ajanata/PretendYoureXyzzy	низька	середня	середня	низька
3	akshu20791/spring-hibernate-security-react-ant-design-polling-app	низька	середня	середня	низька
4	anakiou/multitenancy	середня	висока	висока	середня
5	bcssp10/multi-tenant	низька	середня	висока	низька
6	bdot/struts-hibernate	середня	висока	середня	середня
7	benssj5/Projet-JEE-Gestion-des-Comptes-Spring-JPA-Hibernate	висока	низька	низька	низька
8	BhartiKhankure/Vehicle_insurance_management_system	середня	низька	низька	низька
9	brishi19791/HealthCare-WebApplication	висока	низька	низька	низька
10	callistaenterprise/blog-multitenancy	низька	середня	висока	низька
11	cassiomolin/jersey-jwt	низька	висока	висока	низька
12	Cepr0/sb-multitenant-db-demo	низька	висока	висока	низька
13	chrisssss/springboot-hibernatespatial-postgis-demo	висока	середня	низька	низька
14	codedecode25/HibernateDemo	низька	середня	висока	низька
15	coderzc/biubiu	низька	висока	низька	низька
16	CoffeeLatte007/sharebook-jersey-jwt-spring-hibernate	низька	висока	середня	низька
17	colinbut/sales-order-system	середня	низька	висока	низька
18	cr2121/spring	висока	висока	низька	низька
19	cyela/Spring-Web-App	середня	середня	низька	низька
20	dadoonet/hsearch-es-demo	висока	середня	низька	низька
21	daliasheasha/HibernateApp	висока	низька	низька	низька
22	desperatecat/ShoppingPortal_Vue_Springboot_Project	висока	низька	низька	низька
23	devdilip/todo-springboot-mysql-hibernate-rest	висока	середня	низька	низька
24	diegoneuralo/maven	середня	низька	висока	низька
25	discospiff/JavaFullStackEnterpriseWeb	низька	висока	середня	низька
26	dmitrykhramov/Online-Bank-Simulator	висока	низька	низька	низька

Продовження таблиці 2.7

1	2	3	4	5	6
27	dobermai/Hibernate-Flyway-Integration	висока	низька	висока	низька
28	F4bwDP6a6W/spring-mvc-REST	висока	висока	низька	низька
29	foysal-mahmud/E-BookShop----Spring-boot	висока	низька	низька	низька
30	frenmanoj/bookstore	висока	середня	низька	низька
31	Fruzenshtein/security-spr	середня	середня	висока	середня
32	IcedSoul/Shopping	середня	висока	низька	низька
33	jamesward/spring_hibernate_hstore_demo	низька	висока	висока	низька
34	jasenkoh/spring-boot-jersey-hibernate	висока	низька	середня	низька
35	Java-Techie-jt/Payment-Resource	висока	середня	середня	середня
36	jjoe64/shiroexample	низька	середня	висока	низька
37	Joryun/Shop	середня	середня	низька	низька
38	Karszt/sda-zdjava124	низька	висока	висока	низька
39	khozema-nullwala/online-shopping	середня	низька	низька	низька
40	lastfm/musicbrainz-data	висока	низька	низька	низька
41	licyun/SpringMVCgbook	низька	висока	низька	низька
42	michaelisvy/hibernate-4-spring-3.1-samples	середня	низька	середня	низька
43	MiteshPuthran/Shopping-Cart-Spring-MVC-Hibernate-Application	середня	низька	середня	низька
44	mohamedYoussfi/eboutique-jpa-hibernate-spring	висока	низька	низька	низька
45	MoonSulong/Ecommerce	середня	низька	середня	низька
46	OKaluzny/spring-boot-rest-api-postgresql	середня	низька	висока	низька
47	OKaluzny/springboot-rest-api-angularjs-https	низька	середня	висока	низька
48	oregami/dropwizard-guice-jpa-seed	середня	низька	середня	низька
49	pfac/demo-spring-data-jpa-hibernate-h2	середня	висока	середня	середня
50	quarkiverse/quarkus-hibernate-types	висока	низька	низька	низька
51	radoslaw-piwowarczyk/PetClinic-Spring-Thymeleaf-Hibernate	середня	висока	висока	середня
52	rajbhatta/springboot_restapi_hibernate	висока	висока	середня	середня
53	ralscha/eds-starter6-jpa	низька	середня	висока	низька
54	RameshMF/springboot-thymeleaf-crud-pagination-sorting-webapp	середня	середня	середня	середня
55	rampatra/springboot-hibernate	середня	низька	низька	низька
56	rid17pawar/HospitalManagement	середня	низька	середня	низька

Продовження таблиці 2.7

1	2	3	4	5	6
57	royvanrijn/graalvm-native-microservice	низька	висока	висока	низька
58	rutujaBacchuwar/MuzixAppHibernate	середня	середня	висока	середня
59	scbushan05/book-api	середня	середня	висока	середня
60	scottescue/dropwizard-entitymanager	низька	середня	середня	низька
61	soulcode-acad/goservice	висока	низька	низька	низька
62	sroysf/webapp-springmvc-jpa-hibernate	середня	середня	висока	середня
63	stevegocoding/SpringMVC-SpringDataJpa-Hibernate-Demo	висока	висока	середня	середня
64	swapnildipankar/java_jersey_spring_hibernate_maven	середня	середня	низька	низька
65	ThxGodNovember/AprilDragonSpring	низька	висока	низька	низька
66	tolitius/money-making-project	середня	висока	середня	середня
67	vasylmalik/project-hibernate-1	низька	висока	середня	низька
68	yannbriancon/spring-hibernate-query-utils	низька	висока	середня	низька
69	ykameshrao/spring-mvc-angular-js-hibernate-bootstrap-java-single-page-jwt-auth-rest-api-webapp-framework	низька	висока	висока	низька
70	ZhuXS/Spring-Shiro-Spark	середня	середня	низька	низька

В ході аналізу якості 70 застосунків, наведених в таблиці 2.7, були отримані наступні результати:

За моделлю RFC(CBO, WMC): високу якість має 21 застосунок (30.00%), середню якість мають 27 застосунків (38.57%) та низьку якість мають 22 застосунки (31.43%).

За моделлю CBO(WMC, RFC): високу якість мають 23 застосунки (32.86%), середню якість мають 23 застосунки (32.86%) та низьку якість мають 24 застосунки (34.29%).

За моделлю WMC(RFC, CBO): високу якість мають 22 застосунки (31.43%), середню якість має 21 застосунок (30.00%) та низьку якість мають 27 застосунків (38.57%).

Відповідно до загальної оцінки якості застосунків: високу якість мають 0 застосунків, середню якість мають 13 застосунків (18,57%) та низьку якість мають 57 застосунків (81,43%).

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ HIBERNATE

3.1 Аналіз вимог до програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Базуючись на проведеному аналізі предметної області та розроблених нелінійних регресійних моделях було сформульовано вимоги до програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Розроблюване програмне забезпечення отримало назву «JHQualityMeter» та має виконувати наступні функції:

– Завантаження вхідних даних: завантаження файлу з розширенням CSV із вхідними даними у вигляді таблиці з URL адресами застосунків та метриками RFC, CBO, WMC.

– Попередня обробка даних: перевірка метрик тестом Мардіа на нормальність розподілу, одновимірна нормалізація даних методом Бокса-Кокса для кожного параметру, очищення даних за наявності викидів та автоматичне формування файлу з обробленими даними для подальшої побудови моделей.

– Побудова моделей: побудова лінійних та нелінійних регресійних рівнянь для трьох моделей: RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO) на основі оброблених даних з автоматичним відображенням оцінок точності побудованих нелінійних моделей: $R^2, MMRE$ та $PRED(0.25)$.

– Оцінювання якості застосунків: побудова довірчих інтервалів та інтервалів прогнозування для подальшого оцінювання якості завантажених застосунків (класифікуючи кожен з застосунків за шкалою оцінок «висока»,

«середня» та «низька»), а також оцінювання якості будь-якого стороннього застосунку шляхом вводу у текстові поля метрик RFC, CBO, WMC на основі побудованих моделей.

Окрім функціональних вимог, в програмному забезпеченні передбачені також вимоги до організації вхідних та вихідних даних:

Вхідні дані:

– Файл формату CSV з таблицею URL адрес застосунків та метрик RFC, CBO, WMC до них.

– Параметри RFC, CBO та WMC для оцінювання якості стороннього застосунку на основі побудованих моделей.

Вихідні дані:

– Файл з обробленими даними у форматі CSV.

– Лінійні моделі регресій (2.9): RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO).

– Нелінійні моделі регресій (2.10): RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO).

– Оцінки точності побудованих нелінійних моделей: $R^2, MMRE$ та $PRED(0.25)$.

– Таблиця з довірчими інтервалами та інтервалами прогнозування для кожної з нелінійних моделей.

– Оцінка якості кожного застосунку загальної вибірки та опціонально для будь-якого стороннього застосунку на основі побудованих моделей, що класифікуються за шкалою оцінок: «висока», «середня» та «низька».

Технічне завдання на розробку програмного забезпечення «JHQualityMeter» наведено в Додатку А.

Сформульовані вимоги передбачають наявність всього необхідного функціоналу для простої та ефективної роботи розроблюваного програмного забезпечення «JHQualityMeter» та мають бути використані в ході проєктування та реалізації програмного забезпечення.

Також, відповідно до аналізу вимог до програмного забезпечення «JHQualityMeter», було розроблено прототипи вікон програми, які проілюстровано на рисунках 3.1 – 3.3.

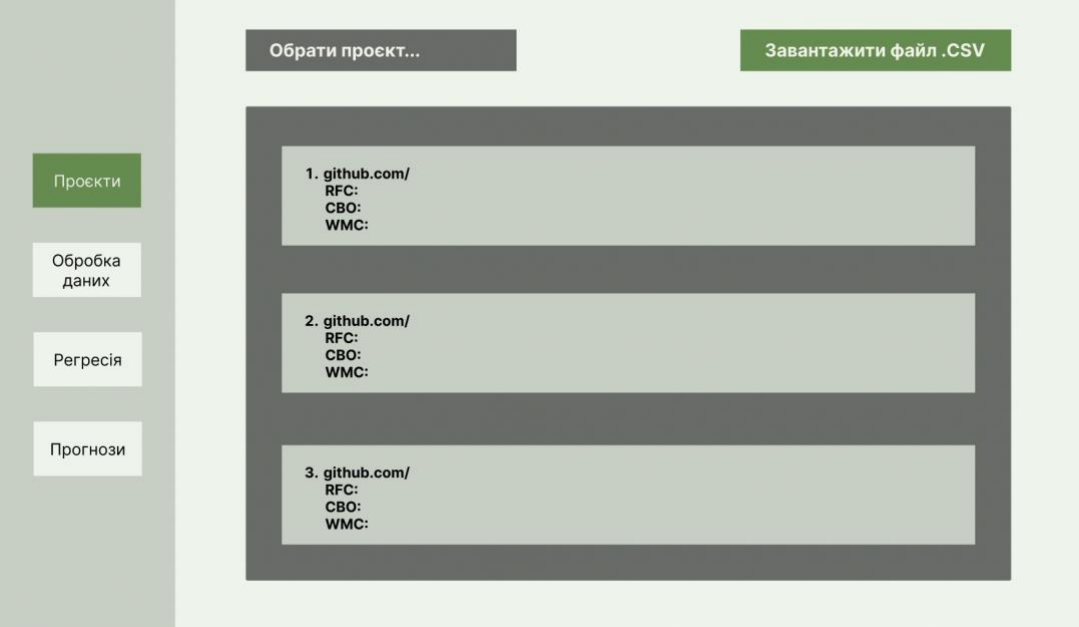


Рисунок 3.1 – Прототип головного екрана програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate «JHQualityMeter»

Головне вікно матиме бічну панель зліва, де повинні розташовуватись вкладки «Проєкти», «Обробка даних», «Регресія» та «Оцінювання», а також основну панель, інтерфейс якої має змінюватись залежно від обраної вкладки. На першій вкладці «Проєкти» мають міститися елементи для додавання файлу з вхідними даними, розкривний список із завантаженими файлами та список завантажених застосунків. Кожен елемент списку матиме такі атрибути: номер застосунку, URL, та метрики RFC, CBO та WMC.

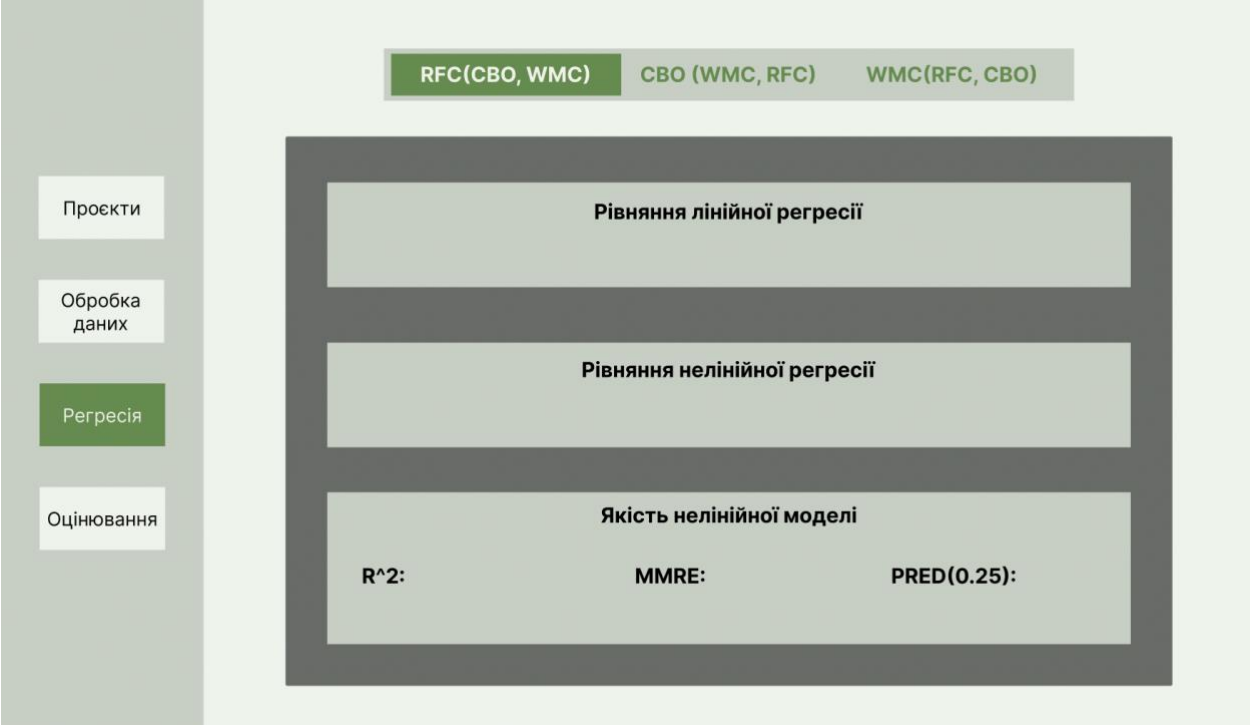


Рисунок 3.2 – Прототип екрана «Регресія» програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate «JHQualityMeter»

На вкладці «Регресія» має знаходитись панель вибору моделі для якої будуватимуться регресійні рівняння, нижче мають відображатися побудовані за обраною моделлю лінійне (2.9) та нелінійне рівняння регресії (2.10), а також оцінки точності побудованої нелінійної моделі: коефіцієнт детермінації R^2, середня величина відносної похибки $MMRE$ та відсоток прогнозованих результатів у допустимому діапазоні $PRED(0.25)$.

Рисунок 3.3 – Прототип екрана «Оцінювання» програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate «JHQualityMeter»

Вкладка «Оцінювання» має містити в собі панель вибору моделі та відображати розкривний список застосунків завантаженої вибірки. Завдяки текстовим полям для вводу метрик буде реалізована можливість оцінювати якість не лише застосунків із завантаженої вибірки, але й сторонніх застосунків, які будуть оцінюватись на основі вже побудованих нелінійних моделей. Відповідна оцінка прогнозу має відображатись під текстовими полями для вводу метрик та класифікувати застосунки на ті, що мають «високу», «середню» та «низьку» якість. Також в нижній частині вкладки має бути наведено діапазони допустимих значень метрик RFC, CBO та WMC, які розраховуватимуться із завантажених даних.

3.2 Розробка архітектури програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Одним із ключових факторів створення якісного та надійного програмного забезпечення є обґрунтований вибір архітектурного підходу, відповідно до якого реалізується програмний продукт. Архітектура визначає принципи організації коду, взаємодію між його компонентами, а також можливість подальшого супроводу програмного забезпечення. Основними вимогами під час вибору архітектури програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate є: можливість простого масштабування, структурованість та чітке розділення коду на різні функціональні блоки, зручність тестування та модифікації програмного забезпечення.

Розробка архітектури програмного забезпечення передбачає подання загальної структури системи, її основних компонентів та механізмів взаємодії між ними. Для реалізації програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate було обрано архітектурний шаблон MVC (Model–View–Controller), який є одним із найпоширеніших шаблонів для розробки програмних застосунків. Популярність цього шаблону обумовлена його простотою для розуміння та чітким поділом програмного забезпечення на три тісно пов’язані між собою компоненти: модель даних, представлення та керуючий модуль [18].

Компонент «Model» відповідає за доступ до керування та модифікації даних з бази даних через керуючий модуль, з яким взаємодіє користувач.

Компонент «View» призначений для відображення інформації користувачеві та формування графічного інтерфейсу програмного забезпечення.

Компонент «Controller» забезпечує взаємодію між моделлю та представленням, перетворюючи дії користувача на запити до даних та відображаючи результат виконання запитів у інтерфейсі.

Загальну схему взаємодії компонентів архітектурного шаблону MVC наведено на рисунку 3.4.

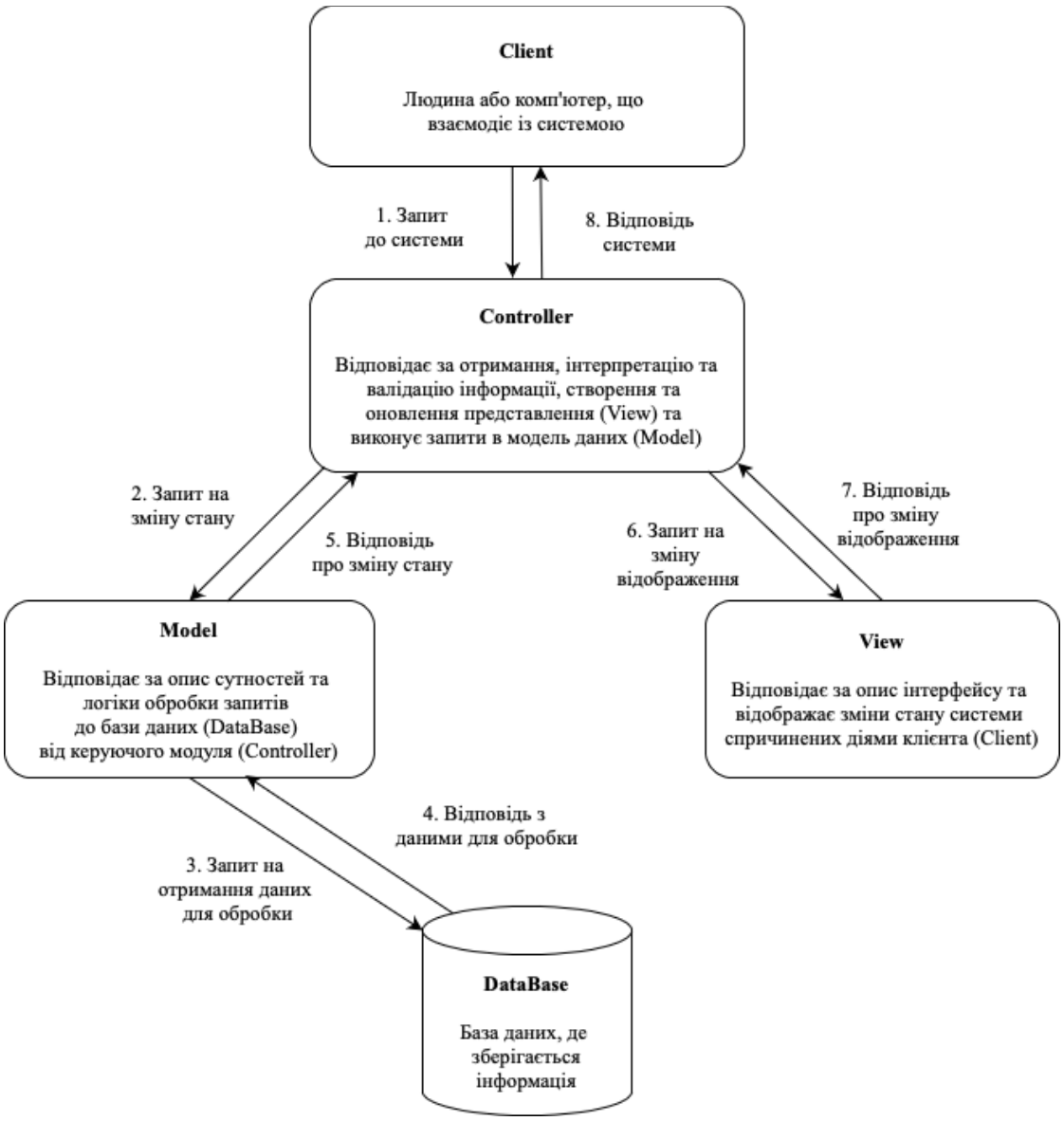


Рисунок 3.4 – Шаблон архітектури MVC (Model-View-Controller)

Варто зазначити, що в традиційному варіанті використання архітектури MVC компонент «Model» зазвичай взаємодіє з базою даних. Проте в межах розроблюваного програмного забезпечення використання системи керування базами даних не буде передбачено. Оброблені в ході роботи застосунку дані

планується зберігати у зовнішніх файлах через взаємодію з файловою системою. Такий підхід допоможе спростити процес розробки програмного забезпечення, пришвидшити його роботу та підвищити безпеку, адже взаємодія з даними відбуватиметься локально.

3.3 Розробка проєкту програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

Важливою та невід'ємною частиною розробки програмного забезпечення після формулювання аналізу вимог та вибору архітектурного шаблону є технічний опис програмного забезпечення, який буде виконано в три етапи. Першим етапом є створення статичної моделі застосунку «JHQualityMeter», що реалізовується шляхом побудови діаграми класів, другим етапом є створення динамічної моделі, яка описується діаграмами послідовностей та третім кроком – обґрунтування вибору мови програмування та засобів розробки.

Статична модель

Для деталізації розроблюваного програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate було розроблено діаграму класів, яку також називають «статичною діаграмою» через властивість описувати програмне забезпечення шляхом відображення статичних зв'язків між класами, що містять в собі атрибути та методи. Серед найбільш розповсюджених зв'язків між класами виокремлюють: асоціацію, агрегацію, композицію, успадкування, реалізацію та залежність [19].

Результат побудови діаграми класів програмного забезпечення «JHQualityMeter» проілюстровано на рисунку 3.5.

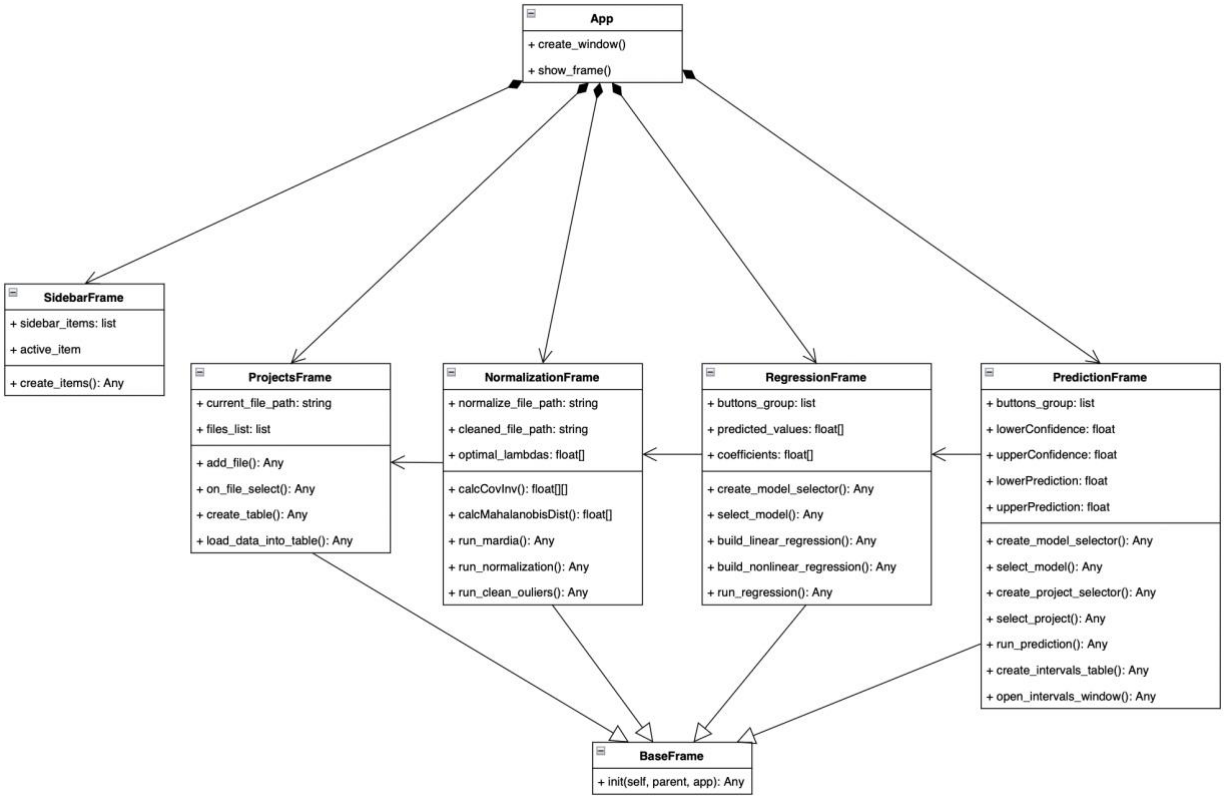


Рисунок 3.5 – Діаграма класів програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate «JHQualityMeter»

Наведена діаграма класів UML містить опис класів програмного забезпечення «JHQualityMeter» та зв’язків між ними.

Клас «App» є головним класом, який запускає головне вікно застосунку, відповідає за переключання між вкладками та пов’язує між собою роботу класів «SidebarFrame», «ProjectFrame», «NormalizationFrame», «RegressionFrame» та «PredictionFrame» та має зв’язок асоціації з ними.

Клас «SidebarFrame» відповідає за візуалізацію бічної панелі з вкладками та має зв’язок агрегації з класом «App».

Клас «ProjectFrame» відповідає за завантаження файлів з вхідними даними та їх відображення. Має зв’язок агрегації з класом «App» та успадковується від класу «BaseFrame».

Клас «NormalizationFrame» відповідає за обробку вхідних даних, яка включає перевірку на нормальність розподілу тестом Мардіа, розрахунок квадратів відстаней Махаланобіса, пошук оптимальних лямбд для

нормалізації методом Бокса-Кокса, нормалізацію даних та видалення викидів у нормалізованих даних. Має зв’язок агрегації з класом «App», асоціації з класом «ProjectFrame» та успадковується від класу «BaseFrame».

Клас «RegressionFrame» відповідає за побудову лінійного та нелінійного регресійних рівнянь на нормалізованих даних за обраною моделлю: RFC(CBO, WMC), CBO(WMC, RFC) або WMC(RFC, CBO), розрахунок коефіцієнтів регресії на основі вибірки та показників якості моделей: $R^2, MMRE$ та $PRED(0.25)$. Має зв’язок агрегації з класом «App», асоціації з класом «NormalizationFrame» та успадковується від класу «BaseFrame».

Клас «PredictionFrame» відповідає за розрахунок довірчих інтервалів та інтервалів прогнозування на основі побудованого нелінійного рівняння регресії за обраною моделлю, оцінювання якості кожного з застосунків загальної вибірки та стороннього застосунку за введеними метриками RFC, CBO та WMC. Має зв’язок агрегації з класом «App», асоціації з класом «RegressionFrame» та успадковується від класу «BaseFrame».

Клас «BaseFrame» є базовим класом для побудови вкладок програмного застосунку.

Динамічна модель

Наступним етапом для відображення динамічної взаємодії між об’єктами та їхніми компонентами є побудова діаграм послідовності.

Діаграми послідовності належать до групи UML-діаграм взаємодії та призначені для відображення порядку виконання операцій, а також процесів приймання та передачі повідомлень між елементами системи (об’єктами) в межах окремого сценарію функціонування. Повідомлення можуть містити в собі як узагальнені дії, так і конкретні методи класів, опис яких було наведено раніше на діаграмі класів (рисунок 3.5) [20].

Діаграми послідовності для прецедентів «Завантажити CSV», «Побудувати лінійне та нелінійне рівняння за обраною моделлю» та «Оцінити якість стороннього застосунку» наведено на рисунках 3.6-3.8.

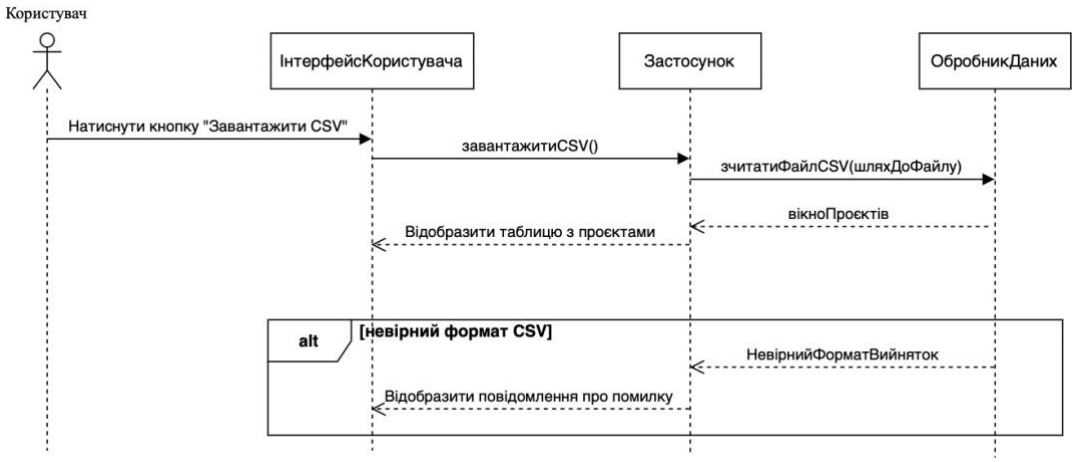


Рисунок 3.6 – Діаграма послідовності для прецеденту «Завантажити CSV»

Діаграма послідовності (рисунок 3.6) ілюструє процес завантаження файлу формату CSV в застосунку «JHQualityMeter». Користувач завантажує файл CSV, через інтерфейс користувача, внаслідок чого застосунок зчитує файл та в залежності від правильності завантаженого формату або відображає таблицю з застосунками в інтерфейсі, або повідомляє про помилку.

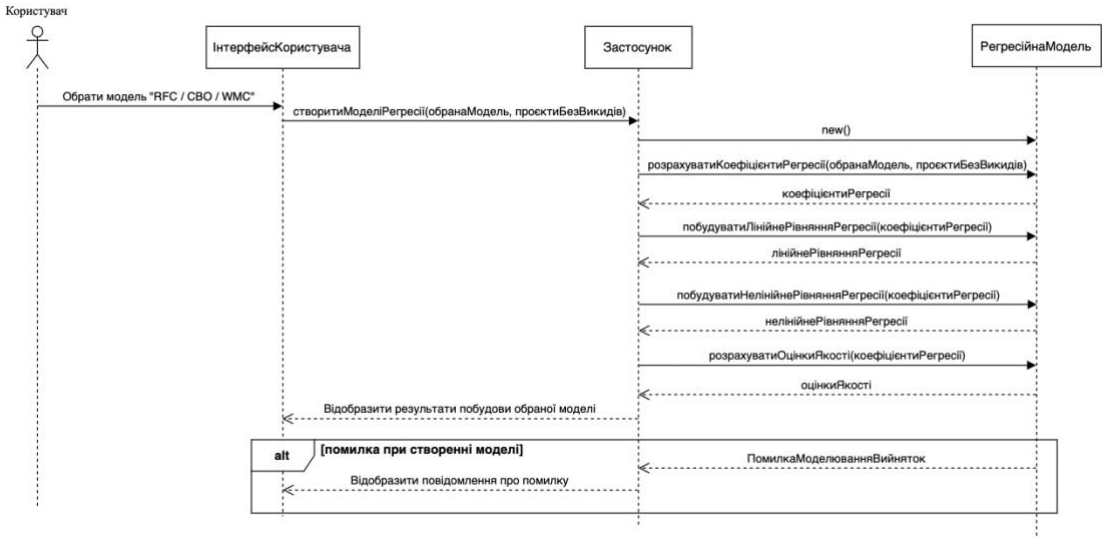


Рисунок 3.7 – Діаграма послідовності для прецеденту «Побудувати лінійне та нелінійне рівняння за обраною моделлю»

Діаграма послідовності (рисунок 3.7) ілюструє процес побудови лінійного та нелінійного рівняння регресії за обраною моделлю на основі завантажених вхідних даних в застосунку «JHQualityMeter». Користувач обирає одну з моделей RFC(CBO, WMC), CBO(WMC, RFC) або WMC(RFC, CBO), натискає на відповідну кнопку, після чого очищені від викидів дані завантажуються у функції побудови рівнянь, що ініціює створення нового об’єкту типу Регресійна Модель, розраховуються коефіцієнти регресії, будуються лінійне та нелінійне регресійне рівняння, розраховуються оцінки якості обраної нелінійної моделі та відображаються в інтерфейсі користувача. У випадку виникнення помилки під час створення моделі, наприклад якщо користувач натиснув на кнопку вибору моделей перед тим, як завантажив файл з даними – в інтерфейсі відобразиться повідомлення про помилку.

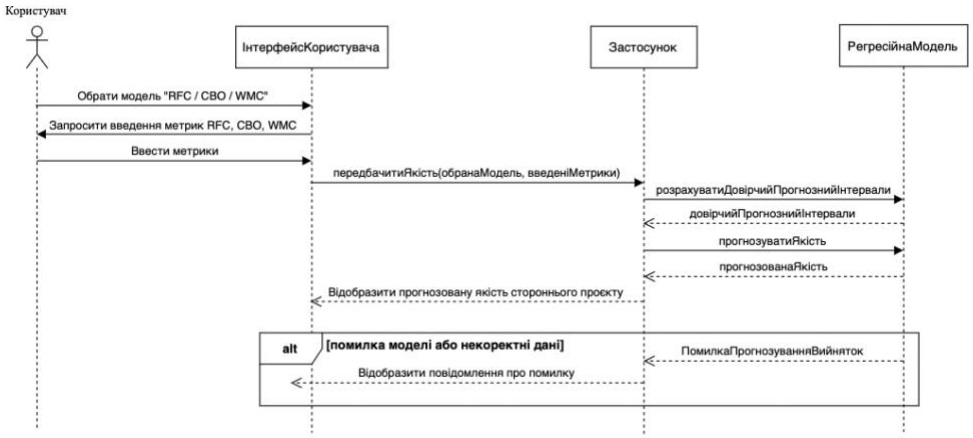


Рисунок 3.8 – Діаграма послідовності для прецеденту «Оцінити якість стороннього застосунку»

Діаграма послідовності (рисунок 3.8) ілюструє процес оцінювання якості стороннього застосунку в програмі «JHQualityMeter». Після вибору в інтерфейсі моделі за якою користувач бажає отримати прогноз якості, в користувача запитується введення в текстові поля відповідних метрик RFC, CBO, WMC застосунку. Після цього дані разом з обраною моделлю передаються у функцію, яка розраховує на основі обраної моделі та введених даних довірчий та прогнозний інтервали, наступним кроком дані

порівнюються з інтервалами та в залежності від результату порівняння користувачу в інтерфейс виводиться відповідна оцінка застосунку («висока», «середня» або «низька»). У випадку некоректно введених вхідних даних або якщо користувач натиснув на кнопку вибору моделей перед тим, як завантажив файл з даними – в інтерфейсі відобразиться повідомлення про помилку.

Обґрунтування вибору мови програмування та засобів розробки

Для розробки програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, було обрано мову програмування Python, графічну бібліотеку Tkinter та середовище розробки Visual Studio Code [21].

Вибір мови програмування Python був обумовлений, в першу чергу, завдяки великій популярності та розповсюдженості цієї мови в серед сучасних розробників програмного забезпечення. Оскільки мова високорівнева та динамічно типізована, вона широко використовується у сферах математичного моделювання, що в поєднанні з простим синтаксисом стає вирішальним фактором вибору саме цієї мови програмування.

Серед основних переваг мови програмування Python можна виокремити:

1. Доступні математичні бібліотеки. Python має широкий набір бібліотек для чисельних обчислень, таких як NumPy, SciPy, pandas та statsmodels, що дозволяє ефективно виконувати обробку даних та побудову нелінійних регресійних моделей.

2. Універсальність. Завдяки простому та лаконічному синтаксису мови, Python використовується в найрізноманітніших сферах, від класичної розробки програмного забезпечення до наукових досліджень та штучного інтелекту [22].

3. Гнучкість. Мова Python підтримує також різні стилі програмування, серед яких об'єктно-орієнтований, функціональний та процедурний підходи.

4. Кросплатформеність. Оскільки мова Python підтримується на всіх основних операційних системах – її використання здатне значно полегшити та пришвидшити адаптацію розроблюваного програмного забезпечення для інших платформ.

Для реалізації графічної частини програмного забезпечення, було вирішено обрати бібліотеку Tkinter, що є кросплатформною графічною бібліотекою для створення користувацьких інтерфейсів мовою програмування Python. Tkinter є зручним інструментом, що надає простий та зрозумілий механізм розробки десктоп-застосунків. Бібліотека підтримує керування розміткою інтерфейсу, обробку подій та базове налаштування вигляду елементів, що дозволяє швидко створювати функціональні та стабільні GUI-рішення [23]. Серед інших графічних бібліотек Tkinter має такі переваги:

1. Забезпечує вбудований та простий у використанні інструментарій для створення графічного інтерфейсу в Python.

2. Містить набір основних віджетів (кнопки, мітки, поля введення, меню), достатній для реалізації інтерактивних настільних застосунків.

3. Не потребує встановлення сторонніх GUI-фреймворків, оскільки входить до стандартного дистрибутиву Python.

4. Добре підходить для розробки настільних програм, таких як калькулятори, форми введення даних та інформаційні панелі.

5. Підтримує подієво-орієнтовану модель програмування, що дозволяє створювати адаптивні та зручні інтерфейси користувача.

Окрім Python та Tkinter було обрано також середовище розробки Visual Studio Code, яке є сучасним редактором коду з відкритим вихідним кодом. Завдяки поєднанню високої продуктивності, гнучкості та розширюваності, Visual Studio Code широко застосовується у професійному середовищі розробників, що робить його влучним вибором для реалізації програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Не менш важливо й те, що Visual Studio Code забезпечує повноцінну підтримку мови Python, зокрема засобами підсвічування синтаксису, автоматичного доповнення коду, перевірку на наявність помилок та налагодження програм. Крім того, можливість розширення функціоналу за допомогою плагінів дозволяє адаптувати середовище під конкретні потреби застосунку, зокрема для роботи з математичними бібліотеками, тестуванням та візуалізацією результатів, що є дуже важливим фактором під час реалізації програмних рішень в області регресійного моделювання.

3.4 Результати розробки програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

В результаті розробки було створено функціональне програмне забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate «JHQualityMeter». Код програмного забезпечення «JHQualityMeter» наведено в Додатку Б. На рисунку 3.9 проілюстровано головну сторінку застосунку «JHQualityMeter».

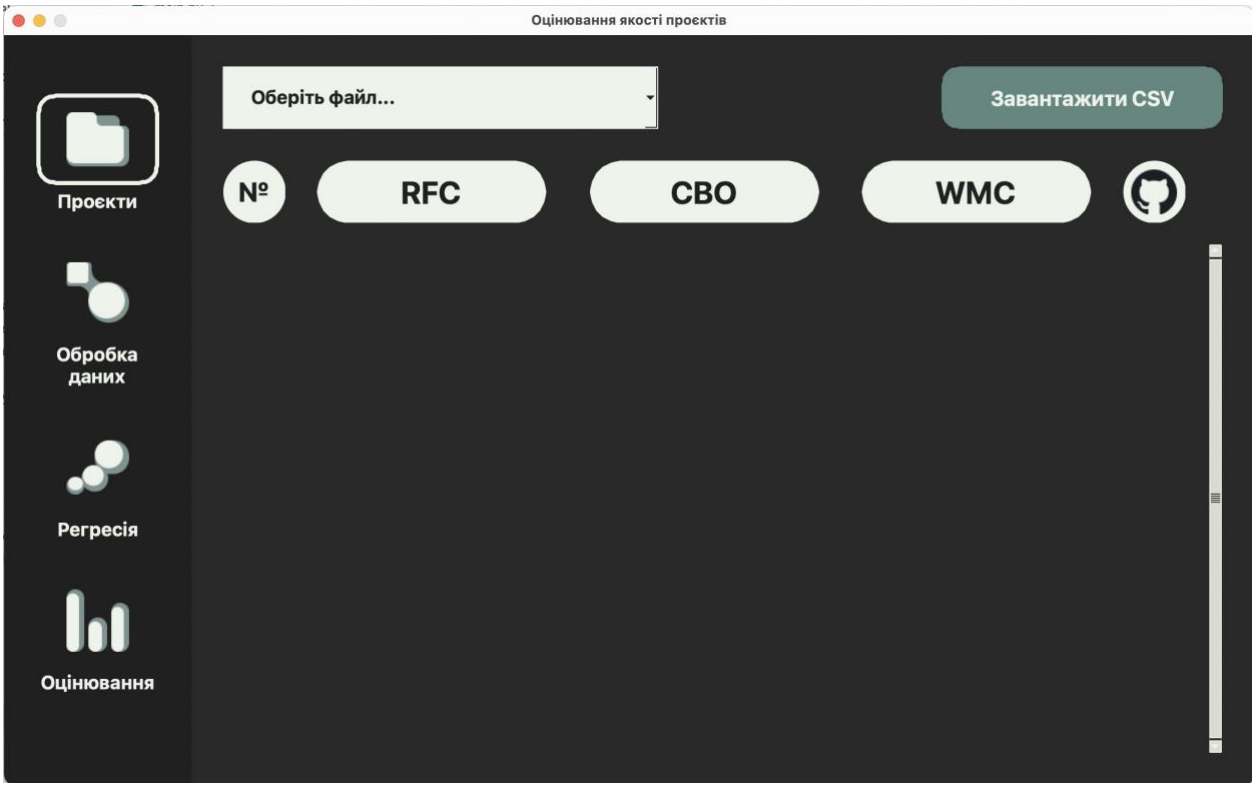


Рисунок 3.9 – Головна сторінка застосунку «JHQualityMeter»

На головній сторінці можна побачити кнопку «Завантажити CSV», через натискання якої користувач має можливість завантажити вхідні дані застосунків, а також бічну панель з компонентами, що мають таке функціональне призначення :

– Проєкти: завантаження та перегляд вхідних даних;

– Обробка даних: перевірка вхідних даних за тестом Мардіа, одновимірна нормалізація методом Бокса-Кокса для кожного параметру, очищення вибірки нормалізованих даних від викидів;

– Регресія: вибір моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO), перегляд побудованих на базі вибраної моделі лінійного та нелінійного регресійних рівнянь, а також відповідних коефіцієнтів та показників якості;

– Оцінювання: вибір моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO), перегляд розкривного списку завантажених застосунків, можливість вводу метрик RFC, CBO та WMC для оцінювання якості стороннього застосунку, побудованого на базі вибраної моделі нелінійного

регресійного рівняння, перегляд значення нелінійної регресії та оцінки якості застосунку, діапазони значень метрик із завантажених проєктів та перегляд в окремому вікні побудованих довірчих інтервалів та інтервалів прогнозування.

На рисунку 3.10 наведено вкладку «Проєкти», де після завантаження вхідних даних, у лівому верхньому куті відображається назва файлу, а нижче наведено таблицю з відповідними значеннями метрик та посиланнями на застосунки з яких метрики було отримано.

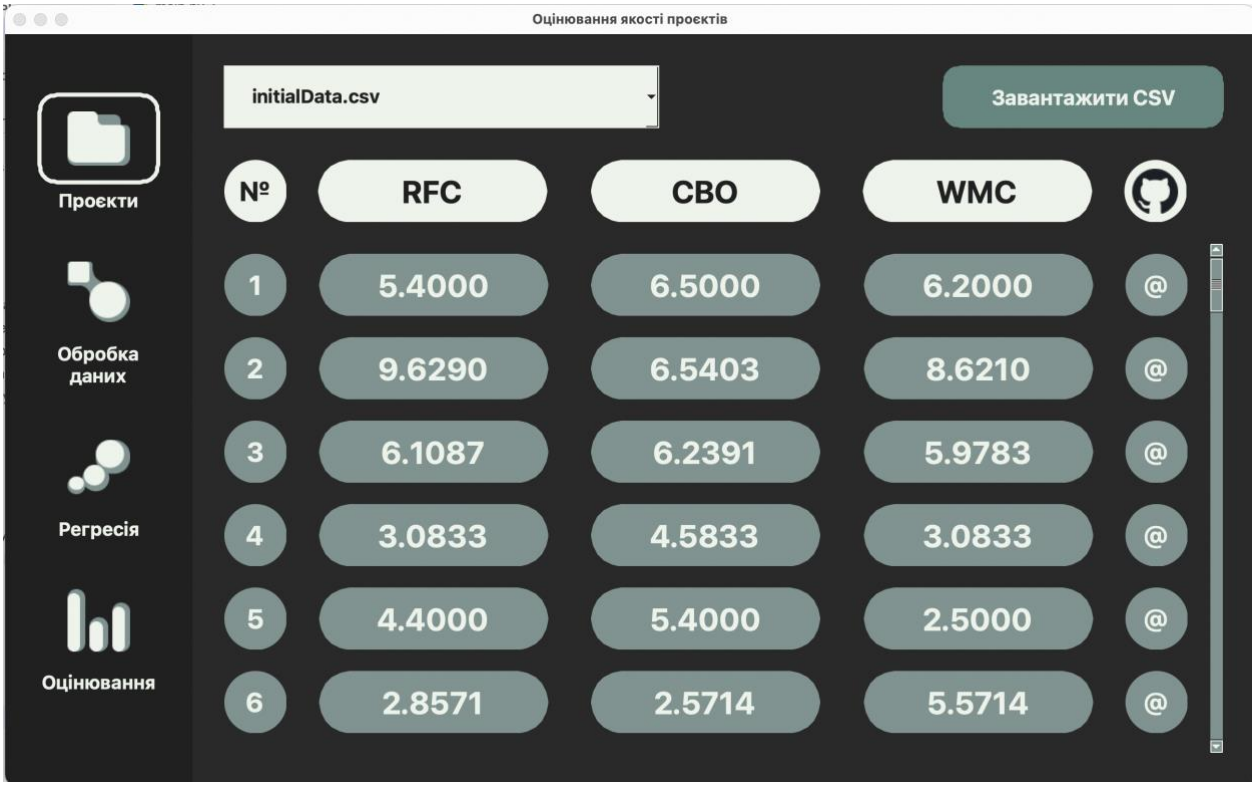


Рисунок 3.10 – Вкладка «Проєкти» застосунку «JHQualityMeter»

На вкладці «Обробка даних» користувач може перевірити нормальність розподілу даних, виконати нормалізацію та очищення даних, що проілюстровано на рисунку 3.11. В ході нормалізації даних методом Бокса-Кокса було отримано такі оптимальні значення лямбд змінних RFC, CBO, WMC: 0,2485, 1,0492, 0,0370 та не було виявлено викидів, про що користувача повідомляється в інтерфейсі. Окрім цього, якщо повторно перевірити дані на тест Мардіа, статуси тестів асиметрії та ексцесу оновляться, оскільки розраховуватимуться вже для очищених від викидів даних.

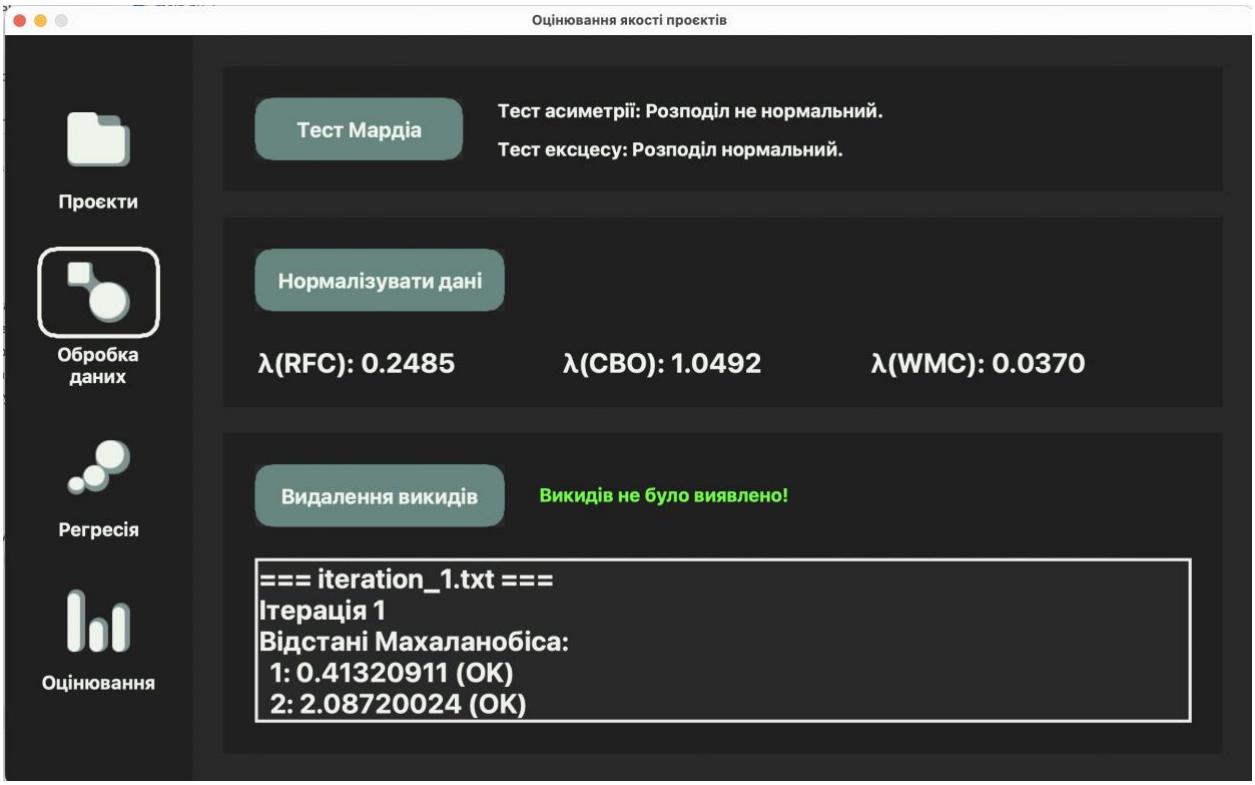


Рисунок 3.11 – Вкладка «Обробка даних» застосунку «JHQualityMeter»

На рисунках 3.12-3.14 проілюстровано вкладку «Регресія», де відображені результати побудови лінійних та нелінійних регресійних рівнянь для кожної з моделей, коефіцієнти b_0, b_1 та b_2, а також відповідні критерії якості моделей: R^2, $MMRE$ та $PRED(0.25)$.

Результати для моделі RFC(CBO, WMC):

$b_0 = -0{,}2599, b_1 = 0{,}1469, b_2 = 0{,}8532.$

$R^2 = 0{,}4772, MMRE = 0{,}2777, PRED(0.25) = 0{,}5571.$

Результати для моделі CBO(WMC, RFC):

$b_0 = 4{,}1336, b_1 = -0{,}7971, b_2 = 1{,}1138.$

$R^2 = 0{,}1682, MMRE = 0{,}2011, PRED(0.25) = 0{,}7571.$

Результати для моделі WMC(RFC, CBO):

$b_0 = 1{,}1486, b_1 = 0{,}4837, b_2 = -0{,}0596.$

$R^2 = 0{,}4090, MMRE = 0{,}3085, PRED(0.25) = 0{,}5714.$

Рисунок 3.12 – Вкладка «Регресія» з обраною моделлю RFC(CBO, WMC) застосунку «JHQualityMeter»

Рисунок 3.13 – Вкладка «Регресія» з обраною моделлю CBO(WMC, RFC) застосунку «JHQualityMeter»

Рисунок 3.14 – Вкладка «Регресія» з обраною моделлю WMC(RFC, CBO) застосунку «JHQualityMeter»

На вкладці «Оцінювання», яка проілюстрована на рисунку 3.15 користувач може ввести в текстові поля значення метрик RFC, CBO, WMC або обрати параметри завантаженого застосунку з вибірки. Модель виконується в діапазонах метрик RFC[1,6000; 13,6222], CBO[2,5714; 9,0952] та WMC[2,1905; 16,2500]. Якщо ввести в текстові поля значення 3, 2 та 7, то побачимо розрахований результат передбачуваного $\hat{Y}$ за моделлю RFC(CBO, WMC), який становить 3,8776, а також прогноз якості, визначений як «високий».

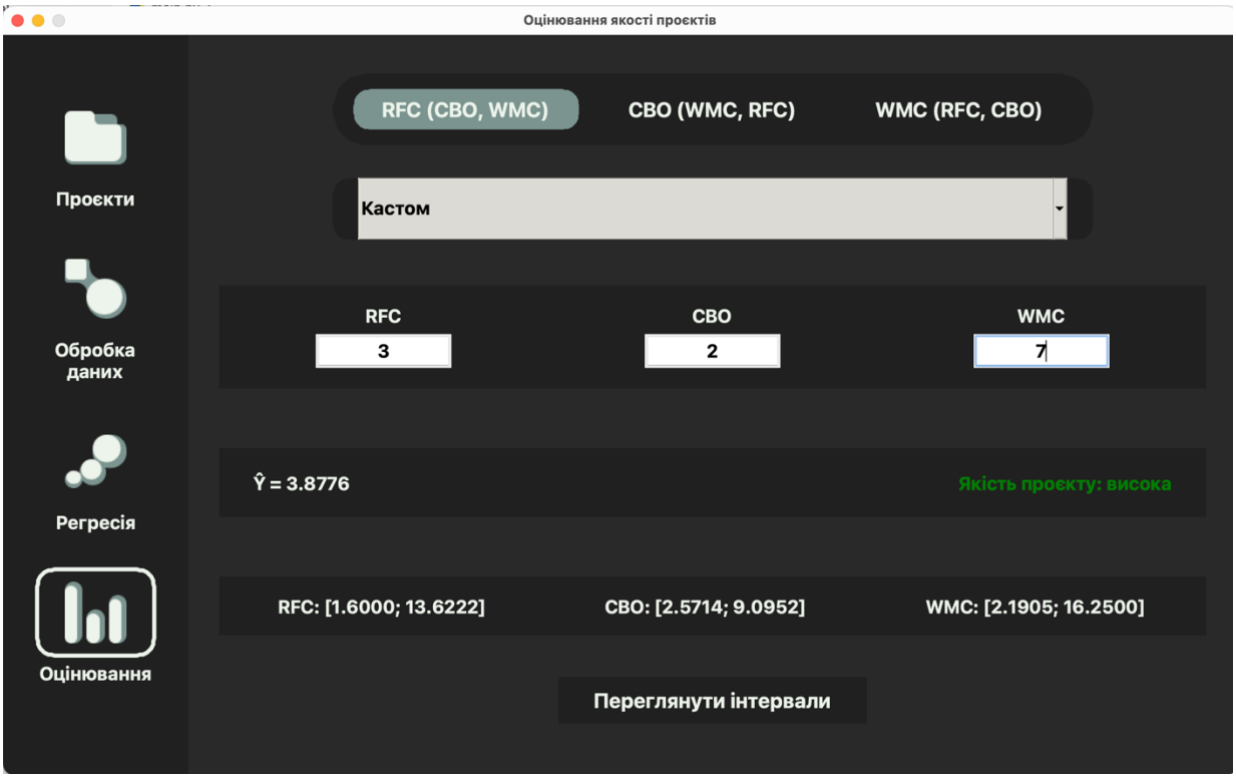
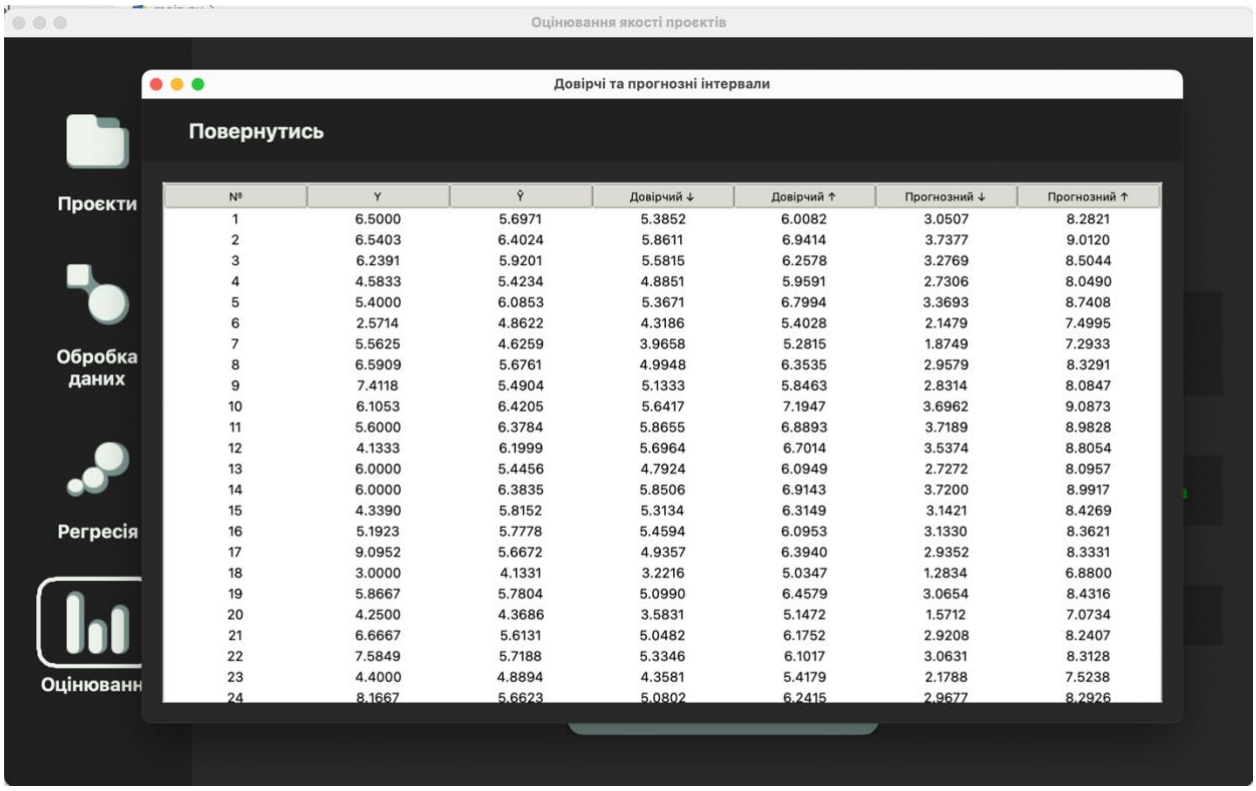


Рисунок 3.15 – Вкладка «Оцінювання» з обраною моделлю RFC(CBO, WMC) застосунку «JHQualityMeter»

Нижче діапазонів значень метрик загальної вибірки знаходиться кнопка «Переглянути інтервали», які будуть відрізнятись в залежності від обраної моделі. На рисунку 3.16 наведено окреме вікно з розрахованими довірчими та прогнозними інтервалами для моделі CBO(WMC, RFC).



№	Y	Ŷ	Довірчий ↓	Довірчий ↑	Прогнозний ↓	Прогнозний ↑
1	6.5000	5.6971	5.3852	6.0082	3.0507	8.2821
2	6.5403	6.4024	5.8611	6.9414	3.7377	9.0120
3	6.2391	5.9201	5.5815	6.2578	3.2769	8.5044
4	4.5833	5.4234	4.8851	5.9591	2.7306	8.0490
5	5.4000	6.0853	5.3671	6.7994	3.3693	8.7408
6	2.5714	4.8622	4.3186	5.4028	2.1479	7.4995
7	5.5625	4.6259	3.9658	5.2815	1.8749	7.2933
8	6.5909	5.6761	4.9948	6.3535	2.9579	8.3291
9	7.4118	5.4904	5.1333	5.8463	2.8314	8.0847
10	6.1053	6.4205	5.6417	7.1947	3.6962	9.0873
11	5.6000	6.3784	5.8655	6.8893	3.7189	8.9828
12	4.1333	6.1999	5.6964	6.7014	3.5374	8.8054
13	6.0000	5.4456	4.7924	6.0949	2.7272	8.0957
14	6.0000	6.3835	5.8506	6.9143	3.7200	8.9917
15	4.3390	5.8152	5.3134	6.3149	3.1421	8.4269
16	5.1923	5.7778	5.4594	6.0953	3.1330	8.3621
17	9.0952	5.6672	4.9357	6.3940	2.9352	8.3331
18	3.0000	4.1331	3.2216	5.0347	1.2834	6.8800
19	5.8667	5.7804	5.0990	6.4579	3.0654	8.4316
20	4.2500	4.3686	3.5831	5.1472	1.5712	7.0734
21	6.6667	5.6131	5.0482	6.1752	2.9208	8.2407
22	7.5849	5.7188	5.3346	6.1017	3.0631	8.3128
23	4.4000	4.8894	4.3581	5.4179	2.1788	7.5238
24	8.1667	5.6623	5.0802	6.2415	2.9677	8.2926

Рисунок 3.16 – Вікно «Довірчі та прогнозні інтервали» з обраною моделлю CBO(WMC, RFC) застосунку «JHQualityMeter»

Опис та інструкція користувача для програмного забезпечення «JHQualityMeter» наведено у Додатку В та у Додатку Г.

3.5 Випробування програмного забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate

В ході випробування програмного забезпечення «JHQualityMeter» для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, була виконана перевірка вимог до функціональних характеристик:

– Завантаження метрик з файлу CSV;
– Перевірка нормальності розподілу вхідних даних тестом Мардіа;
– Нормалізація даних методом Бокса-Кокса;

– Виявлення та видалення викидів;

– Побудова лінійної та нелінійної регресійних моделей;

– Побудова довірчих інтервалів та інтервалів прогнозування;

– Оцінювання якості застосунків за моделями RFC(CBO, WMC), CBO(WMC, RFC), WMC(RFC, CBO).

В таблиці 3.1 наведено випробування програмного забезпечення «JHQualityMeter»

Таблиця 3.1 – Випробування програмного забезпечення «JHQualityMeter»

№	Дія	Очікуваний результат	Результат
1	Завантаження метрик з файлу CSV	Дані успішно завантажені та відображені у вкладці «Проєкти»	Успішно
2	Перевірка нормальності розподілу вхідних даних тестом Мардіа	Дані успішно перевіряються на нормальний розподіл, відображаються тестові статистики асиметрії та ексцесу на вкладці «Обробка даних»	Успішно
3	Нормалізація даних методом Бокса-Кокса	Дані успішно нормалізуються та відображаються у вкладці «Проєкти», відображаються оптимальні лямбди для нормалізованих даних на вкладці «Обробка даних»	Успішно
4	Виявлення та видалення викидів	Викиди успішно виявляються та видаляються на вкладці «Проєкти»	Успішно
5	Побудова лінійної та нелінійної регресійних моделей	Лінійна та нелінійна регресійна модель успішно будується та відображається на вкладці «Регресія»	Успішно
6	Побудова довірчих інтервалів та інтервалів прогнозування	Довірчі інтервали та інтервали прогнозування успішно будуються у вікні «Довірчі та прогнозні інтервали»	Успішно
7	Оцінювання якості застосунків за моделями RFC(CBO, WMC), CBO(WMC, RFC), WMC(RFC, CBO)	Оцінки якості застосунків успішно прогнозуються для моделей RFC(CBO, WMC), CBO(WMC, RFC), WMC(RFC, CBO) на вкладці «Оцінювання»	Успішно

В ході випробувань програмного забезпечення «JHQualityMeter» всі функціональні вимоги були успішно випробувані. Метрики справно завантажувались з файлу CSV, безпомилково перевірялись тестом Мардіа та

нормалізувались методом Бокса-Кокса, викиди виявлялись та успішно видалялись, регресійні моделі та інтервали будувались злагоджено, оцінювання якості застосунків здійснювалось належним чином.

Отже, програмне забезпечення «JHQualityMeter» вдало пройшло всі функціональні тести та може впевнено використовуватись для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Програма і методика випробувань програмного забезпечення «JHQualityMeter» наведена в Додатку Д.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ HIBERNATE

4.1 Аналіз актуальності та ринкових перспектив

Програмне забезпечення, призначене для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate є актуальним та затребуваним в сучасних умовах розвитку ринку інформаційних технологій. Активне застосування Java в корпоративних інформаційних системах, а також поширеність Hibernate як одного з провідних ORM-фреймворків зумовлюють потребу в спеціалізованих інструментах для достовірного аналізу якості застосунків.

Запропоноване програмне забезпечення орієнтоване насамперед на розробників програмних продуктів. Використання автоматизованих засобів аналізу дає змогу своєчасно виявляти потенційні недоліки структури коду, оцінювати рівень його підтримуваності та відповідність сучасним стандартам якості. Очікується, що розроблене програмне забезпечення матиме практичну цінність та перспективи впровадження завдяки поєднанню актуальної предметної області, орієнтації на застосунки з відкритим кодом та можливості комплексного оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

4.2 Розрахунок витрат на розробку

Серед основних статей витрат під час визначення вартості розробки проєкту було враховано: основну та додаткову заробітну плату, витрати на допоміжні матеріали, а також загальногосподарські витрати та витрати на електроенергію.

В таблиці 4.1 наведено витрати на допоміжні матеріали.

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума, грн
Папір (5 пачок)	750
Заправлення картриджа до принтера	450
Флеш-карта (2 шт.)	360
Література	740
Непередбачені витрати	450
Разом	2750

Витрати на розробку програмного забезпечення «JHQualityMeter» наведено в таблиці 4.2.

Пункти витрат	Сума, грн
Основна заробітна плата розробника	65000
Додаткова заробітна плата (премії тощо)	4500
Відрахування на соціальні заходи (22%)	15290
Загальногосподарські витрати	2300
Витрати на допоміжні матеріали	1700
Витрати на електроенергію	4260
Разом	93050

4.3 Прогнозовані доходи та оцінка економічної ефективності

Програмне забезпечення буде реалізовано за ліцензійною моделлю з ціною 400 грн за одну ліцензію. Прогнозований обсяг продажів становить 250 ліцензій на перший рік. Очікуваний дохід: 400 × 250 = 100 000 грн.

Чиста приведена вартість (NPV) [24]:

$$NPV = \sum_{t=1}^{T} \frac{R_t}{(1+r)^t} - C, \tag{4.1}$$

де R_t – доходи за період t, r – ставка дисконтування (10%), C – витрати.

$$NPV = \frac{100\,000}{(1+0{,}1)^1} - 93\,050 = 90\,909 - 93\,050 = -2141 \text{ грн.}$$

Невеликий негативний показник пояснюється високими стартовими витратами.

Рентабельність інвестицій (ROI) [25]:

$$ROI = \frac{\text{Дохід} - \text{Витрати}}{\text{Витрати}} \times 100\%. \tag{4.2}$$

$$ROI = \frac{100000 - 93050}{93050} \times 100\% = 7{,}47\%.$$

Період окупності:

$$\text{Період окупності} = \frac{\text{Витрати}}{\text{Щомісячний дохід}}. \tag{4.3}$$

Щомісячний дохід (рівномірний розподіл): $100\,000/12 = 8333$ грн.

Період окупності $= 93050/8333 = 11.2$ місяців.

Результати економічного аналізу підтверджують доцільність реалізації проєкту. Незначне від’ємне значення показника NPV може бути компенсоване за рахунок зростання обсягів продажів у подальші періоди. Програмний продукт має потенціал окупити понесені витрати протягом 11 місяців та в подальшому забезпечувати стабільний прибуток.

5 ОХОРОНА ПРАЦІ

5.1 Основні законодавчі та нормативні документи, які регламентують охорону праці в Україні

Відповідно до Закону України «Про охорону праці» [26], охорона праці є комплексом взаємопов'язаних правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на забезпечення збереження життя, здоров'я та працездатності людини в процесі здійснення трудової діяльності.

Система охорони праці включає три ключові компоненти: норми трудового законодавства, що регламентують правові аспекти безпеки праці; заходи виробничої санітарії, гігієни праці та техніки безпеки; а також комплекс рішень, пов'язаних із протипожежним захистом та електробезпекою.

Основною метою охорони праці є створення безпечних, нешкідливих та комфортних умов праці, що досягається шляхом вирішення низки складних завдань. До них належать:

– проєктування підприємств, технологічних процесів та конструювання обладнання з обов'язковим урахуванням вимог охорони праці;

– визначення оптимальних співвідношень між факторами виробничого середовища з метою мінімізації їх негативного впливу на здоров'я працівників;

– встановлення та законодавче закріплення допустимих норм небезпечних та шкідливих факторів із подальшим систематичним контролем за їх дотриманням.

Важливим завданням також є розробка та впровадження комплексу заходів, спрямованих на поліпшення умов праці та підвищення її безпеки, базуючись на використанні сучасних досягнень науки та техніки. Особлива

увага приділяється застосуванню ефективних засобів індивідуального та колективного захисту працівників, а також організаційних заходів, що дозволяють нейтралізувати або знизити вплив несприятливих факторів виробничого середовища. Окрім цього, здійснюється розробка та використання методів оцінювання ефективності запланованих та реалізованих заходів з охорони праці.

Дотримання власником підприємства вимог і нормативів правової бази у сфері охорони праці, зокрема положень Конституції України, законів України «Про охорону праці», «Про охорону здоров'я», «Про пожежну безпеку», «Про використання ядерної енергії та радіаційний захист», «Про забезпечення санітарного та епідеміологічного добробуту населення», а також Кодексу законів про працю України, є гарантією забезпечення безпечних умов праці для найманих працівників. Зазначені нормативно-правові акти поширюються на всі підприємства, установи та організації незалежно від форм власності та видів діяльності, а також на всіх осіб, залучених до трудової діяльності.

5.2 Структура управління питаннями охорони праці на підприємстві

Організація діяльності у сфері охорони праці полягає у формуванні такої системи управління, яка найбільш повно відповідає завданню створення безпечних та комфортних умов праці для робітників. Ефективна структура управління охороною праці забезпечує узгодженість дій усіх підрозділів підприємства та сприяє досягненню встановлених вимог безпеки.

Об'єкт управління охороною праці розповсюджується на дії функціональних служб та структурних підрозділів підприємства, спрямованих на забезпечення належних умов безпеки та збереження здоров'я працівників у межах робочих місць, виробничих дільниць, цехів та підприємства в цілому.

Організація та координація робіт у галузі охорони праці передбачає створення відповідних органів управління, визначення обов'язків та відповідальності посадових осіб, а також встановлення порядку їх взаємодії під час прийняття та реалізації управлінських рішень. Загальне управління охороною праці на підприємстві здійснює власник підприємства (роботодавець), а в межах структурних підрозділів – керівники відповідних підрозділів.

Права, обов'язки та відповідальність посадових осіб щодо виконання функцій у сфері охорони праці визначаються посадовими інструкціями, форма яких розробляється уповноваженими державними органами. Відповідно до статті 13 Закону України «Про охорону праці» роботодавець зобов'язаний забезпечити створення належних умов праці на кожному робочому місці та в кожному структурному підрозділі відповідно до вимог нормативно-правових актів, а також гарантувати дотримання прав працівників у сфері охорони праці.

З цією метою роботодавець забезпечує функціонування системи управління охороною праці, що включає створення спеціалізованих служб або призначення відповідальних посадових осіб; розробку та впровадження за участю сторін колективного договору комплексу заходів, спрямованих на досягнення нормативних показників з охорони праці; усунення причин нещасних випадків та професійних захворювань, а також виконання профілактичних заходів, визначених за результатами їх розслідування.

Крім того, роботодавець організовує проведення аудиту охорони праці, лабораторних досліджень умов праці, атестації робочих місць на відповідність вимогам нормативних актів, а також оцінювання технічного стану виробничого обладнання. До його обов'язків також належить розробка та затвердження внутрішніх нормативних документів з охорони праці, здійснення постійного контролю за дотриманням працівниками вимог безпеки та організація заходів із популяризації безпечних методів праці й розвитку співпраці з працівниками у сфері охорони праці.

У разі відсутності в чинних нормативно-правових актах вимог, необхідних для забезпечення безпечних умов праці під час виконання окремих видів робіт, роботодавець зобов’язаний самостійно вжити відповідних заходів для гарантування безпеки працівників.

5.3 Аналіз шкідливих і небезпечних факторів при роботі за персональним комп'ютером

Оскільки повне усунення можливості виникнення раптових та небезпечних ситуацій у процесі трудової діяльності є неможливим, кожен працівник зобов’язаний перед початком роботи та після її завершення перевіряти технічну справність обладнання й електричних комунікацій, що використовуються під час виконання посадових обов’язків. Перевірці, зокрема, підлягають системний блок, монітор, клавіатура, комп’ютерна миша, принтер, навушники (за потреби) та інші технічні засоби.

Відповідно до вимог Закону України «Про охорону праці», порядок дій працівників у разі виникнення окремих аварійних ситуацій визначається наступним чином:

– у разі появи горілого запаху після ввімкнення персонального комп’ютера або відчуття дії електричного струму при дотику до його металевих елементів необхідно негайно відключити обладнання від електромережі та повідомити про це безпосереднього керівника;

– при виникненні пожежі слід негайно розпочати її гасіння наявними первинними засобами пожежогасіння та повідомити пожежну службу за телефоном 101, а також начальника добровільної пожежної дружини підприємства. При цьому гасіння електроустановок дозволяється здійснювати лише вуглекислотними вогнегасниками або сухим піском з метою запобігання ураженню електричним струмом;

– у випадку отримання травми необхідно припинити виконання робіт, надати постраждалому першу медичну допомогу, викликати бригаду екстреної медичної допомоги за телефоном 103 та, за потреби, забезпечити транспортування до лікувального закладу;

– порядок надання першої (долікарняної) медичної допомоги при різних видах уражень детально викладено в інструкції № 03-ОП «Про надання першої (долікарняної) медичної допомоги при нещасних випадках», яка вивчається працівниками під час первинного та повторних інструктажів з охорони праці;

– у випадку виникнення інших аварійних ситуацій працівник зобов'язаний негайно припинити роботу та повідомити про це керівника робіт.

Окрім аварійних ситуацій, у процесі трудової діяльності працівники можуть зазнавати впливу різноманітних шкідливих факторів, які згідно із законодавством України визначаються як шкідливі виробничі фактори.

Шкідливими виробничими факторами вважаються фактори виробничого середовища та трудового процесу, вплив яких може спричиняти розвиток професійних захворювань, тимчасове або стійке зниження працездатності, підвищення рівня загальної захворюваності, а також негативно впливати на стан здоров'я працівника та його нащадків.

До основних шкідливих факторів належать: пожежонебезпечність виробничих приміщень, ризик ураження електричним струмом та постійний вплив статичної електрики, електромагнітне випромінювання (у тому числі рентгенівське), іонізація повітря, несприятливі параметри мікроклімату, підвищений рівень шуму, недостатнє або нераціональне освітлення, а також психофізіологічні навантаження.

Психофізіологічні порушення тісно пов'язані з нервово-емоційним станом працівника та найбільш виражені під час тривалої роботи за персональним комп'ютером. Таке напруження часто обумовлюється дефіцитом часу, постійною необхідністю обробки значних обсягів інформації та надто частим прийняттям відповідальних рішень, що з часом може призводити до перевтоми, підвищеної тривожності та емоційного виснаження.

Тривала робота за комп’ютерними моніторами також негативно впливає на загальний стан здоров’я працівника, проявляючись порушеннями сну, зниженням працездатності, погіршенням самопочуття, а також зростанням ризику розвитку серцево-судинних та хронічних захворювань, депресивних станів та патологій опорно-рухового апарату.

Одним із найменш помітних, проте важливих чинників є недостатній рівень освітленості робочого простору. Неправильно організоване освітлення приміщення спричиняє швидке зорове напруження, передчасну втому очей та зниження ефективності праці, що у довгостроковій перспективі може призводити до розвитку професійних захворювань органів зору.

5.4 Розробка заходів щодо зменшення дії шкідливих факторів при роботі за персональним комп'ютером

З метою зниження негативного впливу шкідливих факторів на фізичний та психоемоційний стан працівників, що тривалий час працюють за персональним комп’ютером, необхідно дотримуватися низки профілактичних правил. Зокрема, рекомендується щогодини робити перерву тривалістю близько десяти хвилин для відпочинку органів зору, уникаючи в цей час будь-яких візуальних подразників, таких як екрани моніторів, телевізорів або читання дрібного тексту. Окрім цього, суворо забороняється використовувати телевізор як заміну комп’ютерного монітора.

За можливості слід своєчасно очищувати поверхню екрана від забруднень та пилу, оскільки це зменшує напруження зору. Важливим правилом є також регулярне провітрювання приміщення, де розміщена комп’ютерна техніка – не рідше одного разу на годину, а також щоденне проведення вологого прибирання з метою підтримання належного санітарного стану робочого середовища.

Особливу увагу слід приділяти положенню тіла під час роботи. Ноги мають упевнено спиратися на підлогу або спеціальну підставку, стегна повинні утворювати прямий кут із тулубом, а гомілки – зі стегнами. Спина має бути рівною або з незначним нахилом уперед. Оптимальна відстань від очей до екрана монітора повинна становити не менше 55-60 сантиметрів, при цьому центр екрана слід розташовувати на рівні очей або дещо нижче.

Для збереження зору рекомендується регулярно моргати з інтервалом в 3-5 секунд та щоденно виконувати спеціальні вправи для очей. У разі появи вираженого відчуття втоми або сонливості необхідно зробити коротку перерву й дати органам зору можливість повноцінно відновитися.

Окрім дотримання зазначених правил, важливо забезпечити належні умови організації робочого місця. Зокрема, робочий стілець має бути твердим, а його край не має створювати тиск на судини в ділянці під колінами. Під час сидячої роботи доцільно робити перерви тривалістю 15-20 хвилин кожні 1-2 години. Також рекомендується використовувати лише якісні та безпечні для зору джерела освітлення в робочому приміщенні.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Розробка екологічно чистого програмного забезпечення

Розробка енергоефективного програмного забезпечення є одним з ключових факторів у зменшенні негативного впливу на навколишнє середовище. Позитивним прикладом енергоефективного підходу можна вважати програмне забезпечення для оцінювання якості застосунків з відкритим кодом на Java, що розроблюються з використанням фреймворку Hibernate. Завдяки використанню кросплатформних інструментів розробки, таких як середовище розробки Visual Studio Code та мова програмування Python, вдається забезпечити універсальність та зменшити потребу у використанні ресурсоємних спеціалізованих інструментів. Це дозволяє оптимізувати процес розробки, скоротити час виконання завдань і, відповідно, зменшити енергоспоживання обчислювальних систем.

Крім того, застосування відкритого коду сприяє повторному використанню вже наявних рішень та бібліотек, що знижує необхідність створення нових компонентів «з нуля». Такий підхід не лише прискорює розробку, але й зменшує кількість обчислювальних операцій під час аналізу та тестування програмного забезпечення. У поєднанні з оптимізованими алгоритмами оцінювання якості це дає змогу знизити навантаження на сервери та робочі станції, що позитивно впливає на загальне енергоспоживання.

6.2 Заходи для зменшення екологічного впливу ІТ-програм

Для зменшення негативного впливу інформаційних технологій на довкілля застосовується низка стратегічних підходів. Одним із ключових

таких напрямів є оптимізація програмного коду. Розробка ефективних програмних рішень, які раціонально використовують апаратні ресурси, сприяє зниженню енергоспоживання як в процесі створення програмного забезпечення, так і під час його подальшої експлуатації. Не менш вагому роль відіграє впровадження енергоефективних методів тестування, зокрема автоматизованих систем, що дають змогу скоротити кількість надлишкових перевірок та, як наслідок, зменшити витрати енергії.

Важливим аспектом також є активне використання технологій віртуалізації, які дозволяють зменшити кількість фізичних серверів. Розподіл обчислювальних ресурсів у межах віртуальних середовищ забезпечує зниження енергетичних витрат та скорочення витрат на технічне обслуговування інфраструктури. У контексті сталого розвитку доцільно також звертати увагу на мінімізацію споживання паперових носіїв, переводячи документацію, пов’язану з розробкою та тестуванням програмного забезпечення, в електронний формат, що сприяє зменшенню обсягів відходів і збереженню природних ресурсів.

Окрему роль відіграє впровадження відновлюваних джерел енергії для забезпечення функціонування серверної інфраструктури. Перехід дата-центрів на використання екологічно чистих джерел енергії дає змогу скоротити залежність від традиційних енергоресурсів та знизити рівень викидів вуглекислого газу. У сукупності такі заходи становлять ґрунтовну основу концепції «зелених інформаційних технологій», спрямованої на забезпечення екологічної сталості та збереження природного балансу.

6.3 Життєвий цикл програмного забезпечення та його екологічний вплив

Життєвий цикл програмного забезпечення зазвичай складається з трьох ключових етапів: розробки, експлуатації та утилізації. Важко не відмітити, що кожен з них своїм чином впливає на довкілля. Вже на етапі створення

програмних продуктів відбувається помітне споживання енергії, адже для розробки, тестування та обробки даних активно використовуються комп’ютери та сервери. З цієї причини доцільно впроваджувати енергоощадні підходи з самого початку, зокрема застосовувати сучасне обладнання та, за можливості, використовувати екологічно чисті джерела енергії.

Під час експлуатації програмне забезпечення має бути спроєктоване з урахуванням енергоефективності кінцевих пристроїв – смартфонів, ноутбуків та іншої повсякденної електроніки. Оптимізована робота програм дозволяє зменшити споживання енергії, особливо в тих випадках, коли завдання не потребують постійної високої обчислювальної потужності. Це не лише знижує навантаження на пристрої, але й сприяє їх тривалішому використанню.

На завершальному етапі життєвого циклу особливу увагу слід приділяти належній утилізації застарілого обладнання. Сервери та електронні пристрої, що вийшли з експлуатації, мають підлягати правильному перероблюванню, задля мінімізації обсягів електронних відходів. Повторне використання окремих компонентів дає змогу зменшити негативний вплив на навколишнє середовище та зробити використання інформаційних технологій більш екологічно відповідальним.

6.4 Впровадження екологічних стандартів у розробку програмного забезпечення

Однією з ключових складових сталого розвитку інформаційних технологій є впровадження міжнародно визнаних екологічних стандартів. Зокрема, стандарт ISO 14001 [27], який регламентує вимоги до систем екологічного менеджменту, сприяє формуванню екологічно відповідального підходу на всіх етапах життєвого циклу програмного забезпечення. Водночас застосування системи EPEAT [28] дає змогу оцінювати рівень впливу

інформаційно-технічних продуктів на довкілля та стимулює використання більш екологічних рішень.

Концепція «екологічного ІТ» має на меті раціональне використання енергетичних та матеріальних ресурсів, зменшення обсягів викидів вуглекислого газу, обмеження застосування шкідливих речовин та продовження строку експлуатації електронних пристроїв. Дотримання цих принципів у процесі розробки програмного забезпечення дозволяє суттєво знизити негативний вплив інформаційних технологій на навколишнє середовище та сприяє сталому розвитку галузі.

Таким чином, врахування екологічних аспектів під час створення програмного забезпечення є невід’ємним та значущим кроком у напрямі довготривалого розвитку ІТ-сфери. Поєднання енергоефективних рішень, дотримання міжнародних екологічних стандартів та впровадження екологічно безпечних технологій зменшує екологічне навантаження та сприяє збереженню природних ресурсів, що має вирішальне значення для її довгострокового розвитку.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було вдосконалено нелінійні регресійні моделі для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate за рахунок врахування особливостей взаємозв'язків між метриками RFC, CBO та WMC таких застосунків.

Аналіз існуючих підходів показав, що традиційні моделі для оцінювання якості програмного забезпечення з відкритим кодом на Java не враховують всіх особливостей застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate через відмінності в значеннях діапазонів метрик. Через що виникла необхідність у новій, модифікованій моделі для фреймворку Hibernate, що базується на побудові трьох нелінійних регресійних моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO). Для виконання роботи було зібрано та проаналізовано 70 застосунків, де використовується фреймворк Hibernate. Після обробки даних було встановлено нормальність розподілу та відсутність викидів у загальній вибірці, а отже всі вхідні дані, були використанні в нормалізованому вигляді для побудови моделей.

Побудовані регресійні моделі дозволяють достовірно оцінювати якість застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

На основі побудованих регресійних моделей, було також успішно створено програмне забезпечення «JHQualityMeter» для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sellers S. 2025 State of Java Survey & Report. URL: https://securityboulevard.com/2025/01/state-of-java-survey-confirms-majority-of-enterprise-apps-are-built-on-java (дата звернення: 20.10.2025).

2. Bagnulo M., Claise B., Eardley P., Morton A., Akhter A. RFC 8911. URL: https://www.rfc-editor.org/rfc/rfc8911.html (дата звернення: 21.10.2025).

3. Shyam R., Chris F. Metrics suite for object-oriented design. URL: https://www.eso.org/~tcsmgr/oowg-forum/TechMeetings/Articles/OOMetrics.pdf (дата звернення: 21.10.2025).

4. Aanchal S., Kumar. Metrics for Software Components in Object Oriented Environments: A Survey // International Journal of Scientific Research in Computer Science and Engineering. 2013. ISSN 2320-7639. URL: https://isroset.org/pub_paper/IJSRCSE/ISROSET-IJSRCSE-00043.pdf (дата звернення: 21.10.2025).

5. Prykhodko S. Evaluating Quality of Software Systems by the Confidence and Prediction Intervals of Regressions for RFC, CBO and WMC Metrics // WSEAS Transactions on Systems. 2024. Vol. 23. P. 322–330. DOI: https://doi.org/10.37394/23202.2024.23.36 (дата звернення: 22.10.2025).

6. Prykhodko S., Makarova L., Pukhalevych A. Statistical Analysis of the Three-Dimensional Data of Software Metrics RFC, CBO, and WMC that are not Normally Distributed // International Conference on Applied Innovations in IT (ICAIIT). 2025. P. 127-131. DOI: https://doi.org/10.25673/119224 (дата звернення: 22.10.2025).

7. Пухалевич А. В., Коваленко В. В. Нелінійні регресійні моделі для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate // Інформаційні технології: моделі, алгоритми, системи : VI Всеукраїнська науково-практична інтернет-конференція (ITMAS–2025). Миколаїв, 2025. С. 71–73. URL: https://itconf.nuos.edu.ua/2025/proceedings/ (дата звернення: 11.11.2025).

8. McCall J. A., Richards P. K., Walters G. F. Factors in Software Quality: report. Rome (NY), 1977. RADC TR-77-369.

9. Панченко Т., Тузова І., Тузов О., Чумак О., Стародуб В. Аналіз моделей якості програмного забезпечення. URL: https://archive.interconf.center/index.php/2709-4685/article/view/5696 (дата звернення: 22.10.2025).

10. Sevestre P. SQL Injection and How to Prevent It. ULR: https://www.baeldung.com/sql-injection (дата звернення: 19.10.2025).

11. Aniche M. Java code metrics calculator (CK). URL: https://github.com/mauricioaniche/ck (дата звернення: 18.10.2025).

12. Mardia K. V. Measures of multivariate skewness and kurtosis with applications // Biometrika. 1970. Vol. 57. P. 519–530. DOI: https://doi.org/10.1093/biomet/57.3.519 (дата звернення: 22.10.2025).

13. Mahalanobis Distance. URL: https://www.machinelearningplus.com/statistics/mahalanobis-distance/ (дата звернення: 22.10.2025).

14. D., Wulandari S., Sutrisno. Mardia's Skewness and Kurtosis for Assessing Normality Assumption in Multivariate Regression // Enthusiastic: International Journal of Applied Statistics and Data Science. 2021. P. 2–4. URL: https://www.researchgate.net/publication/353213860_Mardia's_Skewness_and_Kurtosis_for_Assessing_Normality_Assumption_in_Multivariate_Regression (дата звернення: 22.10.2025).

15. Normal distribution. URL: https://www.investopedia.com/terms/n/normaldistribution.asp (дата звернення: 25.10.2025).

16. Afifi A. A., Azen S. P. Statistical analysis: a computer-oriented approach. New York; London: Academic Press, 1972.

17. Prykhodko S., Prykhodko N. A Technique for Detecting Software Quality Based on the Confidence and Prediction Intervals of Nonlinear Regression for RFC Metric // 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT). Lviv, 2022. P. 499–502. DOI:

https://doi.org/10.1109/CSIT56902.2022.10000532 (дата звернення: 26.10.2025).

18. Розробка вебсайтів. Technologies. MVC // Brander. URL: https://brander.ua/technologies/mvc (дата звернення: 15.10.2025).

19. Основи UML // KDE Documentation. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html (дата звернення: 15.10.2025).

20. Опис взаємодії за допомогою Sequence diagram // Кафедра комп'ютерних наук. URL: http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/laboratorna-robota-9-diahrama-poslidovnosti.pdf (дата звернення: 17.10.2025).

21. Visual Studio Code. URL: https://code.visualstudio.com (дата звернення: 17.10.2025).

22. Васильченко М. Мова програмування Python для машинного навчання та штучного інтелекту // GoIT. URL: https://goit.global/ua/articles/mova-prohramuvannia-python-dlia-mashynnoho-navchannia-ta-shtuchnoho-intelektu/ (дата звернення: 19.10.2025).

23. Python Tkinter // GeeksforGeeks. URL: https://www.geeksforgeeks.org/python/python-gui-tkinter/ (дата звернення: 19.10.2025).

24. Net Present Value (NPV) // Investopedia. URL: https://www.investopedia.com/terms/n/npv.asp (дата звернення: 12.10.2025).

25. What Is ROI? How to Calculate Return on Investment // TechTarget. URL: https://www.techtarget.com/searchcio/definition/ROI (дата звернення: 12.10.2025).

26. Про охорону праці : Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 01.10.2023. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text (дата звернення: 18.10.2025).

27. ISO 14001:2015 – Environmental management systems // ISO. URL: https://www.iso.org/standard/60857.html (дата звернення: 13.10.2025).

28. EPEAT Registry // EPEAT. URL: https://www.epeat.net/ (дата звернення: 13.10.2025).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JHQualityMeter»

1 Вступ

1.1 Назва програмного забезпечення

Повна назва програмного продукту: Програмне забезпечення «JHQualityMeter» для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

Коротка назва: «JHQualityMeter».

1.2 Сфера застосування

Програмне забезпечення застосовується у сфері розробки застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

2 Підстава для розробки програмного забезпечення

Розробка програмного забезпечення «JHQualityMeter» виконується на підставі наказу на затвердження тем кваліфікаційних робіт магістрів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

3 Призначення розробки

3.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є завантаження та перегляд метрик застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, побудова регресійних

моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO) для оцінювання якості застосунків.

3.2 Експлуатаційне призначення

Експлуатаційним призначенням програмного забезпечення є полегшення та автоматизація процесу оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

4 Технічні вимоги до програми

4.1 Вимоги до функціональних характеристик

4.1.1 Вимоги до переліку виконуваних функцій

Розроблене в ході кваліфікаційної роботи програмне забезпечення має містити наступні функції:

– завантаження файлу формату CSV з метриками RFC, CBO, WMC;

– перегляд завантажених метрик;

– перевірка нормальності розподілу даних тестом Мардіа;

– одновимірна нормалізація даних методом Бокса-Кокса для кожного параметру;

– виявлення та видалення викидів;

– вибір моделі для побудови регресійних рівнянь;

– побудова лінійних регресійних моделей методом найменших квадратів;

– побудова нелінійних регресійних моделей методом зворотного перетворення;

– оцінювання якості побудованих моделей за критеріями точності $R^2, MMRE$ та $PRED(0.25)$;

– побудова довірчих інтервалів та інтервалів прогнозування;

– оцінювання якості застосунків за обраною моделлю RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO).

4.1.2 Вимоги до організації вхідних та вихідних даних

Організація вхідних та вихідних даних програмного забезпечення «JHQualityMeter» має відповідати наявності таких функцій:

Вхідні дані:

– файл формату CSV з URL адресами застосунків та метриками RFC, CBO, WMC;

– параметри RFC, CBO та WMC для оцінювання якості стороннього застосунку на основі побудованих моделей.

Вихідні дані:

– файл з обробленими даними у форматі CSV;

– викиди;

– параметри побудованої лінійної регресії для кожної моделі: RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO);

– параметри нелінійних моделей регресій: RFC(CBO, WMC), CBO(RFC, WMC) та WMC(RFC, CBO);

– показники якості нелінійних регресійних моделей;

– таблиця з довірчими інтервалами та інтервалами прогнозування для кожної з нелінійних моделей;

– оцінка якості кожного застосунку загальної вибірки та опціонально для будь-якого стороннього застосунку на основі побудованих моделей, що класифікуються за шкалою оцінок: «висока», «середня» та «низька».

4.2 Вимоги до надійності

4.2.1 Вимоги до забезпечення надійного функціонування програми

Надійне функціонування програми має бути забезпечене завдяки виконанню сукупності організаційно-технічних заходів, основними з яких є:

– захист від хибних дій користувача (під час завантаження файлу та введення метрик в текстові поля);

– встановлення мови Python не нижче версії 3.13, та бібліотеки Tkinter не нижче версії 8.6.

4.2.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

4.2.3 Вимоги до контролю вхідної та вихідної інформації

Якщо під час завантаження файлу або заповнення форми в поля будуть введені некоректні дані, встановлений валідатор має повідомити про це користувача та надати інструкцію щодо розв'язання помилки.

4.3 Умови експлуатації

4.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації програмного забезпечення мають відповідати умовам експлуатації технічних засобів, що використовуються для роботи з програмним забезпеченням.

4.3.2 Вимоги до чисельності та кваліфікації користувачів

Користувачі програмного забезпечення повинні мати базові навички вміння роботи за комп'ютером.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Вимоги до користувацького обладнання:

Мінімальними системними вимогами комп'ютерного пристрою для роботи з програмним забезпеченням є:

– процесор: Intel Pentium 4 або новіше із підтримкою SSE3;
– оперативна пам'ять: 2 ГБ або більше;
– дисковий простір: 200 МБ;
– пристрої вводу-виводу: маніпулятор типу "миша", клавіатура, монітор з екраном роздільної здатності не менше ніж 1280 на 1024 пікселі.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Вимоги до вихідних кодів і мов програмування

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи Windows (версії не нижче Windows 7) або macOS (версії не нижче 12.7.1).

Microsoft Visual Studio Code буде використовуватись як інтегроване середовище для розробки програми. Вихідні коди програми мають бути реалізовані мовою програмування Python (версії не нижче 3.13) та графічною бібліотекою Tkinter (версії не нижче 8.6).

4.6 Спеціальні вимоги

Програмне забезпечення має взаємодіяти з користувачем через графічний інтерфейс користувача, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

5 Вимоги до програмної документації

Склад програмної документації повинен містити в собі:

– технічне завдання;
– текст програми;
– керівництво користувача;
– опис програми;
– програму та методику випробувань.

6 Техніко-економічні показники

У кваліфікаційній роботі техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Етапи розробки програмного забезпечення наведено в таблиці А.1.

Таблиця А.1 – Етапи розробки

№ п/п	Назва етапів розробки програмного забезпечення	Кінцевий термін виконання
1	Аналіз вимог до ПЗ	30.10.2025
2	Розробка архітектури ПЗ	01.11.2025
3	Розробка проєкту ПЗ	16.11.2025
4	Реалізація ПЗ	26.11.2025
5	Випробування ПЗ	28.11.2025

8 Порядок контролю і прийомки

Контроль здійснюється керівником кваліфікаційної роботи на кожному етапі розробки програмного забезпечення. Приймання здійснюється за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

ДОДАТОК Б – КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JHQualityMeter»

Текст файлу **main.py**

```
import tkinter as tk
from frames.ProjectsFrame import ProjectsFrame
from frames.NormalizationFrame import NormalizationFrame
from frames.RegressionFrame import RegressionFrame
from frames.PredictionFrame import PredictionFrame
from frames.SidebarFrame import SidebarFrame

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.current_file_path = None
        self.selected_file = tk.StringVar()
        self.files_list = []
        self.projects_raw = []
        self.current_data = None
        self.boxcox_lambdas = None
        self.title("Оцінювання якості застосунків")
        self.geometry("1200x720")
        self.configure(bg="#292929")
        self.resizable(False, False)
        self.icons = {}
        self.sidebar_items = []
        self.active_item = None
        self.sidebar = SidebarFrame(self, self)
        self.create_window()

    def create_window(self):
        self.container = tk.Frame(self, bg="#292929")
        self.container.pack(side="left", fill="both", expand=True)
        self.frames = {
            "projects": ProjectsFrame(self.container, self),
            "normalization": NormalizationFrame(self.container, self),
            "regression": RegressionFrame(self.container, self),
            "prediction": PredictionFrame(self.container, self)}
        for frame in self.frames.values():
            frame.place(relx=0, rely=0, relwidth=1, relheight=1)
        self.show_frame("projects")

    def show_frame(self, name):
        frame = self.frames[name]
        frame.tkraise()
        if name == "prediction":
            frame.update_projects_list()

if __name__ == "__main__":
    App().mainloop()
```

Текст файлу **frames/SidebarFrame.py**

```
import tkinter as tk
from model.constants import *
from model.utils import UIUtils
```

```
class SidebarFrame(tk.Frame):
    def __init__(self, parent, app):
        super().__init__(parent, bg=BG_SIDEBAR, width=SIDEBAR_WIDTH)
        self.app = app
        self.pack(side="left", fill="y")
        self.pack_propagate(False)
        self.sidebar_items = []
        self.active_item = None
        self.icons = UIUtils.load_icons(["projects", "normalization",
"regression", "prediction", "github_icon"])
        self.create_items()

    def create_items(self):
        items_container = tk.Frame(self, bg=BG_SIDEBAR)
        items_container.pack(expand=True)
        menu = [
            ("Проєкти", self.icons["projects"], "projects"),
            ("Обробка\nданих", self.icons["normalization"], "normalization"),
            ("Регресія", self.icons["regression"], "regression"),
            ("Оцінювання", self.icons["prediction"], "prediction")
        ]
        for i, (text, icon, frame_name) in enumerate(menu):
            item = self.create_item(items_container, icon, text, frame_name)
            self.sidebar_items.append(item)
            if i == 0:
                self.after(100, item["activate"])

    def create_item(self, parent, icon, text, frame_name):
        item = tk.Frame(parent, bg=BG_SIDEBAR)
        item.pack(expand=True, fill="x", pady=(0, 30))
        canvas = tk.Canvas(item, width=ICON_BOX_W, height=ICON_BOX_H,
            bg=BG_SIDEBAR, highlightthickness=0, cursor="hand2")
        canvas.pack()
        canvas.create_image(ICON_BOX_W // 2, ICON_BOX_H // 2, image=icon)
        label = tk.Label(item, text=text, fg=WHITE,
            bg=BG_SIDEBAR, font=FONT_TEXT, cursor="hand2")
        label.pack()
        state = {"rect": None, "active": False}

        def draw_border():
            if state["rect"] is None:
                pad = ICON_BORDER // 2 + 1
                state["rect"] = UIUtils.draw_rounded_rect(
                    canvas,
                    pad, pad,
                    ICON_BOX_W - pad,
                    ICON_BOX_H - pad,
                    r=ICON_RADIUS,
                    outline=WHITE,
                    width=ICON_BORDER,
                    fill=""
                )
        def remove_border():
            if state["rect"] is not None:
                canvas.delete(state["rect"])
                state["rect"] = None
        def on_enter(event):
            if not state["active"]:
                draw_border()
        def on_leave(event):
            if not state["active"]:
                remove_border()
        def on_click(event):
            for other in self.sidebar_items:
                other["deactivate"]()
            state["active"] = True
```

```
            draw_border()
            self.active_item = state
            self.app.show_frame(frame_name)
        def activate():
            state["active"] = True
            draw_border()
        def deactivate():
            state["active"] = False
            remove_border()
        canvas.bind("<Enter>", on_enter)
        canvas.bind("<Leave>", on_leave)
        canvas.bind("<Button-1>", on_click)
        label.bind("<Button-1>", on_click)

        return {"activate": activate, "deactivate": deactivate}
```

Текст файлу **frames/ProjectsFrame.py**

```
import tkinter as tk
import webbrowser
from tkinter import ttk
from frames.BaseFrame import BaseFrame
from model.constants import *
from model.utils import UIUtils
from model.data_loader import load_data

class ProjectsFrame(BaseFrame):
    def __init__(self, parent, app):
        super().__init__(parent, app)
        self.current_file_path = None
        self.selected_file = tk.StringVar()
        self.files_list = []
        self.icons = UIUtils.load_icons(["github_icon"])
        self.create_file_bar()
        self.create_table(self.main)

    def create_file_bar(self):
        bar = tk.Frame(self.main, bg=BG_MAIN, height=60)
        bar.pack(fill="x", pady=(0, 30))
        bar.pack_propagate(False)

        style = ttk.Style()
        style.theme_use("clam")
        style.configure("File.TCombobox", fieldbackground=WHITE,
background=WHITE,
            foreground=WHITE, bordercolor=WHITE, lightcolor=WHITE,
darkcolor=BG_MAIN,
            arrowcolor=BG_SIDEBAR, padding=(25, 8, 10, 8), relief="raised"
        )
        style.map("File.TCombobox", fieldbackground=[("readonly", WHITE)],
            background=[("readonly", WHITE)], foreground=[("readonly",
BG_SIDEBAR)]
        )
        combo_canvas = UIUtils.create_rounded_button(bar, "Оберіть файл...",
418, 60, BG_MAIN,
            command=lambda: None, radius=ICON_RADIUS
        )
        combo_canvas.pack(side="left")
        self.combo = ttk.Combobox(
            combo_canvas,
            textvariable=self.selected_file,
            font=FONT_TEXT,
            state="readonly",
            style="File.TCombobox"
```

```
        )
        self.combo.pack(fill="both", expand=True)
        self.selected_file.set("Оберіть файл...")
        self.combo.bind("<<ComboboxSelected>>", self.on_file_selected)
        btn = UIUtils.create_rounded_button(
            bar, "Завантажити CSV",
            270, 60, BUTTON_COLOR,
            command=lambda: (
                UIUtils.add_file(self, load_data),
                self.notify_prediction_frame()
            ),
            radius=ICON_RADIUS
        )
        btn.pack(side="right")

    def on_file_selected(self, event=None):
        UIUtils.on_file_select(self, load_data)
        self.notify_prediction_frame()

    def create_table(self, parent):
        table_container = tk.Frame(parent, bg=BG_MAIN)
        table_container.pack(fill="both", expand=True)
        header = tk.Frame(table_container, bg=BG_MAIN)
        header.pack(fill="x", padx=(0, 15), pady=(0, 2 * CELL_PADY))
        col_weights = [2, 1, 2, 2, 2, 2, 2, 1, 2]
        for col, w in enumerate(col_weights):
            header.grid_columnconfigure(col, weight=w)
        UIUtils.rounded_circle_cell(header, "№", bg_color=WHITE,
fg=BG_SIDEBAR)\
            .grid(row=0, column=0, sticky="w")
        UIUtils.header_cell(header, "RFC", LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, WHITE, BG_SIDEBAR)\
            .grid(row=0, column=2, sticky="ew")
        UIUtils.header_cell(header, "CBO", LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, WHITE, BG_SIDEBAR)\
            .grid(row=0, column=4, sticky="ew")
        UIUtils.header_cell(header, "WMC", LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, WHITE, BG_SIDEBAR)\
            .grid(row=0, column=6, sticky="ew")
        git_icon = UIUtils.rounded_circle_cell(header, "", bg_color=WHITE)
        git_icon.create_image(CELL_HEIGHT // 2, CELL_HEIGHT // 2,
image=self.icons["github_icon"])
        git_icon.grid(row=0, column=8, sticky="w")
        body_container = tk.Frame(table_container, bg=BG_MAIN)
        body_container.pack(fill="both", expand=True)
        self.canvas = tk.Canvas(body_container, bg=BG_MAIN,
highlightthickness=0)
        self.canvas.pack(side="left", fill="both", expand=True)
        style = ttk.Style()
        style.configure("Vertical.TScrollbar", background=ACCENT,
troughcolor=ACCENT,
            bordercolor=BG_MAIN, arrowcolor=WHITE, lightcolor=WHITE,
            darkcolor=WHITE, relief="raised"
        )
        scrollbar = ttk.Scrollbar(body_container, orient="vertical",
            command=self.canvas.yview, style="Vertical.TScrollbar"
        )
        scrollbar.pack(side="right", fill="y")
        self.canvas.configure(yscrollcommand=scrollbar.set)

        self.rows_frame = tk.Frame(self.canvas, bg=BG_MAIN)
        self.canvas_window = self.canvas.create_window((0, 0),
window=self.rows_frame, anchor="nw")
        for col, w in enumerate(col_weights):
            self.rows_frame.grid_columnconfigure(col, weight=w)
```

```
        def _on_frame_configure(event):
            self.canvas.configure(scrollregion=self.canvas.bbox("all"))
            self.canvas.itemconfig(
                self.canvas_window,
                width=self.canvas.winfo_width()
            )
        self.rows_frame.bind("<Configure>", _on_frame_configure)

        def _on_mousewheel(event):
            pos = self.canvas.yview()[0]
            pos -= event.delta * 0.001
            pos = max(0, min(1, pos))
            self.canvas.yview_moveto(pos)
        self.canvas.bind("<Enter>",
            lambda e: self.canvas.bind_all("<MouseWheel>", _on_mousewheel))
        self.canvas.bind("<Leave>",
            lambda e: self.canvas.unbind_all("<MouseWheel>"))

    def data_row(self, parent, row_index, idx, rfc, cbo, wmc, github_url):
        UIUtils.rounded_circle_cell(parent, idx, bg_color=ACCENT, fg=WHITE)\
            .grid(row=row_index, column=0, sticky="w", pady=CELL_PADY)
        UIUtils.header_cell(parent, rfc, LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, ACCENT, WHITE)\
            .grid(row=row_index, column=2, sticky="ew", pady=CELL_PADY)
        UIUtils.header_cell(parent, cbo, LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, ACCENT, WHITE)\
            .grid(row=row_index, column=4, sticky="ew", pady=CELL_PADY)
        UIUtils.header_cell(parent, wmc, LONG_CELL_WIDTH, CELL_HEIGHT,
                            CIRCLE_RADIUS, ACCENT, WHITE)\
            .grid(row=row_index, column=6, sticky="ew", pady=CELL_PADY)
        UIUtils.rounded_circle_cell(
            parent, "@",
            click_command=lambda: webbrowser.open(github_url)
        ).grid(row=row_index, column=8, sticky="w", pady=CELL_PADY)

    def notify_prediction_frame(self):
        prediction = self.app.frames.get("prediction")
        if prediction:
            prediction.update_projects_list()
            prediction.update_ranges_from_file()
```

Текст файлу **frames/NormalizationFrame.py**

```
import os
import csv
import tkinter as tk
from tkinter import messagebox
from frames.BaseFrame import BaseFrame
from model.constants import *
from model.utils import UIUtils
from model.data_loader import load_data
from model.mardia_test import run_mardia_test
from model.normalization import normalize_file
from model.outlier_removal import remove_outliers_file

class NormalizationFrame(BaseFrame):
    def __init__(self, parent, app):
        super().__init__(parent, app)
        self.app = app
        self.current_file_path = None
        self.test_results = None
        self.normalized_file_path = None
        self.cleaned_file_path = None
        self.main.grid_columnconfigure(0, weight=1)
```

```
        self.main.grid_rowconfigure(0, weight=0)
        self.main.grid_rowconfigure(1, weight=1)
        self.main.grid_rowconfigure(2, weight=1)
        self.main.grid_rowconfigure(3, weight=1)
        self.main.grid_rowconfigure(4, weight=12)
        self.block_mardia = tk.Frame(self.main, bg=BG_SIDEBAR, height=150)
        self.block_mardia.grid(row=0, column=0, sticky="ew")
        self.block_mardia.grid_propagate(False)

        self.mardia_btn = UIUtils.create_rounded_button(self.block_mardia,
            "Тест Мардіа", 200, 60, BUTTON_COLOR, self.run_mardia,
radius=ICON_RADIUS
        )
        self.mardia_btn.pack(side="left", padx=(30, 30), pady=30)
        self.results_frame = tk.Frame(self.block_mardia, bg=BG_SIDEBAR)
        self.results_frame.pack(side="left", padx=0, pady=20, anchor="w")

        self.asymmetry_label = tk.Label(self.results_frame, text="",
font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR)
        self.asymmetry_label.pack(anchor="w", pady=5)

        self.kurtosis_label = tk.Label(self.results_frame, text="",
font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR)
        self.kurtosis_label.pack(anchor="w", pady=5)

        self.block_normalization = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.block_normalization.grid(row=2, column=0, sticky="nsew")
        self.block_normalization.grid_propagate(False)

        self.normalize_btn =
UIUtils.create_rounded_button(self.block_normalization, "Нормалізувати дані",
            240, 60, BUTTON_COLOR, self.run_normalization, radius=ICON_RADIUS
        )
        self.normalize_btn.pack(side="top", anchor="nw", padx=30, pady=30)

        self.lambdas_frame = tk.Frame(self.block_normalization,
bg=BG_SIDEBAR)
        self.lambdas_frame.pack(fill="both", padx=30, pady=0)
        self.lambda_vars = {}

        for i in range(len(NUMERIC_COLS)):
            self.lambdas_frame.grid_columnconfigure(i, weight=1)

        for col_idx, col in enumerate(NUMERIC_COLS):
            item = tk.Frame(self.lambdas_frame, bg=BG_SIDEBAR)
            item.grid(row=0, column=col_idx, sticky="nsew")

            tk.Label(item, text=f"λ({col}):", font=FONT_CELL, fg=WHITE,
bg=BG_SIDEBAR).pack(side="left")
            var = tk.StringVar(value="—")
            self.lambda_vars[col] = var

            tk.Label(item, textvariable=var, font=FONT_CELL, fg=WHITE,
bg=BG_SIDEBAR).pack(side="left", padx=(0, 0))

        self.block_clean = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.block_clean.grid(row=4, column=0, sticky="nsew")
        self.block_clean.grid_propagate(False)
        self.block_clean.grid_columnconfigure(0, weight=0)
        self.block_clean.grid_columnconfigure(1, weight=0, minsize=0)
        self.block_clean.grid_columnconfigure(2, weight=1)

        self.clean_btn = UIUtils.create_rounded_button(self.block_clean,
"Видалення викидів",
            240, 60, BUTTON_COLOR, self.run_clean_outliers,
radius=ICON_RADIUS
```

```
        )
        self.clean_btn.grid(row=0, column=0, sticky="nw", padx=30, pady=30)
        self.no_outliers_label = tk.Label(self.block_clean, text="",
font=FONT_TEXT,
            fg="#00FF00", bg=BG_SIDEBAR
        )
        self.no_outliers_label.grid(row=0, column=2, sticky="nw", padx=0,
pady=(45))

        self.log_text = tk.Text(self.block_clean, height=5, bg=BG_MAIN,
            fg=WHITE, font=FONT_CELL, wrap="none"
        )
        self.log_text.grid(row=1, column=0, columnspan=3, sticky="nsew",
padx=30, pady=(0, 0))
        self.log_text.config(state="disabled")

    def run_mardia(self):
        if self.app.current_data is None:
            messagebox.showwarning("Увага", "Спочатку оберіть файл у
Проєктах")
            return

        asym_normal, kurt_normal = run_mardia_test(self.app.current_data)
        self.test_results = (asym_normal, kurt_normal)

        self.asymmetry_label.config(
            text=f"Тест асиметрії: {'Розподіл нормальний.' if asym_normal
else 'Розподіл не нормальний.'}"
        )
        self.kurtosis_label.config(
            text=f"Тест ексцесу: {'Розподіл нормальний.' if kurt_normal else
'Розподіл не нормальний.'}"
        )

    def run_normalization(self):
        if self.app.current_data is None:
            messagebox.showwarning("Увага", "Спочатку оберіть файл у
Проєктах")
            return

        normalized_file_name = "Normalized_Data.csv"
        projects_frame = self.app.frames["projects"]

        existing_files = [f[0] for f in projects_frame.files_list]
        if normalized_file_name in existing_files:
            messagebox.showwarning("Увага", "Цей файл вже було нормалізовано,
оберіть інший файл!")
            return

        normalized_data, lambdas = normalize_file(self.app.current_data)
        self.app.current_data = normalized_data
        self.app.boxcox_lambdas = {"RFC": lambdas[0], "CBO": lambdas[1],
"WMC": lambdas[2]}

        original_table = load_data(projects_frame.current_file_path)
        csv_path =
os.path.join(os.path.dirname(projects_frame.current_file_path),
normalized_file_name)

        with open(csv_path, mode="w", newline="", encoding="utf-8") as f:
            writer = csv.writer(f, delimiter=';')
            writer.writerow(["URL", "RFC", "CBO", "WMC"])
            for i, row in enumerate(normalized_data):
                url = original_table[i][4] if len(original_table[i]) > 4 else
""
                writer.writerow([url, *row])
```

```
        projects_frame.files_list.append((normalized_file_name, csv_path))
        projects_frame.combo["values"] = [f[0] for f in
projects_frame.files_list]
        self.normalized_file_path = csv_path
        for col, l in zip(NUMERIC_COLS, lambdas):
            self.lambda_vars[col].set(f"{l:.4f}")
        messagebox.showinfo("Успіх", f"Дані успішно нормалізовано і збережено
у {csv_path}")

    def run_clean_outliers(self):
        if self.app.current_data is None:
            messagebox.showwarning("Увага", "Спочатку оберіть файл у
Проєктах")
            return

        projects_frame = self.app.frames["projects"]
        original_table = load_data(projects_frame.current_file_path)
        base_dir = os.path.dirname(projects_frame.current_file_path)

        cleaned_file_name = "Cleaned_Data.csv"
        cleaned_file_path = os.path.join(base_dir, cleaned_file_name)
        existing_files = [f[0] for f in projects_frame.files_list]
        if cleaned_file_name in existing_files:
            messagebox.showwarning("Увага", "Файл очищення від викидів вже
існує!")
            return

        log_folder = os.path.join(base_dir, "outlier_logs")
        cleaned_data, removed_indices, log_folder = remove_outliers_file(
            data=self.app.current_data,
            alpha=0.005,
            log_folder=log_folder,
            output_path=cleaned_file_path
        )

        self.app.current_data = cleaned_data
        self.cleaned_file_path = cleaned_file_path

        with open(cleaned_file_path, mode="w", newline="", encoding="utf-8")
as f:
            writer = csv.writer(f, delimiter=';')
            writer.writerow(["URL", "RFC", "CBO", "WMC"])
            for i, row in enumerate(cleaned_data):
                url = original_table[i][4] if len(original_table[i]) > 4 else
""
                writer.writerow([url, *row])

        projects_frame.files_list.append((cleaned_file_name,
cleaned_file_path))
        projects_frame.combo["values"] = [f[0] for f in
projects_frame.files_list]

        self.log_text.config(state="normal")
        self.log_text.delete("1.0", tk.END)

        log_files = sorted(os.listdir(log_folder))
        for log_file in log_files:
            path = os.path.join(log_folder, log_file)
            with open(path, "r", encoding="utf-8") as f:
                self.log_text.insert(tk.END, f"=== {log_file} ===\n")
                for line in f:
                    self.log_text.insert(tk.END, line)
                self.log_text.insert(tk.END, "\n")
        self.log_text.config(state="disabled")
        if len(removed_indices) == 0:
            if not hasattr(self, 'no_outliers_label'):
```

```
                self.no_outliers_label = tk.Label(
                    self.block_clean,
                    text="Викидів не було виявлено!",
                    font=FONT_TEXT,
                    fg="#00FF00",
                    bg=BG_SIDEBAR
                )
                self.no_outliers_label.pack(side="top", anchor="ne", padx=30,
pady=(10,0))
            else:
                self.no_outliers_label.config(text="Викидів не було
виявлено!")
        else:
            if hasattr(self, 'no_outliers_label'):
                self.no_outliers_label.config(text="")
```

Текст файлу **frames/RegressionFrame.py**

```
import io
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
from frames.BaseFrame import BaseFrame
from model.constants import *
from model.utils import UIUtils
from model.regression import build_regression

class RegressionFrame(BaseFrame):
    def __init__(self, parent, app):
        super().__init__(parent, app)
        self.model_var = tk.IntVar(value=0)
        self.buttons_group = []
        self.linear_image = None
        self.nonlinear_image = None

        self.main.grid_rowconfigure(0, weight=1)
        self.main.grid_rowconfigure(1, weight=0)
        self.main.grid_rowconfigure(2, weight=4)
        self.main.grid_rowconfigure(3, weight=1)
        self.main.grid_rowconfigure(4, weight=8)
        self.main.grid_rowconfigure(5, weight=1)
        self.main.grid_rowconfigure(6, weight=2)
        self.main.grid_rowconfigure(7, weight=1)
        self.main.grid_rowconfigure(8, weight=2)
        self.main.grid_columnconfigure(0, weight=1)

        self.create_model_selector()
        self.create_linear_block()
        self.create_nonlinear_block()
        self.create_metrics_block()
        self.create_reference_block()

    def create_model_selector(self):
        wrapper_width = 740
        wrapper_height = 70
        wrapper_canvas = tk.Canvas(self.main, width=wrapper_width,
height=wrapper_height, bg=BG_MAIN, highlightthickness=0)
        wrapper_canvas.grid(row=0, column=0, pady=0, sticky="n")
        UIUtils.draw_rounded_rect(wrapper_canvas, 0, 0, wrapper_width,
wrapper_height, r=ICON_RADIUS*2, fill=BG_SIDEBAR)
        self.button_frame = tk.Frame(wrapper_canvas, bg=BG_SIDEBAR)
        self.button_frame.place(relx=0.5, rely=0.5, anchor="center")
        labels = ["RFC (CBO, WMC)", "CBO (WMC, RFC)", "WMC (RFC, CBO)"]
```

```
        for idx, text in enumerate(labels):
            btn = UIUtils.create_special_rounded_button(
                self.button_frame,
                text,
                LONG_CELL_WIDTH,
                40,
                command=lambda i=idx: self.select_model(i),
                radius=ICON_RADIUS,
                active_buttons_list=self.buttons_group
            )
            btn.grid(row=0, column=idx, padx=10, pady=0)

        if self.buttons_group:
            self.buttons_group[0].set_active(True)

    def select_model(self, idx):
        self.model_var.set(idx)
        self.run_regression()

    def create_linear_block(self):
        self.linear_frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.linear_frame.grid(row=2, column=0, sticky="nsew", padx=0,
pady=0)

        tk.Label(self.linear_frame, text="Лінійне регресійне рівняння:",
                 font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR).grid(row=0,
column=0, sticky="w", padx=30, pady=(15,15))

        self.linear_formula_label = tk.Label(
            self.linear_frame,
            text="Ẑ_y = b0 + b1 Z_x1 + b2 Z_x2 + ε",
            font=FONT_TEXT,
            fg=WHITE,
            bg=BG_SIDEBAR,
            justify="left"
        )
        self.linear_formula_label.grid(row=1, column=0, sticky="w", padx=30)

    def create_nonlinear_block(self):
        self.nonlinear_frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.nonlinear_frame.grid(row=4, column=0, sticky="nsew", padx=0,
pady=0)

        tk.Label(self.nonlinear_frame, text="Нелінійне регресійне рівняння:",
                 font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR).grid(row=0,
column=0, sticky="w", padx=30, pady=(15,15))

        self.nonlinear_formula_label = tk.Label(
            self.nonlinear_frame,
            text="Ŷ = [ (b0 + b1*(X1^λx1-1)/λx1 + b2*(X2^λx2-1)/λx2) *
λy]^(1/λy) + ε",
            font=FONT_TEXT,
            fg=WHITE,
            bg=BG_SIDEBAR,
            justify="left"
        )
        self.nonlinear_formula_label.grid(row=4, column=0, sticky="w",
padx=30)

    def create_metrics_block(self):
        self.metrics_frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.metrics_frame.grid(row=6, column=0, sticky="nsew", padx=0,
pady=0)
        tk.Label(self.metrics_frame, text="Результати:", font=FONT_TEXT,
fg="#7CFC00", bg=BG_SIDEBAR)\
            .grid(row=0, column=0, sticky="w", padx=30, pady=(15,15))
```

```
        self.metrics_label = tk.Label(self.metrics_frame, text="R² =
MMRE =                          PRED(0.25) =", font=FONT_TEXT,
fg="#7CFC00", bg=BG_SIDEBAR)
        self.metrics_label.grid(row=6, column=0, sticky="w", padx=30)

    def create_reference_block(self):
        self.reference_frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.reference_frame.grid(row=8, column=0, sticky="nsew", padx=0,
pady=0)

        tk.Label(self.reference_frame, text="Референтні значення якості:",
font=FONT_TEXT, fg="#FFD700",
            bg=BG_SIDEBAR).grid(row=0, column=0, sticky="w", padx=30,
pady=(15,15))

        tk.Label(self.reference_frame, text="R² > 0.7                    MMRE <
0.25                    PRED(0.25) > 0.75",
            font=FONT_TEXT, fg="#FFD700", bg=BG_SIDEBAR).grid(row=1,
column=0, sticky="w", padx=30)

    def render_latex(self, formula, font_size=16):
        fig = plt.figure(figsize=(0.01, 0.01))
        fig.patch.set_alpha(0)
        plt.axis('off')
        plt.text(0, 0, formula, fontsize=font_size, color=WHITE)

        buf = io.BytesIO()
        fig.savefig(buf, format='png', bbox_inches='tight', dpi=125,
transparent=True)
        plt.close(fig)
        buf.seek(0)
        img = Image.open(buf)
        photo = ImageTk.PhotoImage(img)
        buf.close()
        return photo

    def run_regression(self):
        if self.app.current_data is None:
            return
        if self.app.boxcox_lambdas is None:
            messagebox.showwarning("Увага", "Спочатку необхідно виконати
нормалізацію даних")
            return

        idx = self.model_var.get()
        result = build_regression(data=self.app.current_data,
            lambdas=self.app.boxcox_lambdas, target_idx=idx
        )

        b0 = result["linear"]["b0"]
        b1 = result["linear"]["b1"]
        b2 = result["linear"]["b2"]
        x1, x2 = result["x_order"]
        lam_y = result["lambdas"]["Y"]
        lam_x1 = result["lambdas"]["X1"]
        lam_x2 = result["lambdas"]["X2"]
        sigma = result["sigma"]

        linear_formula = rf"$\hat{{Z}}y = {b0:.4f} + {b1:.4f} \cdot
Z_{{{x1}}} + {b2:.4f} \cdot Z_{{{x2}}} + {sigma:.4f}$"
        nonlinear_formula = (
            rf"$\hat{{Y}} = [ ( {b0:.4f} + {b1:.4f} "
            rf"\frac{{X_{{{x1}}}^{{{lam_x1:.4f}}}-1}}{{{lam_x1:.4f}}} +
{b2:.4f} "
            rf"\frac{{X_{{{x2}}}^{{{lam_x2:.4f}}}-1}}{{{lam_x2:.4f}}} ) \cdot
{lam_y:.4f} ]^"
```

```
            rf"\frac{{1}}{{{lam_y:.4f}}} + {sigma:.4f}$"
        )

        self.linear_image = self.render_latex(linear_formula, font_size=10)
        self.nonlinear_image = self.render_latex(nonlinear_formula,
font_size=10)

        self.linear_formula_label.config(image=self.linear_image, text="")
        self.nonlinear_formula_label.config(image=self.nonlinear_image,
text="")

        m = result["metrics"]
        self.metrics_label.config(
            text=f"R² = {m['R2']:.4f}    MMRE = {m['MMRE']:.4f}    PRED(0.25)
= {m['PRED']:.4f}"
        )
```

Текст файлу **frames/PredictionFrame.py**

```
import numpy as np
import tkinter as tk
from tkinter import ttk, messagebox
from model.constants import *
from model.utils import UIUtils
from frames.BaseFrame import BaseFrame
from model.regression import build_regression
from model.prediction import (nonlinear_predict, fill_intervals_table,
classify_project_quality,
    compute_intervals, compute_intervals_for_custom)
from model.data_loader import load_data

class PredictionFrame(BaseFrame):
    def __init__(self, parent, app):
        super().__init__(parent, app)
        self.app = app
        self.model_var = tk.IntVar(value=0)
        self.project_var = tk.StringVar()
        self.buttons_group = []
        self.input_vars = {k: tk.StringVar() for k in ("RFC", "CBO", "WMC")}
        self.main.grid_columnconfigure(0, weight=1)
        for i, w in enumerate((1, 1, 1, 1, 2, 1, 2, 1, 2)):
            self.main.grid_rowconfigure(i, weight=w)

        self.create_model_selector()
        self.create_project_selector()
        self.create_input_block()
        self.create_result_block()
        self.create_ranges_block()
        self.create_intervals_button()

    def create_model_selector(self):
        canvas = tk.Canvas(self.main, width=740, height=70, bg=BG_MAIN,
highlightthickness=0)
        canvas.grid(row=0, column=0)
        UIUtils.draw_rounded_rect(canvas, 0, 0, 740, 70, r=ICON_RADIUS * 2,
fill=BG_SIDEBAR)
        frame = tk.Frame(canvas, bg=BG_SIDEBAR)
        frame.place(relx=0.5, rely=0.5, anchor="center")

        for i, text in enumerate(("RFC (CBO, WMC)", "CBO (WMC, RFC)", "WMC
(RFC, CBO)")):
            UIUtils.create_special_rounded_button(
                frame, text, LONG_CELL_WIDTH, 40,
                command=lambda idx=i: self.select_model(idx),
```

```
                radius=ICON_RADIUS, active_buttons_list=self.buttons_group
            ).grid(row=0, column=i, padx=10)

        if self.buttons_group:
            self.buttons_group[0].set_active(True)

    def select_model(self, idx):
        if self.app.boxcox_lambdas is None:
            messagebox.showwarning("Увага", "Спочатку виконайте
нормалізацію")
            return
        self.model_var.set(idx)
        self.calculate_prediction()

    def create_project_selector(self):
        bar = tk.Frame(self.main, bg=BG_MAIN, height=60)
        bar.grid(row=2, column=0, sticky="ew")
        bar.grid_propagate(False)
        canvas = UIUtils.create_rounded_button(bar, "", 740, 60, BG_SIDEBAR,
command=lambda: None, radius=ICON_RADIUS)
        canvas.pack()
        self.combo = ttk.Combobox(canvas, textvariable=self.project_var,
font=FONT_TEXT, state="readonly")
        self.combo.pack(fill="both", expand=True, padx=25)
        self.combo.configure(state="disabled")
        self.combo.bind("<<ComboboxSelected>>", self.on_project_selected)

    def update_projects_list(self):
        pf = self.app.frames["projects"]
        if not pf.current_file_path:
            self.combo.configure(state="disabled")
            self.combo["values"] = []
            self.project_var.set("")
            return

        table = load_data(pf.current_file_path)
        self.projects_raw = []
        values = ["Кастом"]

        for i, (_, rfc, cbo, wmc, _) in enumerate(table, start=1):
            self.projects_raw.append((rfc, cbo, wmc))
            values.append(f"Застосунок №{i}")

        self.combo.configure(state="readonly", values=values)
        self.project_var.set("Кастом")

    def on_project_selected(self, event=None):
        choice = self.project_var.get()
        if choice == "Кастом":
            for v in self.input_vars.values():
                v.set("")
            return

        idx = int(choice.split("№")[1]) - 1
        rfc, cbo, wmc = self.projects_raw[idx]
        self.input_vars["RFC"].set(str(rfc))
        self.input_vars["CBO"].set(str(cbo))
        self.input_vars["WMC"].set(str(wmc))
        self.calculate_prediction()

    def create_input_block(self):
        frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        frame.grid(row=4, column=0, sticky="ew", pady=5)
        for i, key in enumerate(("RFC", "CBO", "WMC")):
            frame.grid_columnconfigure(i, weight=1)
            block = tk.Frame(frame, bg=BG_SIDEBAR)
```

```
            block.grid(row=0, column=i, padx=20, pady=15)
            tk.Label(block, text=key, font=FONT_TEXT,
                     fg=WHITE, bg=BG_SIDEBAR).pack()
            tk.Entry(block, textvariable=self.input_vars[key],
                     font=FONT_TEXT, width=10,
                     justify="center").pack(pady=5)

    def create_result_block(self):
        frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        frame.grid(row=6, column=0, sticky="ew", pady=5)
        frame.grid_columnconfigure(0, weight=1)
        frame.grid_columnconfigure(1, weight=1)

        self.y_label = tk.Label(frame, text="Ŷ = —",
                                font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR)
        self.y_label.grid(row=0, column=0, padx=30, pady=20, sticky="w")

        self.quality_label = tk.Label(frame, text="Якість застосунку: —",
                                      font=FONT_TEXT, fg=WHITE,
bg=BG_SIDEBAR)
        self.quality_label.grid(row=0, column=1, padx=30, pady=20,
sticky="e")

    def create_ranges_block(self):
        self.ranges_frame = tk.Frame(self.main, bg=BG_SIDEBAR)
        self.ranges_frame.grid(row=8, column=0, sticky="ew", pady=(5, 15))
        self.range_labels = {}

        for i, key in enumerate(("RFC", "CBO", "WMC")):
            self.ranges_frame.grid_columnconfigure(i, weight=1)
            lbl = tk.Label(self.ranges_frame, text=f"{key}: [—; —]",
                           font=FONT_TEXT, fg=WHITE, bg=BG_SIDEBAR)
            lbl.grid(row=0, column=i, pady=15)
            self.range_labels[key] = lbl

    def update_ranges_from_file(self):
        pf = self.app.frames["projects"]
        if not pf.current_file_path:
            return

        table = load_data(pf.current_file_path)
        ranges = {
            "RFC": (min(float(r[1]) for r in table), max(float(r[1]) for r in
table)),
            "CBO": (min(float(r[2]) for r in table), max(float(r[2]) for r in
table)),
            "WMC": (min(float(r[3]) for r in table), max(float(r[3]) for r in
table)),
        }

        for k, (a, b) in ranges.items():
            self.range_labels[k].config(text=f"{k}: [{a:.4f}; {b:.4f}]")

    def calculate_prediction(self):
        try:
            x_input = [float(self.input_vars[k].get()) for k in ("RFC",
"CBO", "WMC")]
        except ValueError:
            return
        if self.app.current_data is None or self.app.boxcox_lambdas is None:
            return

        result = build_regression(self.app.current_data,
self.app.boxcox_lambdas, self.model_var.get())
        y_hat = nonlinear_predict(x_input, result, self.model_var.get())
        self.y_label.config(text=f"Ŷ = {y_hat:.4f}")
```

```
        choice = self.project_var.get()
        if choice == "Кастом":
            y_hat, *_ , quality = compute_intervals_for_custom(
                x_input, result, self.app.current_data
            )
            color = {"висока": "green", "середня": "orange", "низька":
"red"}.get(
                quality, WHITE
            )
            self.y_label.config(text=f"Ŷ = {y_hat:.4f}")
            self.quality_label.config(text=f"Якість застосунку: {quality}",
fg=color)
            return

        raw = load_data(self.app.frames["projects"].current_file_path)
        y_actual = np.array([float(r[1 + self.model_var.get()]) for r in
raw])

        ci_l, ci_u, pi_l, pi_u = compute_intervals(
            self.app.current_data, self.app.boxcox_lambdas,
            self.model_var.get(), build_regression
        )

        idx = int(choice.split("№")[1]) - 1
        quality = classify_project_quality(
            y_actual[idx], ci_l[idx], ci_u[idx], pi_l[idx], pi_u[idx]
        )

        color = {"висока": "green", "середня": "orange", "низька":
"red"}.get(
            quality, WHITE
        )
        self.quality_label.config(text=f"Якість застосунку: {quality}",
fg=color)

    def create_intervals_button(self):
        frame = tk.Frame(self.main, bg=BG_MAIN)
        frame.grid(row=9, column=0, pady=(10, 20))
        UIUtils.create_special_rounded_button(
            frame, "Переглянути інтервали", 300, 45,
            command=self.open_intervals_window, radius=ICON_RADIUS
        ).pack()

    def open_intervals_window(self):
        if self.app.current_data is None or self.app.boxcox_lambdas is None:
            messagebox.showwarning("Увага", "Немає даних для розрахунку
інтервалів")
            return

        win = tk.Toplevel(self)
        win.title("Довірчі та прогнозні інтервали")
        win.geometry("1000x600")
        win.configure(bg=BG_MAIN)

        top = tk.Frame(win, bg=BG_SIDEBAR, height=60)
        top.pack(fill="x")
        top.pack_propagate(False)

        UIUtils.create_special_rounded_button(
            top, "Повернутись", 180, 40,
            command=win.destroy, radius=ICON_RADIUS
        ).pack(side="left", padx=20, pady=10)
        table_frame = tk.Frame(win, bg=BG_MAIN)
        table_frame.pack(fill="both", expand=True, padx=20, pady=20)
        self.create_intervals_table(table_frame)
    def create_intervals_table(self, parent):
```

```
        columns = ("idx", "y", "y_hat", "ci_low", "ci_high", "pi_low",
"pi_high")
        headers = ("№", "Y", "Ŷ", "Довірчий ↓", "Довірчий ↑",
                   "Прогнозний ↓", "Прогнозний ↑")
        tree = ttk.Treeview(parent, columns=columns,
                            show="headings", height=15)
        for c, h in zip(columns, headers):
            tree.heading(c, text=h)
            tree.column(c, anchor="center", width=120)
        tree.pack(fill="both", expand=True)
        raw_data = np.array([
            [float(r[1]), float(r[2]), float(r[3])]
            for r in load_data(self.app.frames["projects"].current_file_path)
        ])
        fill_intervals_table(
            tree, self.model_var, raw_data,
            self.app.current_data, self.app.boxcox_lambdas,
            build_regression
        )
```

Текст файлу **frames/BaseFrame.py**

```
import tkinter as tk
from model.constants import *

class BaseFrame(tk.Frame):
    def __init__(self, parent, app):
        super().__init__(parent, bg=BG_MAIN)
        self.app = app
        self.main = tk.Frame(self, bg=BG_MAIN)
        self.main.pack(fill="both", expand=True, padx=30, pady=30)
```

Текст файлу **model/data_loader.py**

```
import os
import pandas as pd
REQUIRED_COLUMNS_CSV = ["URL", "RFC", "CBO", "WMC"]

def load_data(file_path: str):
    if not file_path:
        raise ValueError("Файл не обрано")
    ext = os.path.splitext(file_path)[1].lower()
    if ext != ".csv":
        raise ValueError("Формат завантаженого файлу не підтримується.
Завантажте файл CSV.")
    df = pd.read_csv(file_path, sep=';', encoding="utf-8-sig")
    df.columns = df.columns.str.strip()
    for col in REQUIRED_COLUMNS_CSV:
        if col not in df.columns:
            raise ValueError(f"В файлі відсутня колонка '{col}'")
    df = df[REQUIRED_COLUMNS_CSV]
        table_data = []
    for i, row in df.iterrows():
        try:
            rfc = f"{float(str(row['RFC']).replace(',', '.')):.4f}"
            cbo = f"{float(str(row['CBO']).replace(',', '.')):.4f}"
            wmc = f"{float(str(row['WMC']).replace(',', '.')):.4f}"
        except ValueError:
            continue
        url = str(row['URL']).strip()
        table_data.append((i + 1, rfc, cbo, wmc, url))
    return table_data
```

Текст файлу **model/mardia_test.py**

```
import numpy as np
from scipy.stats import chi2, norm
ALPHA = 0.005
def run_mardia_test(data: np.ndarray, save_d2: bool = True) -> tuple:
    N, k = data.shape
    mean_vector = data.mean(axis=0)
    cov_matrix = np.cov(data, rowvar=False, ddof=0)
    inv_cov_matrix = np.linalg.inv(cov_matrix)
    centered = data - mean_vector
    D2 = np.einsum('ij,jk,ik->i', centered, inv_cov_matrix, centered)

    skewness = 0
    for i in range(N):
        for j in range(N):
            xi = centered[i]
            xj = centered[j]
            skewness += (np.dot(np.dot(xi, inv_cov_matrix), xj))**3
    skewness /= N**2
    Z_skew = (N * skewness) / 6

    kurtosis = 0
    for i in range(N):
        xi = centered[i]
        kurtosis += (np.dot(np.dot(xi, inv_cov_matrix), xi))**2
    kurtosis /= N
    chi_crit = chi2.ppf(1 - ALPHA, df=k * (k + 1) * (k + 2) / 6)
    mu = k * (k + 2)
    sigma = np.sqrt(8 * k * (k + 2) / N)
    z_crit = norm.ppf(1 - ALPHA, loc=mu, scale=sigma)
    asym_normal = Z_skew <= chi_crit
    kurt_normal = kurtosis <= z_crit

    return asym_normal, kurt_normal
```

Текст файлу **model/normalization.py**

```
import numpy as np
from scipy.optimize import minimize_scalar

def boxcox_transform(x, lam):
    if np.any(x <= 0):
        raise ValueError("Значення мають бути строго додатними.")
    return np.log(x) if lam == 0 else (np.power(x, lam) - 1) / lam

def optimal_lambdas(data: np.ndarray):
    return [
        minimize_scalar(
            lambda l: -((l - 1) * np.sum(np.log(col)) -
                        (len(col) / 2) *
np.log(np.mean((boxcox_transform(col, l) - np.mean(boxcox_transform(col,
l)))**2))),
            bounds=(-5, 5), method='bounded'
        ).x
        for col in data.T
    ]

def normalize_data(data: np.ndarray, lambdas):
    return np.column_stack([boxcox_transform(data[:, i], l) for i, l in
enumerate(lambdas)])

def normalize_file(data: np.ndarray):
    return normalize_data(data, optimal_lambdas(data)), optimal_lambdas(data)
```

Текст файлу **model/regression.py**

```
import numpy as np
from sklearn.metrics import r2_score

def inverse_boxcox(z: np.ndarray, lam: float) -> np.ndarray:
    if lam == 0:
        return np.exp(z)
    return np.power(lam * z + 1, 1 / lam)

def build_regression(data: np.ndarray, lambdas: dict, target_idx: int):
    names = ["RFC", "CBO", "WMC"]
    y_name = names[target_idx]
    lam_y = lambdas[y_name]

    Zy = data[:, target_idx]
    if target_idx == 0:       # RFC ← (CBO, WMC)
        Zx = np.column_stack([data[:, 1], data[:, 2]])
        x_order = ["CBO", "WMC"]
    elif target_idx == 1:     # CBO ← (WMC, RFC)
        Zx = np.column_stack([data[:, 2], data[:, 0]])
        x_order = ["WMC", "RFC"]
    else:                     # WMC ← (RFC, CBO)
        Zx = np.column_stack([data[:, 0], data[:, 1]])
        x_order = ["RFC", "CBO"]
    lam_x1 = lambdas[x_order[0]]
    lam_x2 = lambdas[x_order[1]]

    X = np.column_stack([
        np.ones(len(Zx)),
        Zx[:, 0],
        Zx[:, 1]
    ])

    beta = np.linalg.inv(X.T @ X) @ X.T @ Zy
    b0, b1, b2 = beta
    Zy_hat = X @ beta
    Y_hat = inverse_boxcox(Zy_hat, lam_y)
    y_true = inverse_boxcox(Zy, lam_y)
    r2 = r2_score(y_true, Y_hat)
    mre = np.abs(y_true - Y_hat) / y_true
    mmre = np.mean(mre)
    pred25 = np.mean(mre <= 0.25)
    sigma = np.std(y_true - Y_hat, ddof=1)

    return {"linear": {"b0": float(b0), "b1": float(b1), "b2": float(b2)},
"x_order": x_order,
        "lambdas": {"Y": float(lam_y), "X1": float(lam_x1), "X2":
float(lam_x2)}, "sigma": float(sigma),
        "metrics": {"R2": float(r2), "MMRE": float(mmre), "PRED":
float(pred25)}
    }
```

ДОДАТОК В – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JHQualityMeter»

Програмне забезпечення «JHQualityMeter» розроблене для автоматизації процесу оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Воно забезпечує можливість нормалізовувати дані, очищати їх від викидів, будувати на основі очищеної вибірки регресійні моделі для оцінювання якості застосунків.

Призначення та функціональні можливості:

1. Завантаження вхідних даних

Програма підтримує завантаження файлів формату CSV з вхідними даними, що подані у вигляді URL адрес застосунків та метрик RFC, CBO та WMC.

2. Перевірка нормальності розподілу даних

Завантажені дані перевіряються програмою на наявність нормального розподілу даних за тестом Мардіа. Нормальний розподіл мають дані, значення асиметрії та ексцесу яких не перевищують критичні значення тестових статистик.

3. Нормалізація даних

Нормалізація даних відбувається за методом Бокса-Кокса для кожного параметру (2.6), внаслідок чого формується файл з нормалізованими даними.

4. Видалення викидів

Програма здійснює поетапний пошук та видалення викидів з вибірки нормалізованих даних за умови, якщо розраховані значення квадратів відстаней Махаланобіса D^2 (2.1) або тестових статистик T_S (2.7) перевищують відповідні критичні значення $\chi^2_{0,005,10}$ та $F_{\text{крит}}$ (2.8). Процес має ітеративний характер та повторюватиметься допоки всі викиди не будуть видалені.

5. Побудова регресійних моделей

Відбувається розрахунок параметрів моделі, а саме коефіцієнтів b_0, b_1 b_2 (2.12), використовуються побудовані нелінійні регресійні рівняння для прогнозування метрик RFC, CBO та WMC на основі відповідних моделей RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO), а також розраховуються довірчі інтервали та інтервали прогнозування (2.11).

6. Тестування моделі

Визначаються показники якості побудованої моделі R^2, $MMRE$ та $PRED(0.25)$ та перевіряється достовірність оцінок якості застосунків, що ґрунтуються на порівнянні фактичних значень метрик RFC, CBO, WMC з межами інтервалів.

7. Збереження результатів

Результати у вигляді нормалізованих та очищених від викидів вибірок даних зберігаються у зовнішні файли через взаємодію з файловою системою.

Мова розробки та технології:

Програмне забезпечення «JHQualityMeter» розроблено мовою програмування Python з використанням графічної бібліотеки Tkinter.

Python: використовується для реалізації всієї логіки обробки даних і математичних розрахунків, зокрема завантаження та валідації CSV-файлів, нормалізації даних методом Бокса–Кокса, перевірки нормальності розподілу за тестом Мардіа, видалення викидів, побудови лінійних та нелінійних регресійних моделей, обчислення показників якості, а також розрахунку довірчих і прогнозних інтервалів.

Tkinter: забезпечує створення графічного інтерфейсу користувача, який дозволяє взаємодіяти з програмою в інтерактивному режимі. За допомогою Tkinter реалізовано навігацію між функціональними модулями, завантаження та вибір файлів, введення значень метрик, відображення результатів моделювання, формул регресійних рівнянь, інтервалів та класифікації якості застосунків.

Принцип роботи:

– Користувач завантажує файл формату CSV із метриками програмних застосунків.

– Програма виконує перевірку нормальності розподілу та попередню обробку даних, що включає нормалізацію метрик та поетапне видалення викидів.

– На основі очищених даних будуються лінійні та нелінійні регресійні рівняння з використанням перетворення Бокса-Кокса за обраною моделлю.

– Виконується розрахунок прогнозних значень, а також довірчих та прогнозних інтервалів для кожного спостереження.

– Користувач може обрати застосунок із набору даних або ввести власні значення метрик RFC, CBO та WMC для отримання індивідуальної оцінки.

– Якість застосунку автоматично визначається та класифікується за рівнями («висока», «середня», «низька») відповідно до положення фактичного значення відносно інтервалів.

– Отримані результати можуть бути переглянуті в табличному вигляді та збережені у файл формату CSV.

Технічні характеристики:

Системні вимоги:

– Операційна система: Windows, macOS.

Мінімальні вимоги:

– Процесор: Intel Pentium 4 або новіше із підтримкою SSE3;

– Оперативна пам'ять: 2 ГБ або більше;

– Дисковий простір: 200 МБ.

Функціональні особливості:

– Підтримка роботи з великими вибірками даних;

– Швидке виконання розрахунків завдяки реалізації алгоритмів через Python;

– Зручний інтерфейс для завантаження та перегляду даних.

Переваги використання:

– Автоматизація процесів. Програма автоматично виконує обробку даних, побудову моделі та оцінювання якості застосунків.

– Достовірність результатів. Застосування нормалізації, видалення викидів та нелінійної регресії підвищує точність прогнозів.

– Гнучкість аналізу. Підтримується робота як з наборами даних, так і з власними значеннями метрик RFC, CBO та WMC.

– Зручна інтерпретація. Результати подаються у вигляді прогнозів, інтервалів та зрозумілої класифікації.

– Автономність. Усі обчислення виконуються локально без використання зовнішніх сервісів.

«JHQualityMeter» є ефективним та надійним засобом для розробників, що автоматизує та спрощує процес оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JHQualityMeter»

Програмне забезпечення «JHQualityMeter» призначене для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Програма містить весь необхідний інструментарій для обробки даних, побудови регресійних моделей та оцінювання на їх основі якості застосунків.

1. Завантаження та встановлення

– Завантажте виконуваний файл JHQualityMeter;

– Розпакуйте архів та запустіть файл JHQualityMeter.exe.

2. Завантаження даних

– Натисніть на кнопку «Завантажити CSV» на головній сторінці програми;

– Оберіть файл формату CSV, де дані наведено в наступній послідовності: URL, RFC, CBO, WMC;

– Дані будуть автоматично завантажені та відображені у відповідній таблиці на вкладці «Проєкти».

На рисунку Г.1 проілюстровано завантаження даних в «JHQualityMeter».

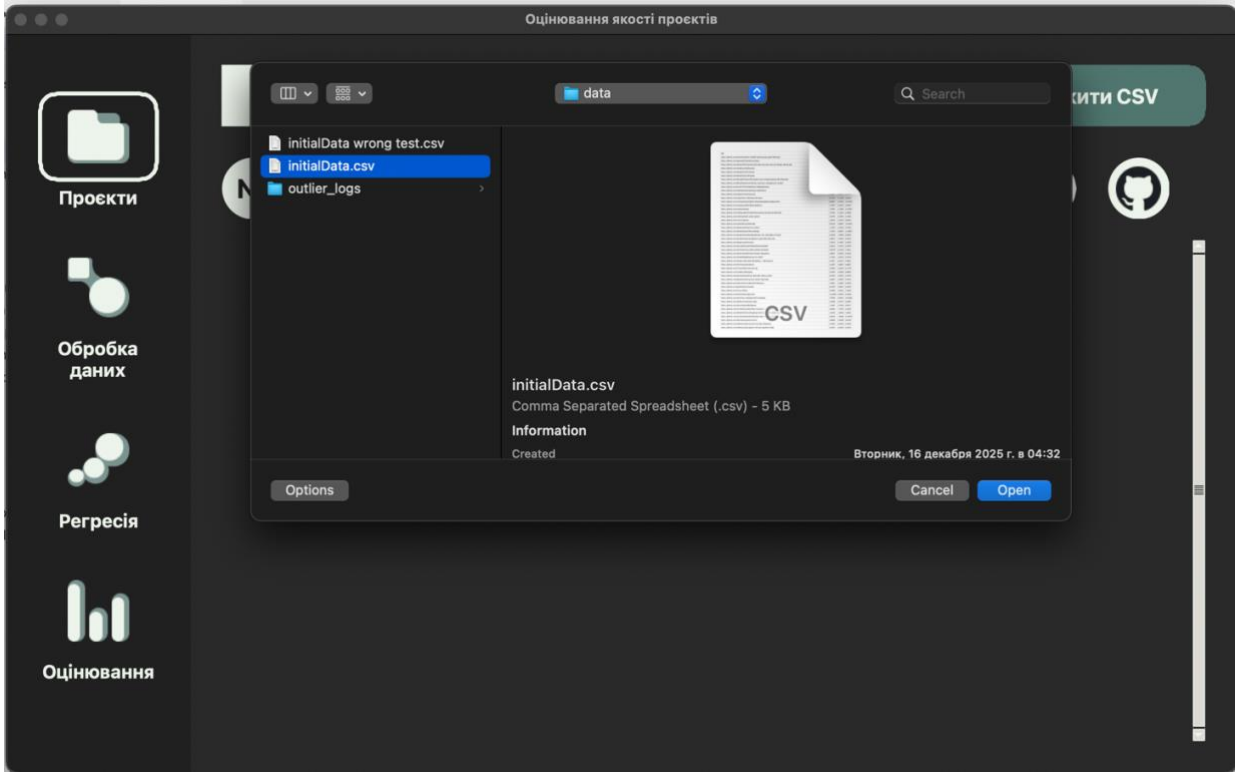


Рисунок Г.1 – Завантаження даних в «JHQualityMeter»

3. Перегляд та обробка даних

– Використовуйте бічне меню для переходу між різними вкладками програми;

– Вкладка «Проєкти» – перегляд завантажених застосунків;

– Вкладка «Обробка даних» – перевірка даних на нормальність розподілу, нормалізація та очищення від викидів, а також перегляд ітерацій виявлених викидів;

– Вкладка «Регресія» – перегляд побудованих регресійних рівнянь, їх коефіцієнтів та критеріїв якості побудованих нелінійних моделей;

– Вкладка «Оцінювання» – оцінювання якості застосунків за побудованими моделями RFC(CBO, WMC), CBO(WMC, RFC) та WMC(RFC, CBO).

4. Використання побудованих моделей

– Перейдіть до вкладки «Оцінювання»;

– Для перегляду оцінювання якості застосунків завантаженої вибірки, оберіть з розкривного списку номер застосунку, оцінку якого бажаєте переглянути;

– Для розрахунку оцінки якості стороннього застосунку, оберіть з розкривного списку пункт «Кастом» та введіть у порожні текстові поля значення метрик RFC, CBO та WMC;

– Обирайте різні моделі побудови регресій для отримання найбільш повного оцінювання якості застосунків.

– Переглядайте побудовані довірчі інтервали та інтервали прогнозування по натисканню кнопки «Переглянути інтервали».

На рисунку Г.2 проілюстровано використання однієї з побудованих моделей.

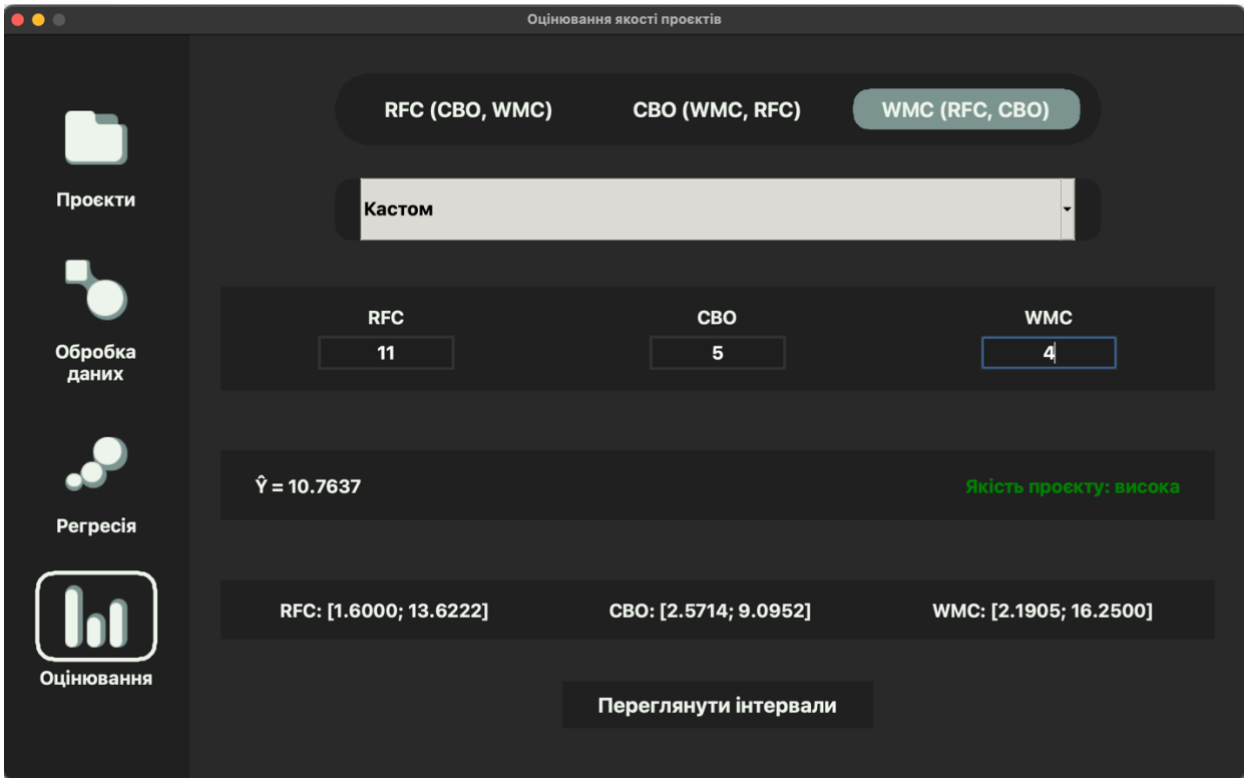


Рисунок Г.2 – Використання побудованої моделі WMC(RFC, CBO) в «JHQualityMeter»

ДОДАТОК Д – ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JHQualityMeter»

1. Об’єкт випробування

Об’єктом випробування є програмне забезпечення «JHQualityMeter» для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate. Програмне забезпечення використовує регресійні моделі, що базуються на метриках RFC, CBO та WMC для обробки та оцінювання якості застосунків.

2. Мета випробування

Метою випробування програмного забезпечення «JHQualityMeter» є перевірка належності виконання функціональної частини та достовірності побудованих регресійних моделей, які мають повністю відповідати заявленим вимогам до надійності та продуктивності. Основними цілями випробування є:

– Оцінювання функціональної відповідності програмного забезпечення заявленим вимогам;

– Перевірка коректності розрахунків;

– Тестування продуктивності та стійкості до навантажень;

– Виявлення та виправлення можливих дефектів та недоліків у роботі.

3. Вимоги до програмного забезпечення

В ході проведення випробувань програмного забезпечення «JHQualityMeter» функціональні характеристики мають бути перевірені на відповідність вимогам, що викладені в пункті «Вимоги до функціональних характеристик» технічного завдання, наведеного в Додатку А.

4. Вимоги до програмної документації

Програмна документація має містити в собі такі документи:

1) Текст програми.

2) Опис програми.

3) Програма та методика випробувань.

4) Інструкція користувача.

5. Засоби та порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

– CPU: Intel(R) Pentium G4560 @ 3.50GHz;

– RAM: 8 GB 2400MHz;

– SSD: Crucial MX500 500GB;

– Монітор з роздільною здатністю 1920×1080 пікселів.

5.2 Програмні засоби, що використовуються під час випробувань

– Windows 10 Home x64;

– Visual Studio Code 1.107.1;

– Python 3.13;

– Tkinter 8.6.

5.3 Порядок проведення випробувань

Випробування програми «JHQualityMeter» складається з перевірки вимог до програмної документації та перевірки вимог до функціональних характеристик програмного забезпечення.

6. Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Під час перевірки вимог до програмної документації треба перевірити:

– наявність усіх необхідних документів і їх відповідність вимогам до змісту й формату;

– відповідність документів вимогам до змісту, зокрема перевірити наявність необхідних даних і опису програми;

– відповідність програми та методики випробувань вимогам до тестування;

– чи містить інструкція користувача зрозумілі кроки для використання програмного забезпечення;

– структуру й оформлення документів, включаючи заголовки, нумерацію сторінок, зміст, читабельність тексту, а також його відповідність граматичним і орфографічним правилам;

– чи відповідають документи стандартам документації та чи мають зрозумілу структуру.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

В ході тестування програмного забезпечення «JHQualityMeter» для оцінювання якості застосунків з відкритим кодом на Java, що створюються з використанням фреймворку Hibernate, необхідно виконати тести для перевірки вимог до функціональних характеристик, які полягають в наступному:

– Завантаження даних з файлу формату CSV: коректність завантаження URL адрес та метрик RFC, CBO, WMC;

– Перевірка нормальності розподілу даних: правильність розрахунків тестових статистик за тестом Мардіа;

– Нормалізація даних: правильність нормалізації даних методом Бокса-Кокса;

– Видалення викидів з вибірки: ідентифікація та коректне видалення з вибірки аномальних значень;

– Побудова регресійних моделей: правильність розрахунку коефіцієнтів моделі b_0, b_1 та b_2 та прогнозування метрик RFC, CBO та WMC;

– Розрахунок довірчих інтервалів та інтервалів прогнозування: правильність обчислення інтервалів для усіх моделей;

– Оцінювання якості побудованих моделей: правильність розрахунку показників якості R^2, $MMRE$ та $PRED(0.25)$;

– Оцінювання якості застосунків: коректність класифікації якості та відповідність оцінок якості кожному з застосунків;

– Збереження результатів у форматі CSV: коректність збереження вибірки нормалізованих та очищених даних в окремі файли;

– Обробка некоректних даних: стабільність та надійність роботи програми при завантажені порожніх файлів або файлів з некоректними даними.

В таблиці Д.1 наведено перелік випробувань.

Таблиця Д.1 – Перелік випробувань основних функціональних можливостей

№	Дія	Очікуваний результат
1	Завантаження даних з файлу формату CSV	URL адреси та метрики RFC, CBO, WMC успішно завантажено без помилок
2	Перевірка нормальності розподілу даних	Дані перевірено на нормальність розподілу за тестом Мардіа
3	Нормалізація даних	Дані нормалізовано методом Бокса-Кокса, відображені оптимальні значення лямбд для кожної метрики
4	Видалення викидів з вибірки	Аномальні значення виявлено та видалено
5	Побудова регресійних моделей	Моделі побудовано, коефіцієнти b_0, b_1 та b_2 розраховано
6	Розрахунок довірчих інтервалів та інтервалів прогнозування	Інтервали розраховано коректно, успішно визначено межі
7	Оцінювання якості побудованих моделей	Показники якості R^2, $MMRE$ та $PRED(0.25)$ повністю відповідають очікуваним значенням
8	Оцінювання якості застосунків	Класифікація оцінок відбувається успішно, застосунки оцінюються належним чином
9	Збереження результатів у форматі CSV	Нормалізовані та очищені дані збережено в зазначеному форматі, структура файлу відповідає вимогам
10	Обробка некоректних даних	В інтерфейсі користувача успішно відображається відповідне попередження або повідомлення про помилку

Отримання очікуваного результату для кожної дії переліку з таблиці Д.1 означатиме успішне підтвердження виконання всіх функціональних вимог.