МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2022 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: «Регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та розробка програми для її реалізації»

Виконала: студентка групи 6151м
_____ Давлатова Д.Х.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.т.н.,
доцент
(посада, науковий ступень, вчене звання)
_____ Макарова Л.М.
(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«10» жовтня 2022 р.

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

Студенту Давлатовій Діані Хакимжонівні
(Прізвище, ім'я, по батькові)

1. Тема роботи: Регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та розробка програми для її реалізації

Керівник роботи Макарова Лідія Миколаївна, к.т.н., доцент

Затверджені наказом ректора № 798 уч від «28» 09 2022 р.

2. Термін подання роботи: 14.11.2022 р.

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____

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

5. Перелік презентаційних матеріалів _____

Титульний слайд, Актуальність обраної теми, Мета, об'єкт та предмет дослідження, Наукова новизна та практичне значення отриманих результатів, Нелінійна регресійна модель для оцінювання розміру веб-застосунків, Ескізний проект програми, Технічний проект програми, Робочий проект програми, Висновки, Заключний аркуш.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 10.10.2022 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	11.10.2022	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	12.10.2022	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	13.10.2022	НС*
4.	Підготовка розділу з проекту програмного забезпечення	03.11.2022	
5.	Підготовка організаційно-економічного розділу	06.11.2022	
6.	Підготовка розділу з охорони праці	09.11.2022	
7.	Підготовка розділу з охорони навколишнього середовища	11.11.2022	
8.	Підготовка розділу Висновки	12.11.2022	
9.	Оформлення списку використаних джерел та додатків	13.11.2022	
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.)	14.11.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	19.11.2022	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	20.11.2022	
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	28.11.2022	

* - за результатами наукового стажування (НС), яке було з 01.09.2022 по 09.10.2022 р.

Студент _____ Давлатова Д.Х.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

РЕФЕРАТ

Давлатова Діана Хакимжонівна

«Регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2022 р.

Обсяг роботи: 121 стор., 20 табл., 13 рис., 57 використаних джерел, 5 додатків.

Актуальність теми роботи: Проблема отримання ефективної системи оцінювання розміру веб-застосунків – це важливе завдання, що вимагає удосконалення існуючих методів. Враховуючи те, що всі моделі безпосередньо залежать від мови програмування та її особливостей маємо, що для обраної мови програмування, фреймворку та бібліотеки не було створено відповідної моделі, реалізованої для особливостей мови Python, тому є доцільним створити таку модель.

Таким чином, побудова нелінійних регресійних моделей із використанням нормалізуючих перетворень для підвищення достовірності оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, і створення на їх основі інформаційної технології переробки інформації є актуальною та має практичну цінність.

Мета і завдання дослідження: підвищення достовірності оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework та розробка програми для її реалізації. Для досягнення поставленої мети необхідно вирішити 5 завдань.

Об’єктом досліджень є процес оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

Предметом досліджень є регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

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

Наукова новизна одержаних результатів полягає в удосконаленні нелінійної регресійної моделі для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії. Ця двофакторна нелінійна регресійна модель у порівнянні з побудованою однофакторною нелінійною регресійною моделлю та існуючою однофакторною нелінійною регресійною моделлю з використанням мови Java краща за параметрами якості *MMRE* та *PRED*(0,25).

Практичне значення одержаних результатів. Розроблено програму для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, на основі побудованої регресійної моделі.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 р.).

Публікації. Результати кваліфікаційної роботи висвітлено в 1 науковій праці – тезах конференції.

Ключові слова: *нелінійна регресійна модель, веб-застосунок, оцінювання розміру веб-застосунків, логарифмічне перетворення, Python, Django Rest Framework.*

ABSTRACT

Davlatova Diana

"A regression model to estimate the size of web applications created in Python using the Django Rest Framework and software development for its implementation"

The qualification work for obtaining a master's degree in specialty 121 "Software engineering". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2022

The work contains: 121 pages, 20 tables, 13 figures, 57 references, 5 appendices.

Relevance of the topic: The problem of obtaining an effective system for evaluating the size of web applications is an important task that requires improvement of existing methods. Considering the fact that all models directly depend on the programming language and its features, we have that for the selected programming language, framework and library there was no corresponding model implemented for the features of the Python language, so it is appropriate to create such a model.

Thus, the construction of nonlinear regression models using normalizing transformations to increase the reliability of estimating the size of web applications created in Python using the Django Rest Framework, and creation of the information technology of information processing that based on them, is relevant and has practical value.

The purpose and tasks of the study: increasing the reliability of estimating the size of web applications created in Python using the Django Rest Framework and developing a program for its implementation. To achieve the goal, it is nesessery to solve 5 tasks.

The object of the study is the process of estimating the size of web applications created in Python using the Django Rest Framework.

The subject of the study is a regression model for estimating the size of web applications created in Python using the Django Rest Framework.

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

Scientific novelty of the obtained results lies in the improvement of the nonlinear regression model for estimating the size of web applications created in the Python language using the Django Rest Framework, due to the application of a normalizing transformation based on the decimal logarithm and regression outliers. This two-variable nonlinear regression model, compared with the built one-variable nonlinear regression model and the existing one-variable nonlinear regression model using the Java language, is better in terms of quality parameters *MMRE* and *PRED*(0.25).

The practical significance of the obtained results. A program has been developed for estimating the size of web applications created in Python using the Django Rest Framework, based on a implemented regression model.

Approbation of research results. The main provisions and results of the research presented in the qualification paper were approved at the III All-Ukrainian Scientific and Practical Internet Conference "Information Technologies: Models, Algorithms, Systems" (Mykolaiv, October 26-28, 2022).

Publications. The results of the qualification work are highlighted in 1 scientific work - theses of the conference.

Keywords: *nonlinear regression model, web application, web application size estimation, logarithmic transformation, Python, Django Rest Framework.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МНК – метод найменших квадратів;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

COCOMO – the Constructive Cost Model;

COSMIC – Common Software Measurement International Consortium;

FSM – Functional Size Measurement;

LOC – a number of lines of code;

MMRE – a mean magnitude of relative error;

MRE – a magnitude of relative error;

PRED – percentage of prediction;

UML – Unified Modeling Languag.

ВСТУП

В умовах нескінченного технологічного розвитку та прагнень людства до швидкого й зручного отримання інформації є постійна необхідність у створенні нових продуктів, програмних забезпечень (ПЗ), мобільних додатків та веб-застосунків.

Досить часто компанії конкурують між собою, кожен має за мету якомога швидше та якісніше виконати поставлені задачі, а також створити щось нове раніше за інших. Чим більше у компанії та розробників досвіду, тим більше вони мають усвідомлення в тому, що важливо підходити до питання створення програмного продукту з аналітичної та математичної точки зору. Тобто компанії прагнуть максимально оптимізувати та автоматизувати як бізнес-процеси, так і роботу своїх співробітників.

Актуальність теми. Успіх або невдача проекту на ранньому етапі розробки, залежить безпосередньо від ефективності оцінювання трудомісткості, яка в свою чергу залежить від розміру програми, тому одним з тих факторів, що впливають на процес розробки ПЗ, є розмір програми. Проблема отримання ефективної системи оцінювання розміру веб-застосунків – це важливе завдання, що вимагає удосконалення існуючих методів [1].

Важливим етапом аналізу складної системи є побудова математичної моделі, що здійснюється на підставі змістовної моделі і складається з вхідних параметрів, внутрішніх параметрів та вихідних параметрів. Однією з умов побудови якісної адекватної моделі є надійність вхідних даних, тому потрібна первинна обробка експериментальних даних із метою виявлення ненадійних вимірів [2]. Виявлення аномальних значень відноситься до проблеми пошуку зразків у даних, які не відповідають очікуваній поведінці.

Під час проведення регресійного аналізу дослідники, зазвичай, віддають перевагу побудові лінійних моделей, які легко оцінюються, і

результати яких чітко інтерпретуються. Однак дані рідко носять абсолютно лінійний характер, тому для їх адекватного опису доводиться вдаватися до побудови нелінійних залежностей [3].

TIOBE Software представила рейтинг найпопулярніших мов програмування на серпень 2022 року [4]. Порівняно з минулим роком Python додав до популярності 3,56%, перемістившись з другого на перше місце з показником 15,42%. Це найвищий показник популярності цієї мови програмування за час існування рейтингу. Найнижчий був зафіксований у 2003 році (0,97%), коли Python посідав 13 місце в рейтингу [5].

Існують моделі оцінювання розміру різноманітних програмних застосунків створених мовою Java, додатків реалізованих на JavaScript та веб-застосунків створених мовою PHP як із використанням фреймворків, наприклад Laravel, так і без них [6-8]. Але враховуючи те, що всі моделі безпосередньо залежать від мови програмування та її особливостей, використавши розроблені моделі для однієї мови, будуть отримані результати, що суттєво відрізнятимуться від тих, що були б отримані з використанням відповідної моделі. Тобто маємо, що для обраної мови програмування, фреймворку та бібліотеки не було створено відповідної моделі, реалізованої для особливостей мови Python, тому є доцільним створити таку модель.

Таким чином, побудова нелінійних регресійних моделей із використанням нормалізуючих перетворень для підвищення достовірності оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, і створення на їх основі інформаційної технології переробки інформації є актуальною та має практичну цінність.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework та розробка програми для її реалізації.

Щоб досягнути поставленої мети потрібно вирішити певні завдання, а саме: виконати аналіз та порівняння моделей оцінювання розміру ПЗ, що вже існують; зібрати проекти з відкритим вихідним кодом зі створення веб-застосунків мовою Python з використанням Django Rest Framework, та обрати ті з них, що можуть бути використані для створення регресійної моделі; визначити необхідні метрики з кожного проекту та перевірити отримані дані на викиди; виконати нормалізацію даних; на основі лінійної моделі та зворотного нормалізуючого перетворення виконати побудову нелінійної регресійної моделі, розробити програму для реалізації побудованої моделі.

Об’єктом досліджень є процес оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

Предметом досліджень є регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

Методи дослідження. Для досягнення поставлених завдань в кваліфікаційній роботі були використані методи теорії ймовірностей, математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в удосконаленні нелінійної регресійної моделі для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії. Ця двофакторна нелінійна регресійна модель у порівнянні з побудованою однофакторною нелінійною регресійною моделлю та існуючою однофакторною нелінійною регресійною моделлю з використанням мови Java краща за параметрами якості *MMRE* та *PRED*(0,25).

Практичне значення одержаних результатів. В межах кваліфікаційної роботи було розроблено програму для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest

Framework, що дозволила автоматизувати виконання розрахунків для прогнозування кількості рядків коду у зручному для користувача інтерфейсі.

Особистий внесок здобувача. У праці, яка опублікована у співавторстві з керівником роботи [9], здобувачеві належить розробка двофакторної нелінійної регресійної моделі для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 р.).

Публікації. Результати роботи висвітлено у тезах III Всеукраїнської науково-практичної інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи».

1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка веб-додатків, як і світ, у якому ми живемо, базується на даних. Простіше кажучи, аналіз даних – це найоптимальніший спосіб оцінити ефективність програми, зрозуміти, що потрібно покращити, і зробити наступний крок у розвитку бізнесу. Для того, щоб мати змогу передати у просторі та часі чи проаналізувати певні процеси або дані, що були досліджені, необхідно правильно та зручно відобразити результати дослідження.

1.1 Обґрунтування вибору мови програмування, фреймворку та бібліотеки для проведення аналізу розроблюваних веб-застосунків

Нині існує багато мов програмування, кожна з яких має своє спрямування, переваги та недоліки, а також розроблена для вирішення конкретних питань та досягнення певних цілей.

Для проведення аналізу та виконання кваліфікаційної роботи було обрано мову програмування Python [10-12]. Вона поширена серед багатьох відомих компаній і є багатофункціональною мовою, що швидко розвивається, легко масштабується та має дуже зручний, логічний синтаксис, широку та всебічну підтримку від ком’юніті, велику базу фреймворків та готових бібліотек для вирішення найрізноманітніших задач.

Проаналізувавши існуючі фреймворки для обраної мови, було обрано високорівневий та безкоштовний веб-фреймворк Django, що має доступ до вихідного коду, дозволяє легко та швидко створювати безпечні і підтримувані веб-сайти, завдяки чому можна повністю зосередитися на написанні веб-застосунку без необхідності винаходити велосипед [13, 14].

Django має зростаючу і дуже активну спільноту, зручну та велику документацію, безліч варіантів як безкоштовної, так і платної підтримки, а також наступні особливості: контроль версій баз даних, широкий вибір готових бібліотек, підтримка аутентифікації, URL-маршрутизація, підтримка ORM, що дозволяє зв'язати бази даних з концепціями ООП, використання шаблонізаторів.

Структура фреймворку Django пов’язана з особливостями мови програмування Python. Творці реалізували в Django патерн MVC, що має застосовання і в останній версії фреймворку [15]. Завдяки архітектурі MVC розробник має можливість розмежованого доступу до візуального представлення застосунку та його бізнес-логіки.

Django Rest Framework – це бібліотека, яка працює зі стандартними моделями Django для створення гнучкого і потужного API для проекту, тобто REST – це система правил по якій створюються API, а DRF – інструмент для швидкого і надійного створення API [16]. Використання цього фреймворку дозволяє легко стандартизувати запити до бази даних і одночасно створювати RESTful WEB API сайту [17].

1.2 Існуючі моделі оцінювання розміру програмного забезпечення

Щоб спроектувати щось нове та управляти ним важливо в першу чергу сформувати образ цього нового та виконати аналіз, щоб зрозуміти як управління може вплинути і до чого приведе. У зв'язку з цим у пізнавальній та практичній діяльності людини велику, якщо не провідну, роль відіграють моделі та моделювання. Особливо незамінне моделювання під час роботи зі складними об'єктами. Усе це робить моделювання найважливішим інструментом системного аналізу.

Модель у широкому розумінні – це образ (зокрема умовний чи уявний) будь-якого об'єкта чи системи об'єктів, що використовується за певних умов

як «заступник/представник». Модель – це спрощена копія об'єкта, яка відтворює цікаві для нас властивості та характеристики вихідного об'єкта або об'єкта проектування [18].

З кінця сімдесятих років дослідження щодо розробки ПЗ та оцінювання вартості були численними та різноманітними. Тема все ще дуже жива, про що свідчать численні роботи, наявні в літературі [19-21].

Дослідники ретельно досліджували цю тему як підходів до оцінювання (регресія, аналогія, експертне судження, функціональні точки, моделювання тощо), так і дослідницького підходу (теорія, опитування, експеримент, тематичне дослідження, моделювання тощо). Дослідження проводяться як у промисловому, так і в академічному контексті. Найбільш часто використовуваний підхід до оцінювання заснований на регресії, де модель COCOMO є найвідомішою моделлю [22].

Що стосується валідації методів оцінювання, переважна більшість дослідницьких підходів базується на використанні історичних даних. Крім того, більшість досліджень стосується промислового контексту. Звужуючи тему до веб-додатків, одним із перших дослідників, які ввели метрику розміру для вимірювання веб-додатків, був Тім Брей шляхом статистичного аналізу основних характеристик основних веб-сайтів у 1995 році. Спочатку моделі, що використовувалися для оцінювання веб-зусиль, були такими ж, як ті, що використовувалися для загальних програмних додатків.

Одним із перших науковців, які представили метод, спеціально розроблений для Інтернету, був Рейфер через метрику WO та модель WEBMO. Пізніше ця модель була використана іншими дослідниками для порівняння різних моделей оцінювання, але з різними результатами, іноді не схожими на інші [23, 24].

Багато дослідницьких робіт з оцінювання веб-зусиль також були проведені Мендесом та його співробітниками.

Однак наразі не існує моделі, здатної адекватно оцінити розмір веб-додатку.

1.3 Метрики програмного забезпечення для оцінювання розміру

Метрики додатків – це вікно ефективності компаній, а також можливість завжди тримати руку на пульсі актуальних тенденцій. Метрики самі собою не є цінністю, але їх поєднання може вказати на правильний напрямок для безперервної оптимізації стратегії створення програми [25].

Оцінювання розміру ПЗ розглядається як фундаментальна діяльність щодо завдань управління програмним забезпеченням.

Планування робіт і подальше оцінювання часу та зусиль прогнозуються на основі розміру ПЗ. Кількість рядків коду (LOC) і функціональність ПЗ є двома показниками, які часто використовуються для визначення розміру програми. LOC – це прямий показник, який легко підрахувати і яким легко маніпулювати. Існує багато дискусій щодо використання LOC в оцінюванні розміру ПЗ. Критики базуються на аргументах, що кількість рядків коду залежить від мови програмування та вимагає деталей, які може бути складно подолати до аналізу та до завершення роботи над проектом [26].

FSM (Functional Size Measurement) можна застосовувати на ранній стадії розробки веб-додатків. FSM – це концепція вимірювання розміру ПЗ з урахуванням функціональних вимог, які вимагає клієнт [27]. Функціональний аналіз точок (FPA) був вперше розроблений Альбрехтом для підтримки цієї концепції [28]. Техніку FSM можна використовувати для раннього оцінювання веб-зусиль, і вона не залежить від жодної технології чи мови програмування, і її можна використовувати протягом усього життєвого циклу розробки. Марко проаналізував методи FSM і COSMIC (Common Software Measurement International Consortium) для визначення розміру ПЗ та пояснив, що FSM висвітлив деякі проблеми під час використання FSM для оцінювання розміру веб-сайту [29]. З іншого боку, метод COSMIC був корисним для оцінювання зусиль веб-додатку. Це легко зрозуміти, проста і економічно ефективна реалізація. За допомогою COSMIC можна було отримати певне

точне оцінювання для великих проектів. COSMIC-FFP – це метод вимірювання функціонального розміру ПЗ для оцінювання розміру ПЗ [30].

Розробка веб-додатків стає дуже популярною на основі концептуальних моделей, і ці моделі можна коригувати та переглядати в будь-який час у процесі розробки додатків. Менеджерам з оцінювання може бути корисним детальний аналіз програму перед початком розробки.

Зібрати інформацію про реальні промислові проекти дуже складно. Більшість компаній не передають дані про свої проекти. Технології змінюються настільки швидко, що стало важко створити стандартну модель для оцінювання. Різні компанії використовують різні підходи, характер і розмір проектів різний.

Прогнозування роботи з розробки ПЗ з високою точністю все ще є проблемою для менеджерів з оцінювання. Існують різні методи оцінювання вартості, які можна класифікувати за трьома різними категоріями: експертне оцінювання, алгоритмічні моделі та машинне навчання. У експертному судженні оцінювання проводиться на основі даних і відповідних проектів. Алгоритмічні моделі, різні зусилля та методи оцінювання є його частиною, як COCOMO. Машинне навчання, ця техніка є заміною алгоритмічних моделей і включає нечітку логіку, регресійні дерева. У запропонованому алгоритмі оцінювання зусиль і витрат базується на умовах набору даних. COCOMO потрібно багато даних, щоб зробити оцінювання, але вона забезпечує чіткі результати, хоча експертне оцінювання є швидким прогнозованим підходом до оцінювання.

1.4 Застосування методів регресійного аналізу для оцінювання розміру програмного забезпечення

Щоб передбачити одну змінну на основі іншої, потрібно використовувати розширення лінійної кореляції, що називається лінійною

регресією. Лінійна регресія розширює ідею діаграми розсіювання, яка використовується в кореляції, і додає лінію, яка найкраще «підходить» до даних. Оскільки лінійна регресія є розширенням лінійної кореляції, вона моделює лінійний компонент зв'язку між змінними. Якщо зв'язок не має лінійного компонента, тоді кореляція буде близькою до 0, а лінійна регресія матиме незначну точність прогнозування або взагалі її не матиме.

Багатофакторне лінійне рівняння регресії набуває загального вигляду:

$$\hat{Y} = b_0 + b_1X_1 + b_2X_2 + \cdots + b_kX_k, \quad (1.1)$$

де: $\hat{Y}$ – прогнозоване значення залежної змінної;

b_0 – точка перетину Y, тобто прогнозоване значення Y, коли всі значення X дорівнюють 0;

$b_0, b_1, b_2, \ldots, b_k$ – параметри лінійного регресійного рівняння;

$X_0, X_1, X_2, \ldots, X_k$ – конкретні значення незалежних змінних.

Якщо використовується лише один аргумент, то маємо однофакторне лінійне регресійне рівняння:

$$\hat{Y} = b_0 + b_1X_1. \quad (1.2)$$

Для перевірки якості та адекватності лінійної регресії необхідно обчислити: коефіцієнт детермінації R^2, середню величину відносної похибки *MMRE*, рівень прогнозування *PRED*.

Коефіцієнт детермінації R^2 визначається за наступною формулою:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}, \quad (1.3)$$

де: Y_i – емпіричне значення;

$\hat{Y}_i$ – розраховане значення;

$\bar{Y}$ – середнє значення.

Огляд літератури показав, що *MMRE* є найбільш широко використовуваним і разом з *PRED* базується на тому самому базовому значенні відносної похибки *MRE* без одиниці вимірювання, що визначається наступним чином:

$$MRE_i = \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right|. \tag{1.4}$$

Стверджується, що *MRE* є корисним, оскільки він не накладає надмірних нівечень на великі проекти та не має одиниць, тобто не залежить від масштабу. В свою чергу *MMRE* визначається як середнє значення *MRE*:

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

Рівень прогнозування *PRED* визначають за формулою:

$$PRED(l) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 1 & if\ MRE_i \leq 0{,}25 \\ 0 & otherwise \end{cases}. \tag{1.6}$$

Зазвичай задовільними є значення *MMRE* – не більше 0,25 та *PRED*(0,25) – не менше 0,75.

Важливим завданням є знаходження викидів, що визначаються як ненормальні значення в наборі даних, які не відповідають регулярному розподілу та можуть значно спотворити будь-яку регресійну модель, тому з викидами потрібно обережно поводитися, щоб отримати правильну вибірку з даних. Зазвичай дані, зібрані в реальному світі, складаються з кількох спостережень за кількома функціями, і багато значень можуть бути розміщені

неправильно або просто бути артефактом обробки даних. Якою б не була причина викиду, викиди необхідно проаналізувати та переконатися, що вони справжні. Якщо викиди реальні, можна взяти ці викиди або просто відкинути їх, щоб створити кращу модель регресії.

Щоб оцінити вплив видалення викидів на лінії регресії необхідно визначити, додатним чи від'ємним є нахил лінії регресії до видалення викиду та визначити вплив на коефіцієнт кореляції *r*. Видалення викиду робить кореляцію сильнішою. Якщо нахил був позитивним, видалення викиду збільшить значення *r*, наблизивши його до 1. Якщо нахил був від'ємним, видалення викиду зменшить значення *r*, наблизивши його до -1.

Тобто викид – це точка даних, яка не відповідає решті даних. Викид може бути вищим або нижчим, ніж очікувалося, або зміщеним більше вправо чи вліво, ніж очікувалося. Викиди можуть впливати на лінії регресії, роблячи лінії регресії менш точними при прогнозуванні інших даних.

Лінія регресії – це пряма лінія, яка наближає дані. З лінійними даними часто легше працювати, тому лінії регресії часто використовуються для прогнозування та аналізу даних у спрощений спосіб.

Для перевірки даних на викиди використовується квадрат відстані Махаланобіса [31, 32], що розраховується за формулою:

$$d_i^2 = (Z_i - \bar{Z})^T S_N^{-1} (Z_i - \bar{Z}), \tag{1.7}$$

де: Z_i – i-ий елемент, що має координати Z_{x_i}, Z_{y_i};

N – кількість елементів у вибірці;

$\bar{Z}$ – вектор вибіркових середніх;

S_N – коваріаційна матриця, що розраховується за формулою (1.8):

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \bar{Z})\,(Z_i - \bar{Z})^T. \tag{1.8}$$

Після обчислення d_i^2 можна застосовувати або критерій Пірсона χ^2 або критерій Фішера F. Ті дані, для яких значення квадрата відстані Махаланобіса більше, ніж квантиль розподілу χ^2, вважаються викидами, і ці значення відкидаються, а розрахунки проводяться наново допоки всі значення не будуть задовольняти вимогам.

Коефіцієнт кореляції r – це значення від -1 до 1, яке вказує на те, наскільки сильний лінійний зв’язок між двома змінними. Чим ближче до 1 значення r, тим сильнішим є позитивний лінійний зв’язок. Чим ближче до -1 значення r, тим сильнішим є негативний лінійний зв’язок. Якщо r=0, це означає відсутність лінійного зв’язку між змінними.

Часто є необхідність визначити не тільки кореляційне співвідношення між характеристиками, що вивчаються, а й встановити певну обумовленість між ними, представивши виявлений зв'язок у суворій аналітичній формі.

Результат дослідження – експериментальна залежність впливу будь-якого фактора на зміну параметра, що вивчається, може бути не тільки представлений у вигляді графіка, але і описаний математично з використанням апроксимуючого виразу. Дослідження такої ситуації є завданням регресійного аналізу, який дає передбачення (прогнозування) однієї змінної на підставі іншої [33].

У регресійному аналізі одне з досліджуваних показників, саме змінна яка прогнозується, є функцією і позначається "y", а друга називається аргументом і використовується безпосередньо для прогнозування, позначається "x" , тобто. маємо чіткий розподіл ролей. Якщо ж маємо декілька аргументів, то отримуємо більш складну, а саме багатофакторну регресійну модель.

Таким чином, якщо в кореляції дається спроба визначення наявності чи відсутності зв'язку між величинами, то метою регресійного аналізу є визначення не тільки сили та характеру зв'язку, а й кількісний вплив однієї змінної на аргумент.

Багатофакторна лінійна регресійна модель в загальному випадку має вигляд:

$$Z_Y = \hat{Z}_Y + \varepsilon = b_0 + b_1 Z_{X_1} + b_2 Z_{X_2} + \cdots + b_k Z_{X_k} + \varepsilon, \qquad (1.9)$$

де ε – це випадкова похибка, розподіл якої в загальному випадку залежить від незалежних змінних, але математичне очікування якої рівне нулю.

Нелінійна регресійна модель будується за перетвореним у вигляді десяткового логарифму лінійним рівнянням (1.9):

$$Y = 10^{\varepsilon + b_0} \mathrm{X}_1^{b_1} \mathrm{X}_2^{b_2} \cdot \ldots \cdot \mathrm{X}_k^{b_k}. \qquad (1.10)$$

Щоб визначити коефіцієнти рівняння регресії b застосовують різні методи (графічний, метод середніх), проте найбільшого поширення набув метод найменших квадратів (МНК). Суть МНК полягає в мінімізації суми квадратів відхилень значень пояснюваної змінної, що спостерігаються, від її розрахункових значень, тобто. необхідно знайти мінімум квадратичної функції двох змінних b_0 та b_1:

$$F(b_0, b_1) = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^{N} (Y_i - b_0 - b_1 X_i)^2. \qquad (1.11)$$

Необхідною умовою мінімуму функції двох змінних є рівність нулю похідних за b_0 та b_1, тому маємо формули для обчислення коефіцієнтів парної лінійної регресії:

$$b_0 = \frac{\sum_{i=1}^{N} X_i^2 \sum_{i=1}^{N} Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} X_i Y_i}{N \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2},$$

$$b_1 = \frac{N\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{N\sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2}. \quad (1.12)$$

Нехай обговорюється деяка залежність $y = f(x)$, яка відображає якийсь процес, що має плавну течію, і тому всі параметри системи змінюються поступово, без стрибків. У цих випадках експериментальні точки, нанесені на графіку, повинні б укладатися на деяку плавну криву (в окремому випадку, пряму). Однак на практиці певний розкид експериментальних точок завжди спостерігається, що пов'язано з мінливістю (помилками) вимірювань, що реєструються. Зрозуміло, що такого розкиду вдалося б уникнути, якби результати вимірювань виявилися абсолютно вільними від помилок, і тоді точки, що відповідають цим результатам, суворо лягали б на відповідну плавну криву, або пряму лінію. Тому всі процеси, які мають свідомо плавну течію, прийнято зображати також плавними кривими, проводячи їх не через точки, а так, щоб крива проходила якомога ближче до всіх точок на графіку.

За наступними формулами визначаються відповідно довірчий інтервал та інтервал передбачення лінійної регресії:

$$\hat{Z}_y \pm t_{\alpha/2,\nu} S_{Z_Y} \left\{\frac{1}{N} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+)\right\}^{1/2}, \quad (1.13)$$

$$\hat{Z}_y \pm t_{\alpha/2,\nu} S_{Z_Y} \left\{1 + \frac{1}{N} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+)\right\}^{1/2}, \quad (1.14)$$

де: $S_{Z_Y}^2 = \frac{1}{\nu}\sum_{i=1}^{N} (Z_{Y_i} - \hat{Z}_{Y_i})^2$,$\nu = N - k - 1$;

$t_{\alpha/2,\nu}$ – квантіль t-розподілу Стьюдента з ν степенями вільності та $\alpha/2$ рівнем значущості;

$(Z_X^+)^T Z_X^+$ – $k \times k$ матриця:

$$(Z_X^+)^T Z_X^+ = \begin{pmatrix} S_{Z_1 Z_1} & S_{Z_1 Z_2} & \cdots & S_{Z_1 Z_k} \\ S_{Z_1 Z_2} & S_{Z_2 Z_2} & \cdots & S_{Z_2 Z_k} \\ \cdots & \cdots & \cdots & \cdots \\ S_{Z_1 Z_k} & S_{Z_2 Z_k} & \cdots & S_{Z_k Z_k} \end{pmatrix}, \quad (1.15)$$

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

Таким чином отримаємо більш достовірну математичну модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

2 ПОБУДОВА РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ PYTHON З ВИКОРИСТАННЯМ DJANGO REST FRAMEWORK

2.1 Двофакторна нелінійна регресійна модель для оцінювання розміру програмного забезпечення

Нормальний (або гаусівський) розподіл заклав основу для незліченних статистичних методологічних основ, не останньою з яких є класична лінійна регресія. Перетворення даних таким чином, щоб можна було застосовувати методи, які вимагають нормальності, обійшло, мабуть, більш складну проблему розробки статистичних методів, які могли б врахувати ненормальні дані. Практика перетворення даних, щоб вони виглядали нормально за допомогою певного нормалізуючого перетворення широко використовується через її простоту. Хоча, можливо, це не найелегантніше рішення проблеми, часто ця техніка працює добре як швидке та прагматичне рішення.

Щоб мати змогу застосовувати необхідні тести до даних важливо розуміти чи відповідають дані нормальному розподілу ймовірностей. Якщо дані відповідають нормальному розподілу ймовірностей, можна застосувати параметричні тести – порівнювати значення даних із розподілом, який має відому форму та може бути оцінений на основі значення параметрів: *z*-тест для великих вибірок, *t*-тест для малих ($n<30$) вибірок. Якщо дані не відповідають відомому розподілу ймовірностей, будь-яке порівняння даних із параметрами сукупності може бути недійсним.

Якщо відомо, що дані дотримуються нормального розподілу, можливо, вони виглядають ненормально на графіку, оскільки вибірок занадто мало. З іншого боку, якщо мається достатньо вибірок для представлення справжньої сукупності, можна підібрати різні типи розподілів, щоб виконати кращий опис даних. Коли відомо, що вибіркові дані відповідають логарифмічному

нормальному розподілу, можна рухатися вперед, використовуючи методи, які припускають логарифмічний нормальний розподіл. Дані фактично розподіляються нормально, але, можливо, їм знадобиться перетворення, щоб виявити їх нормальність. Наприклад, логарифмічний нормальний розподіл стає нормальним розподілом після того, як до нього застосовують логарифм. Ці типи перетворень – зміна масштабу розподілу за допомогою експонент або логарифму – називаються степеневими перетвореннями.

Для нормалізації рядів даних було розглянуто перетворення Бокса-Кокса, перетворення Джонсона та логарифмічне перетворення.

Перетворення Бокса-Кокса є найпопулярнішим методом у сімействі степеневих перетворень [34]. Перетворення Бокса-Кокса – це тип степеневого перетворення для перетворення незвичайних даних у звичайні шляхом піднесення розподілу до степеня λ. Трансформація Бокса-Кокса – це статистичний метод, який, як відомо, має виправний вплив на сильно спотворені дані. По суті, це просто підвищення розподілу до степеня λ, щоб перетворити ненормальний розподіл у нормальний. Параметр λ для Бокса-Кокса має діапазон -5 < λ < 5. Винятком із цього правила є випадки, коли параметр λ дорівнює 0 – log буде взято до розподілу – log(*Y*). Перетворення Бокса-Кокса не працює, якщо дані менші за 0:

$$x(\lambda) = \begin{cases} \dfrac{x^{\lambda} - 1}{\lambda}, \text{якщо } \lambda \neq 0; \\ \ln(x), \text{якщо } \lambda = 0. \end{cases}$$

Перетворення Джонсона – це перетворення, що є дуже потужним і його можна використовувати з даними, які містять нульові та від'ємні значення, але воно складніше й надає лише загальну статистику можливостей:

$$z = \gamma + \eta q(x, \varphi, \lambda); \quad \eta > 0; \quad -\infty < \varphi < \infty; \quad \lambda > 0; -\infty < \gamma < \infty,$$

де: q – довільна функція;

$\gamma, \eta, \varphi, \lambda$ – параметри розподілу Джонсона;

z – нормально розподілена випадкова величина.

Якщо перетворення Джонсона є ефективним і нормальний розподіл добре підходить для перетворених даних, точки на графіку для перетворених даних повинні точно відповідати встановленій лінії нормального розподілу. Відхилення від прямої лінії вказують на те, що підгонка неприйнятна і що перетворення Джонсона неефективне.

У логарифмічному перетворенні кожна змінна x замінюється на $\log(x)$ з основою 10, основою 2 або натуральним логарифмом і має зворотне до нього перетворення:

$$z = \lg(x), \tag{2.1}$$

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

Нормалізуючі перетворення не завжди є легкою панацеєю. З моменту свого впровадження трансформація Бокса-Кокса мала відомі недоліки. Перетворення Джонсона намагається мінімізувати розбіжність Кулбака-Лейблера між нормальним розподілом і розподілом перетворених даних [35]. Таким чином, жоден метод не гарантує нормалізації вектора, і жоден метод не буде найоптимальнішим. Це проблематично, особливо в ситуаціях без попереднього знання про форму розподілу, наприклад, у налаштуваннях високовимірної регресії, де нормалізація має бути автоматичною.

Спотвореність даних впливає на їх оцінювання, тому дані зі значним відхиленням від нормальності можуть бути більш придатними для аналізу після відповідного перетворення. У той час як нормальність може не бути суворо необхідною, викиди та серйозні порушення асиметрії та ексцесу можуть вплинути на структуру діаграм, а отже, і на результати аналізу.

Отже, було обрано десятковий логарифм в якості нормалізуючого перетворення для побудови регресійної моделі для оцінювання розміру веб-

застосунків, що створюються мовою Python з використанням Django Rest Framework.

2.2 Побудова двофакторної нелінійної регресійної моделі для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework

Щоб виконати оцінювання розміру веб-затосунків на GitHub [36], веб-сайті та хмарній службі, що допомагає розробникам зберігати свій код і керувати ним, а також відстежувати та контролювати зміни в коді, було зібрано дані з метрик 71 проекту з відкритим кодом на Python з використанням Django Rest Framework.

Процес збору метрик проектів написаних мовою Python є досить не простою задачею, оскільки фактично немає такого ПЗ, що надає повну інформацію про проект. На рисунку 2.1 зображено розширення до PyCharm Statistic [37, 38] завдяки якому було зібрано наступні метрики, фрагмент яких наведено в таблиці 2.1: кількість рядків чистого коду без коментарів та порожніх рядків Source code lines; Count – кількість файлів з розширенням .py; кількість рядків з коментарями Comment lines; кількість порожніх рядків Blank lines та загальна кількість рядків Total lines, KLOC.

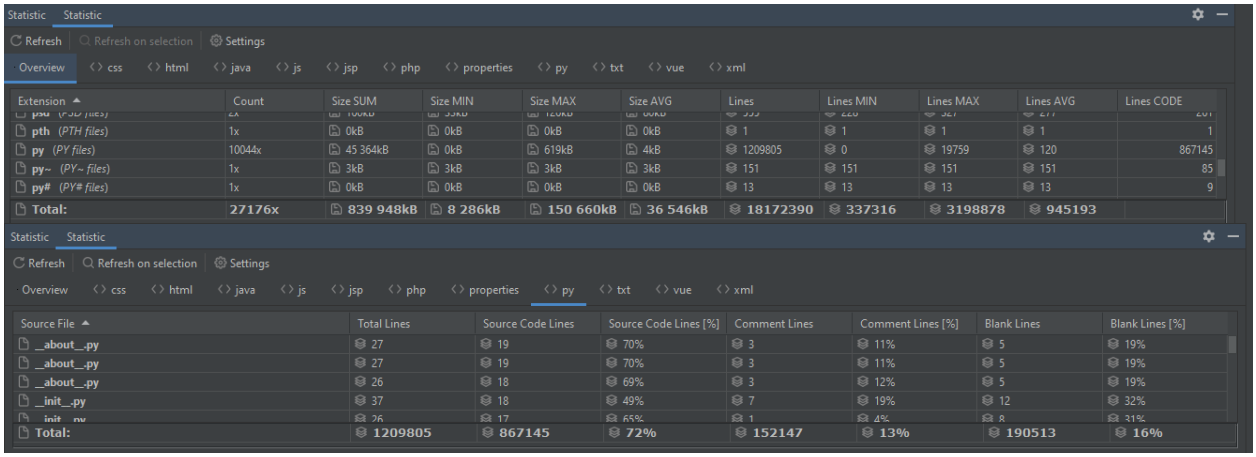

Рисунок 2.1 – Збір метрик проекту за допомогою плагіну Statistic

Також було додатково написано скрипт, що підраховує кількість класів Classes та методів Methods написаних мовою Python в кожному з обраних проектів.

Таблиця 2.1 – Фрагмент даних з метрик веб-застосунків з відкритим кодом на Python з використанням Django Rest Framework

№	Назва проекту	Source code lines	Classes	Methods	Count	Comment lines	Blank lines	Total lines, KLOC
…								
10	django-vue-template-master	152	2	7	15	64	79	0,295
11	docker-tutorial-master	221	5	15	12	61	73	0,355
12	django-rest-framework-social-oauth2-master	363	25	24	11	98	112	0,573
13	VueDjangoAntdProBookShop-master	20124	473	1987	397	3796	4837	28,757
14	django-rest-framework-json-api-master	9974	344	465	90	1351	2064	13,389
15	django-rest-framework-gis-master	3945	90	170	30	401	631	4,977
16	everbug-master	679	16	96	25	37	210	0,926
17	cmdb-master	2453	67	146	98	352	588	3,393
18	django-rest-framework-filters-master	2806	313	236	40	663	1034	4,503
19	dj-rest-auth-master	2781	87	182	33	494	762	4,037
20	django-rest-framework-tutorial-master	334	12	26	20	88	107	0,529
21	bag-of-holding-master	4742	131	216	45	185	1254	6,181
22	modern-django-master	509	4	31	47	165	214	0,888
23	OnlineMooc-2	25010	723	592	46	397	1894	27,301
24	django-rest-framework-docs-master	794	74	68	33	104	329	1,227
25 …	osf.io-develop	230414	1974	17285	2119	26551	49222	306,187

Після виконання збору даних було вирішено побудувати двофакторну нелінійну регресійну модель, тому в якості залежної змінної *Y* було обрано значення загальної кількості рядків коду проекту, в тисячах рядків коду (KLOC), а в якості аргументів виступають значення кількості класів X_1 та кількості методів X_2 проекту.

Необхідно виконати первинний аналіз даних, а саме переконатися, що вибірка не містить викидів, але перш за все важливо виконати перевірку предикторів на наявність мультиколінеарності. У регресійній моделі лінійна залежність між двома або більше незалежними змінними демонструє присутність мультиколінеарності, що безперечно є негативним явищем і не дозволяє здійснити оцінювання окремого впливу кожного фактору на залежну змінну, оскільки маємо пов'язаність факторів між собою або високий ступінь кореляції.

Серед майбутніх предикторів в моделі множинної лінійної регресії наявність мультиколінеарності визначимо за коефіцієнтами впливу дисперсії (VIFs). Для лінійної моделі множинної регресії з k-предикторами X_i, i=1,...,k, VIFs – це діагональні елементи оберненої кореляційної матриці kxk k-предикторів. Якщо значення VIFs більше за 10, то дані мають проблеми з мультиколінеарністю, а у разі, коли значення VIFs знаходяться у межах від 1 до 5, то мультиколінеарності немає [39].

Обернена кореляційна матриця представлена нижче:

$$\begin{pmatrix} 3{,}10 & -2{,}55 \\ -2{,}55 & 3{,}10 \end{pmatrix}.$$

Виконавши перевірку впевнились, що обрані метрики, а саме: кількість класів X_1 та кількість методів X_2 проекту не мультиколінеарні, тому можемо використовувати їх в якості двох незалежних змінних.

Наступним кроком необхідно перевірити вихідні дані на нормальність. Перевірку виконуватимемо за допомогою критерію Пірсона (критерій χ^2):

$$\chi^2 = \sum_{j=1}^{m} \frac{\left(n_j - \mathrm{n}p_j\right)^2}{\mathrm{n}p_j}, \tag{2.3}$$

де: m – кількість підінтервалів, на які розбивається інтервал [x_{min}, x_{max}];

x_{min} та x_{max} – відповідні мінімальне та максимальне значення випадкової величини X;

n_j – кількість значень випадкової величини, які потрапляють у j-ий підінтервал;

p_j – ймовірність того, що значення випадкової величини X потрапляють у j-ий підінтервал.

За формулою Старджеса можна визначити кількість підінтервалів, на які розбивається інтервал [x_{min}, x_{max}]: $m = \log_2 n + 1 = 3{,}3 \lg n + 1$.

Усі три змінні Y, X_1 та X_2 не підпорядковуються нормальному закону оскільки χ^2 розрахунковий для них суттєво перевищує χ^2 критичний при рівні значимості $\alpha = 0{,}05$ та кількості ступенів вільності $\nu = 4$, що визначається як $\nu = m - k - 1$, де k – кількість параметрів, від яких залежить закон розподілу. Тому виконуємо нормалізацію вихідних даних за допомогою обраного нормалізуючого перетворення на основі десяткового логарифму за формулою (2.1).

В таблиці 2.2 наведені вихідні дані, дані отримані після нормалізації, а також значення квадрату відстані Махаланобіса, що були розраховані за формулою (1.7), та додатковий *TS* (Test Statictic) для розрахунку критерію Фішера (фрагмент даних).

Таблиця 2.2 – Фрагмент даних обробки та дослідження вихідних емпіричних даних

№	Y	X_1	X_2	Z_y	Z_{x1}	Z_{x2}	d_i^2	TS для d_i^2
…								
10	0,295	2	7	-0,5302	0,3010	0,8451	**14,42**	**4,60**
11	0,355	5	15	-0,4498	0,6990	1,1761	**8,24**	2,63
12	0,573	25	24	-0,2418	1,3979	1,3802	6,43	2,05
13	28,757	473	1987	1,4587	2,6749	3,2982	**8,05**	2,57
14	13,389	344	465	1,1267	2,5366	2,6675	1,67	0,53
15	4,977	90	170	0,6970	1,9542	2,2304	0,21	0,07
16	0,926	16	96	-0,0334	1,2041	1,9823	1,38	0,44
17	3,393	67	146	0,5306	1,8261	2,1644	0,12	0,04
18	4,503	313	236	0,6535	2,4955	2,3729	1,38	0,44
19	4,037	87	182	0,6061	1,9395	2,2601	0,08	0,02
20	0,529	12	26	-0,2765	1,0792	1,4150	5,08	1,62
21	6,181	131	216	0,7911	2,1173	2,3345	0,34	0,11
22	0,888	4	31	-0,0516	0,6021	1,4914	4,76	1,52
23	27,301	723	592	1,4362	2,8591	2,7723	3,45	1,10
24	1,227	74	68	0,0888	1,8692	1,8325	2,28	0,73
25	306,187	1974	17285	2,4860	3,2953	4,2377	**29,23**	**9,33**
…								

При $m = 3$, $\alpha = 0{,}05$ та кількості значень $n = 71$ отримали критичні значення критерію Пірсона та Фішера відповідно $\chi^2_{\text{кр}} = 7{,}81$ та $F_{\text{кр}} = 2{,}74$.

Використовуючи критерій Пірсона маємо 9 викидів, в той час як використавши критерій Фішера отримали 6 викидів, тож було прийнято рішення, що отримані викиди у кількості 9 штук видаляємо із вибірки загального набору даних і повторно виконуємо перевірку на викиди для набору даних із 62 проектів. В результаті виконання другої ітерації викидів виявлено не було, тому наступні розрахунки ведемо при $n = 62$.

За допомогою методу найменших квадратів знаходимо коефіцієнти b_0=-1,6796, b_1=0,0534, b_2=0,9571 та за допомогою рівняння (1.9) будуємо лінійну регресійну модель для нормалізованих даних:

$$Z_Y = -1{,}6796 + 0{,}0534 Z_{X_1} + 0{,}9571 Z_{X_2} + \varepsilon.$$

Оскільки було виконано побудову лінійної регресії для нормалізованих даних необхідно впевнитись, що залишки лінійної регресійної моделі підкорюються нормальному закону розподілу. Перевірку залишків ε на нормальність (рисунок 2.2) також проводимо на основі критерію χ^2 Пірсона.

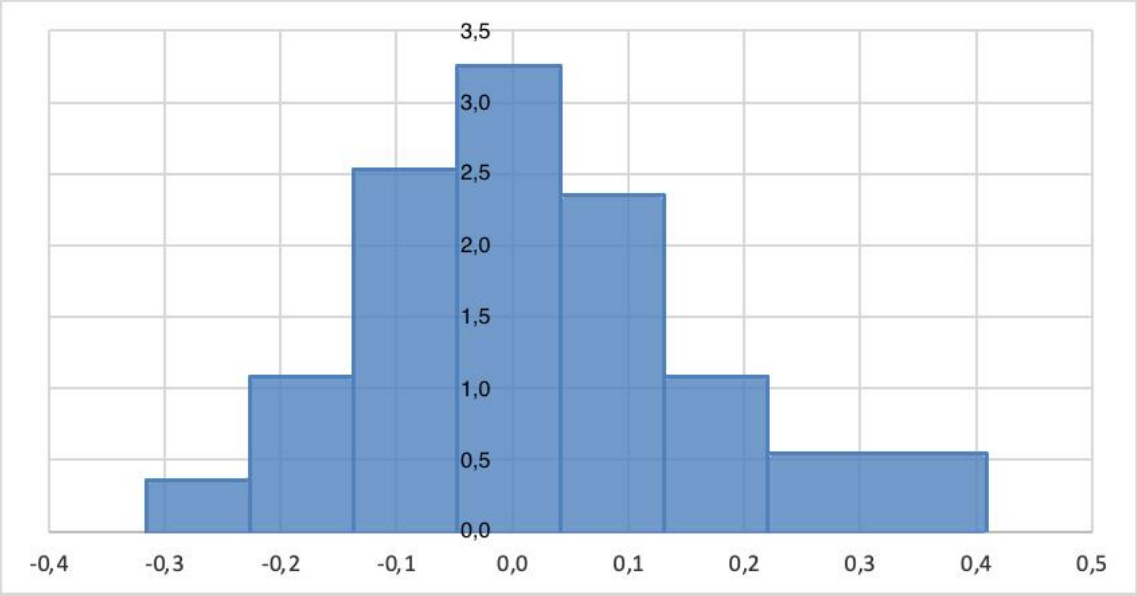

Рисунок 2.2 – Перевірка розподілу частот для залишків ε

Розрахували критичне та розрахункове значення критерію Пірсона: $\chi^2_{\text{кр}} = 9{,}49$ та $\chi^2 = 0{,}22$ відповідно, з чого можемо зробити висновок, що залишки розподілені за нормальним законом, тобто можемо будувати дану лінійну регресійну модель і за рівнянням (1.10) можемо перейти до нелінійної регресійної моделі:

$$Y = 10^{\varepsilon - 1{,}6796} X_1^{0{,}0534} X_2^{0{,}9571}.$$

Використавши формули (1.3) – (1.6) для нелінійної регресійної моделі було отримано значення коефіцієнту детермінації R^2 = 0,8936, середньої величини відносної похибки *MMRE* = 0,2342 і рівня прогнозування *PRED*(0,25) = 0,6290.

Далі виконаємо побудову довірчого інтервалу та інтервалу передбачення для нелінійної регресії, використовуючи отриману раніше лінійну регресію та відповідні розрахункові інтервали для лінійної регресії за формулами (1.13) – (1.15).

Фрагмент розрахованих границь інтервалу передбачення наведено в таблиці 2.3.

Таблиця 2.3 – Фрагмент розрахованих границь інтервалу передбачення для використовуваних даних

№	Y	X_1	X_2	$\hat{Y}$	Інтервал передбачення	
					Нижня границя	Верхня границя
…						
12	0,573	25	24	0,5200	0,2865	0,9437
14	13,389	344	465	10,2041	5,6213	18,5229
15	4,977	90	170	3,6262	2,0080	6,5485
16	**0,926**	16	96	1,9138	**1,0534**	**3,4768**
17	3,393	67	146	3,0857	1,7147	5,5528
18	4,503	313	236	5,3050	2,9226	9,6296
19	4,037	87	182	3,8638	2,1258	7,0230
20	0,529	12	26	0,5399	0,2974	0,9800
21	6,181	131	216	4,6526	2,5889	8,3613
22	0,888	4	31	0,6025	0,3256	1,1147
23	**27,301**	723	592	13,3770	**7,3500**	**24,3462**
24	1,227	74	68	1,4930	0,8225	2,7101
26	34,607	225	1490	30,4061	16,5606	55,8272
27	8,457	417	532	11,7269	6,4795	21,2242
28	1,226	14	58	1,1731	0,6470	2,1271
29	6,886	41	376	7,4331	4,0569	13,6191
…						

Оскільки для точок 16 та 23 вихідні значення Y виходять за розраховані границі інтервалу передбачення, вважаємо ці точки викидами та видалимо їх із загальної вибірки даних. Таким чином, в остаточному скорегованому наборі даних маємо $n = 60$.

Для цих даних знову виконаємо пошук викидів з використанням квадрату відстані Махаланобіса та оскільки викидів знайдено не було, за допомогою методу найменших квадратів знаходимо коефіцієнти b_0=-1,6417, b_1=-0,0090, b_2=0,9902 і будуємо лінійну регресійну модель:

$$Z_Y = -1{,}6417 - 0{,}0090 Z_{X_1} + 0{,}9902 Z_{X_2} + \varepsilon.$$

На основі критерію χ^2 Пірсона знову впевнились, що залишки ε лінійної регресійної моделі підкорюються нормальному закону розподілу: $\chi^2_{\text{кр}} = 9{,}49$ та $\chi^2 = 0{,}26$, і за рівнянням (1.10) перейдемо до нелінійної регресійної моделі:

$$Y = 10^{\varepsilon - 1{,}6417} X_1^{-0{,}0090} X_2^{0{,}9902}.$$

Використавши формули (1.3) – (1.6) для остаточної нелінійної регресійної моделі було отримано значення коефіцієнту детермінації $R^2 = 0{,}9115$, середньої величини відносної похибки *MMRE* = 0,2140 і рівня прогнозування *PRED*(0,25) = 0,6333.

Побудуємо довірчий інтервал та інтервал передбачення, та оскільки всі значення *Y* знаходяться в межах розрахованих границь інтервалу передбачення, можемо вважати, що побудовано удосконалену двофакторну нелінійну регресійну модель на основі логарифмічного перетворення. Виходячи з того, що було отримано кращі параметри якості для удосконаленої моделі, можемо дійти висновку, що додаткове видалення викидів за межами інтервалу передбачення призводить до покращення якості моделі.

Оскільки для мови програмування Python серед опублікованих матеріалів не було знайдено відповідної моделі для порівняння, було прийнято рішення побудувати однофакторну нелінійну регресійну модель, де в якості незалежної змінної було обрано значення кількості класів проекту *X* і в якості залежної змінної – кількість рядків коду *Y* (KLOC), оскільки це найпростіший варіант оцінювання моделі у разі об'єктно-орієнтованої методології розробки. А також було виконано порівняння з опублікованою

нелінійною однофакторною регресійною моделлю побудованою для веб-застосунків з використанням мови Java [40].

Критерії якості побудованих нелінійних регресійних моделей наведено в таблиці 2.4 для розробленої двофакторної та однофакторної нелінійних регресійних моделей для веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та опублікованої однофакторної регресійної моделі побудованої для веб-застосунків з використанням мови Java.

Таблиця 2.4 – Порівняння критеріїв якості побудованих нелінійних регресійних моделей

Критерії якості	Двофакторна нелінійна регресійна модель з використанням мови Python	Однофакторна нелінійна регресійна модель з використанням мови Python	Однофакторна регресійна модель з використанням мови Java
MMRE	0,2140	0,5811	0,9959
PRED(0,25)	0,6333	0,3099	0,0000

Аналізуючи параметри якості усіх побудованих нелінійних регресійних моделей дійшли наступних висновків:

1. За параметрами *MMRE* та *PRED*(0,25) маємо, що отримана двофакторна нелінійна регресійна модель краща ніж побудована однофакторна модель та порівняльна модель для веб-застосунків розроблених мовою Java.

2. Включення в модель другого фактору суттєво покращує результати розрахунків.

3. Можливо логарифмічне перетворення дає не найкращі результати і для нормалізації даних можна було б використати перетворення Бокса-Кокса, перетворення Джонсона або багатовимірні перетворення та незважаючи на те, що значення рівня прогнозування *PRED*(0,25) не дістало до необхідного мінімально допустимого значення 0,75, загалом по іншим параметрам було

отримано гарні результати і отримана нелінійна регресійна модель має високу якість оцінювання.

Отже, двофакторну нелінійну регресійну модель, що розроблена на основі обраних незалежних змінних, а саме значення кількості класів X_1 та кількості методів X_2 проекту, можна використовувати для виконання прогнозу значень результативного показника розміру у KLOC веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, тому на підставі саме цієї моделі можемо виконувати розробку відповідної програми.

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ PYTHON З ВИКОРИСТАННЯМ DJANGO REST FRAMEWORK

3.1 Ескізний проект програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework

Оскільки було обрано об’єктно-орієнтовану методологію проектування для створення відповідної програми було вирішено використовувати засоби UML [41].

Уніфікована мова моделювання (UML) – це мова візуального кодування та корисний інструмент для дизайнерів і програмістів, що може допомогти поєднати візуалізацію зі стандартизацією, щоб досягти вищої якості, кращої відповідності та підвищення продуктивності.

Для фахівців з інформаційних технологій, які розробляють, тестують або аналізують ПЗ, UML може надати ці переваги та підтримати їх роботу, заощадити час і покращити співпрацю між членами команди. UML надає інженерам і розробникам ПЗ спосіб створювати, документувати та візуалізувати програмні системи і хоча вона не є мовою програмування у звичному сенсі, вона може надавати візуальні представлення, які допомагають розробникам ПЗ краще зрозуміти потенційні результати або помилки в програмах.

3.1.1 Подання вимог до програми моделлю варіантів використання

Діаграма варіантів використання (use case diagram) – діаграма, на якій зображуються варіанти використання проектованої системи, укладені в межу

суб'єкта, та зовнішні актори, а також певні відносини між акторами та варіантами використання [42].

Відобразимо на діаграмі варіантів використання основні вимоги до програми, що було описано у технічному завданні (додаток А). На рисунку 3.1 представлено виявлених акторів, а саме користувача програми як єдиного діючого актора, що повноцінно використовуватиме всі функції створюваної програми.

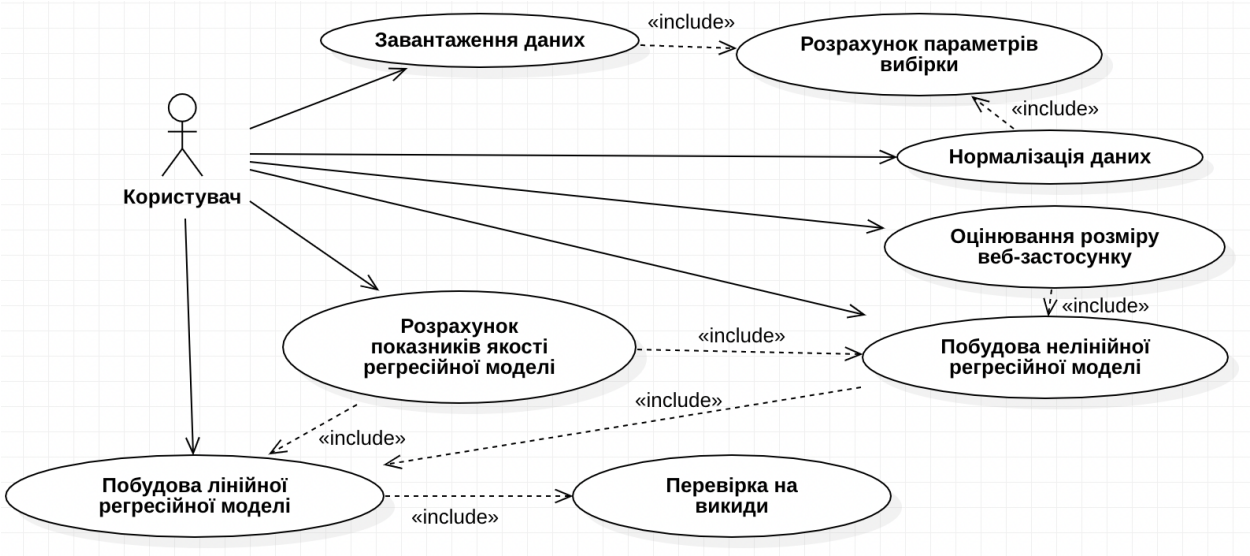

Рисунок 3.1 – Діаграма варіантів використання програми для оцінювання розміру веб-застосунків

Виконаємо аналіз діаграми варіантів використання програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та опишемо специфікацію для кожного з них.

3.1.2 Специфікація варіантів використання розроблюваної програми

В таблицях 3.1 – 3.8 специфіковано всі зазначені на рисунку 3.1 варіанти використання програми для оцінювання розміру веб-застосунків.

Таблиця 3.1 – Специфікація варіанту використання №1 «Завантаження даних»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач завантажує емпіричні дані
Тип	Базовий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Розрахунок параметрів вибірки.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач відкриває програму та ініціює завантаження даних.	2. Виконується завантаження файлу. *Альтернативний потік №1:* Файл не відповідає формату .xlsx. 3. Виконується форматування даних. 4. Відбувається варіант використання «Розрахунок параметрів вибірки». 5. Система відображає вікно із відформатованими даними з можливістю перегляду розрахованих параметрів вибірки.
Альтернативні потоки	
Альтернативний потік №1: Файл не відповідає формату .xlsx.	
	3. Система відображає інформацію про те, що файл не відповідає допустимому формату.

Таблиця 3.2 – Специфікація варіанту використання №2 «Розрахунок параметрів вибірки»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач переглядає розраховані параметри вибірки
Тип	Додатковий
Посилання на інші варіанти використання	Відсутнє
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач ініціює перегляд розрахованих параметрів вибірки.	2. Система розраховує вибіркове середнє, дисперсію та середньоквадратичне відхилення за кожним параметром вибірки. 3. Система відображає вікно із розрахованими параметрами вибірки.

Таблиця 3.3 – Специфікація варіанту використання №3 «Нормалізація даних»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач переглядає сторінку «Нормалізовані дані»
Тип	Базовий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Розрахунок параметрів вибірки.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач переходить до сторінки «Нормалізовані дані».	2. Система виконує нормалізацію вихідних даних десятковим логарифмом. 3. Відбувається варіант використання «Розрахунок параметрів вибірки». 4. Система відображає вікно із нормалізованими десятковим логарифмом даними з можливістю перегляду розрахованих параметрів вибірки.

Таблиця 3.4 – Специфікація варіанту використання №4 «Оцінювання розміру веб-застосунку»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач виконує оцінку розміру веб-застосунку
Тип	Базовий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Побудова нелінійної регресійної моделі.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач переходить до вкладки «Нелінійна модель».	2. Система відображає вікно із полями для введення кількості методів та кількості класів.
3. Користувач вводить кількість методів, кількість класів проекту. 4. Користувач ініціює розрахунок результуючої кількості рядків коду.	5. Відбувається варіант використання «Побудова нелінійної регресійної моделі». 6. Система розраховує результуючу кількість рядків коду проекту. 7. Система виводить у відповідному полі результат розрахунку.

Таблиця 3.5 – Специфікація варіанту використання №5 «Побудова лінійної регресійної моделі»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач виконує побудову лінійної регресійної моделі
Тип	Базовий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Перевірка на викиди.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач переходить до вкладки «Лінійна модель». 2. Користувач ініціює побудову лінійної регресійної моделі.	3. Відбувається варіант використання «Перевірка на викиди». 4. Система знаходить коефіцієнти лінійного рівняння регресії. 5. Система виводить отриману лінійну регресійну модель.

Таблиця 3.6 – Специфікація варіанту використання №6 «Побудова нелінійної регресійної моделі»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач виконує побудову нелінійної регресійної моделі
Тип	Базовий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Побудова лінійної регресійної моделі.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач переходить до вкладки «Нелінійна модель». 2. Користувач ініціює побудову нелінійної регресійної моделі.	3. Відбувається варіант використання «Побудова лінійної регресійної моделі». 4. Система розраховує нелінійне регресійне рівняння за зворотним до логарифмічного перетворення. 5. Система виводить отриману нелінійну регресійну модель.

Таблиця 3.7 – Специфікація варіанту використання №7 «Перевірка на викиди»

Назва	Опис
Актори	Користувач
Короткий опис	Виконання перевірки даних на викиди
Тип	Базовий
Посилання на інші варіанти використання	Відсутнє
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач ініціює перевірку даних на викиди.	2. Система виконує перевірку даних на викиди. 3. Система виводить кількість прибраних даних, що не задовольняють вимогам.

Таблиця 3.8 – Специфікація варіанту використання №8 «Розрахунок показників якості регресійної моделі»

Назва	Опис
Актори	Користувач
Короткий опис	Користувач отримує розраховані показники якості регресійної моделі
Тип	Додатковий
Посилання на інші варіанти використання	Включає в собі варіант використання: • Побудова лінійної регресійної моделі; • Побудова нелінійної регресійної моделі.
Типовий хід подій	
Дії акторів	Відгук системи
1. Користувач ініціює розрахунок показників якості регресійної моделі.	2. Система розраховує показники якості регресійної моделі. 3. Система виводить результати розрахунку.

Отже, згідно з вимогами, до усіх варіантів використання було розроблено специфікації, що описують функціональність розроблюваного програмного продукту.

3.1.3 Подання деяких варіантів використання розроблюваної програми діаграмами діяльності

Діаграма діяльності (active diagram) – це динамічне уявлення системи, тобто діаграма, що показує переходи від одного виду діяльності до іншого але з відображенням лише логічних потоків. Особливий випадок – діаграма

станів, на якій всі або більшість станів прив'язані до дій і всі або більшість переходів зі стану до стану прив'язані до виконання дій у вихідних станах [42].

Виконаємо опис робочих процесів на рисунку 3.2 та рисунку 3.3 для варіантів використання «Завантаження даних» та «Розрахунок параметрів вибірки» відповідно як послідовності дій, які впорядковані, тобто результат одного виду діяльності є вихідними даними для наступного.

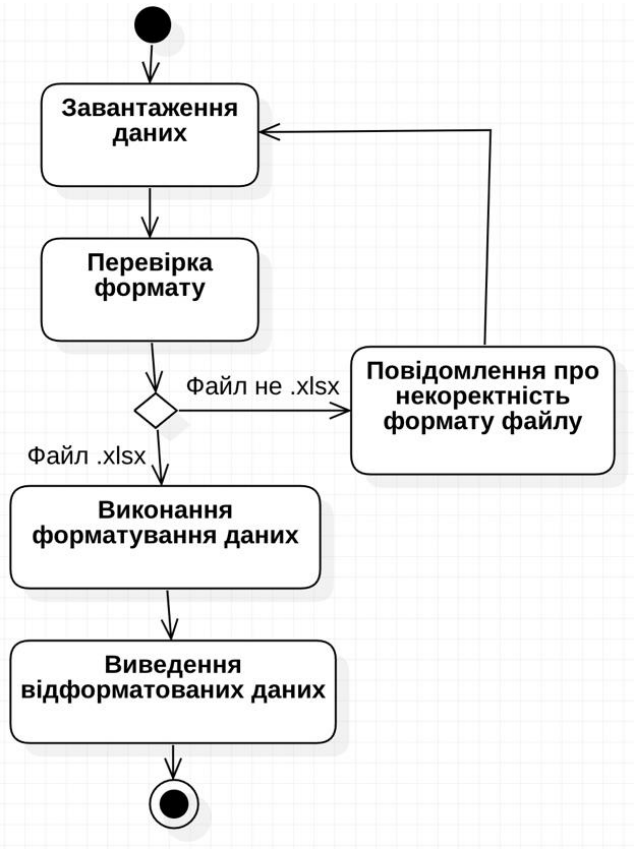

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Завантаження даних»

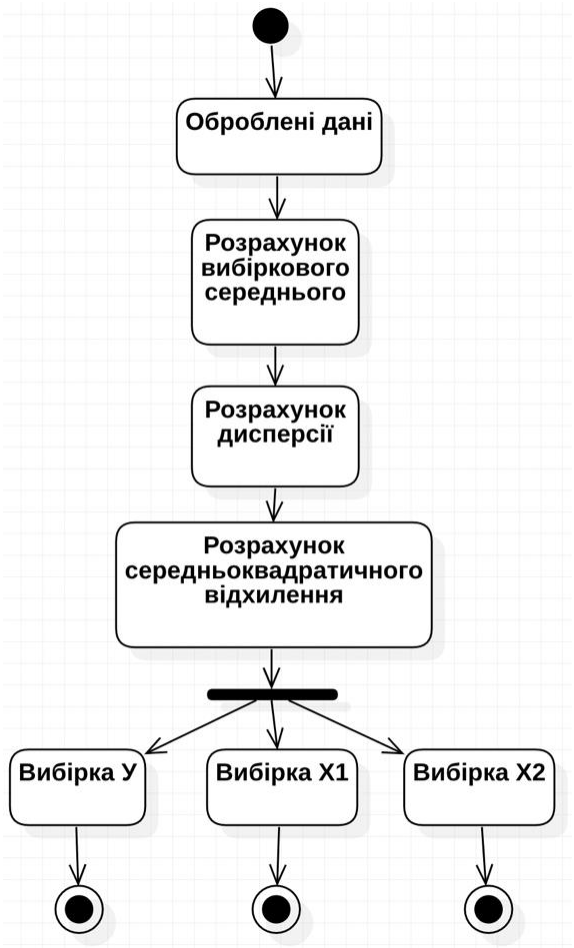

Рисунок 3.3 – Діаграма діяльності для варіанту використання «Розрахунок параметрів вибірки»

Отже, для варіантів використання «Завантаження даних» та «Розрахунок параметрів вибірки» було розроблено діаграми діяльності, завдяки яким отримали краще відображення послідовності дій та подій, що можуть ініціювати дії або бути кінцевим результатом.

3.1.4 Проект користувальницького інтерфейсу

Спільним для успішних програм є наявність виняткового, інтуїтивно зрозумілого зовнішнього вигляду, що сприймається різними користувачами, тому важливо розуміти, що на успіх розроблюваного проекту також впливає користувальницький інтерфейс та UX [43].

На рисунку 3.4 зображено ескіз форми для відображення як емпіричних, так і нормалізованих даних, а також форми для розрахованих параметрів обидвох вибірок в залежності від обраного користувачем екрану.

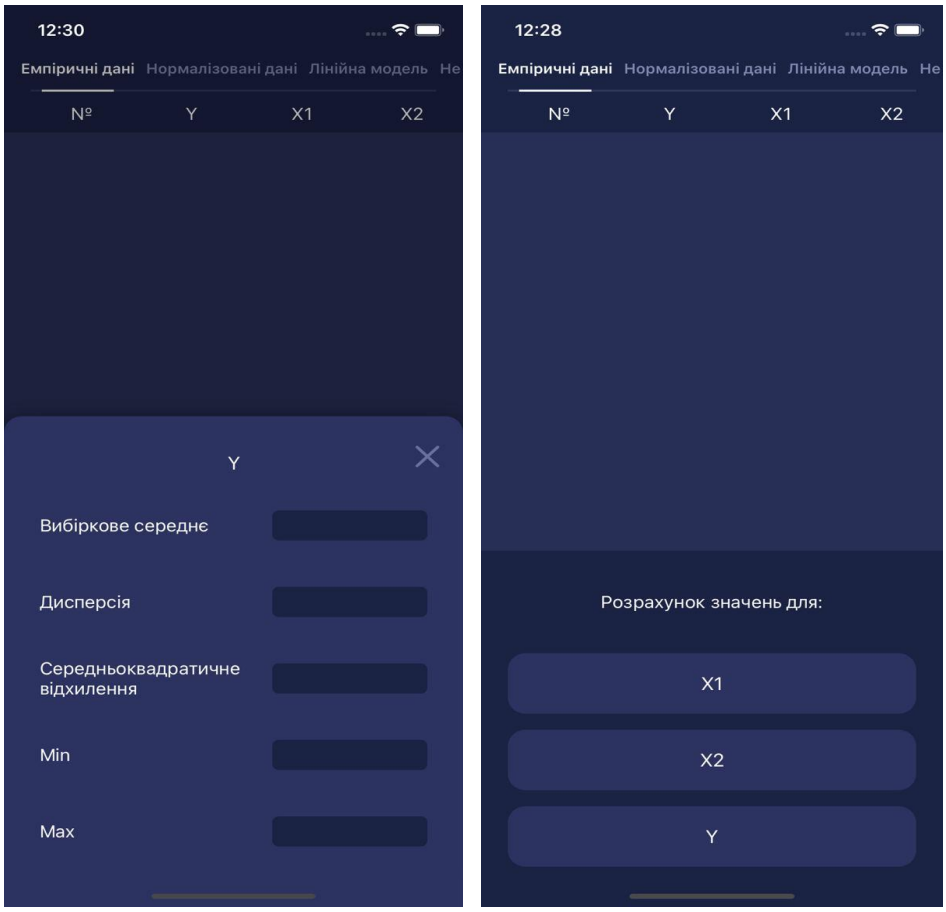

Рисунок 3.4 – Ескіз форм відображення нормалізованих даних та розрахованих параметрів вибірки

Чудовий користувальницький інтерфейс однозначно привертає увагу користувача на програму, тому важливо правильно налаштувати інтерфейс користувача та UX з першого разу, коли виконується розробка програми. Інтерфейс користувача – це елемент, який описує зовнішній вигляд програми, коли користувач взаємодіє з програмою. Інтерфейс користувача гарантує, що користувач може легко взаємодіяти з програмою. Інтерфейс користувача містить такі елементи, як дизайн програм, графіка та презентація. Хороший інтерфейс користувача має бути привабливим для різних користувачів.

Отже, основна мета інтерфейсу користувача – максимально просто, приємно та ефективно взаємодіяти між користувачем і програмою. Вибір кольору, фірмовий стиль і сучасні концепції дизайну – все це процеси розробки інтерфейсу користувача.

3.2 Технічний проект програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework

За технічним завданням, що описується в додатку А, і ескізним проектом розробимо технічний проект програми, та подамо його у вигляді статичної та динамічної моделі.

Статичне моделювання разом з динамічним моделюванням показує організацію системи під час її виконання, заохочує побачити проблему в цілому, тобто комбінацію кількох компонентів та їх взаємодію один з одним.

Розробимо проектну модель, використавши модель варіантів використання в якості вихідних даних. Подамо її у вигляді діаграми класів та діаграми взаємодії, а саме діаграми послідовностей.

3.2.1 Статична модель програми

Статичне моделювання полягає в тому, що виконується моделювання, як організованої системи проблемної області, іншими словами, структура системи, яка зазвичай з меншою ймовірністю зміниться.

Моделі створюються, щоб полегшити розуміння проблеми та створити для неї рішення. Статичне моделювання, порівняно з традиційним підходом, коли проблема аналізується процедурно, пропонує інший погляд на систему, яку потрібно змоделювати – структурний.

Діаграма класів – це діаграма, яка використовується при розробці та моделюванні ПЗ для опису класів та їхніх зв’язків.

Діаграми класів дозволяють моделювати ПЗ на високому рівні абстракції без необхідності переглядати вихідний код. Класи на діаграмі класів відповідають класам у вихідному коді. Діаграма показує імена та атрибути класів, зв’язки між класами, а іноді також методи класів.

На рисунку 3.5 зображено діаграму класів програми з оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

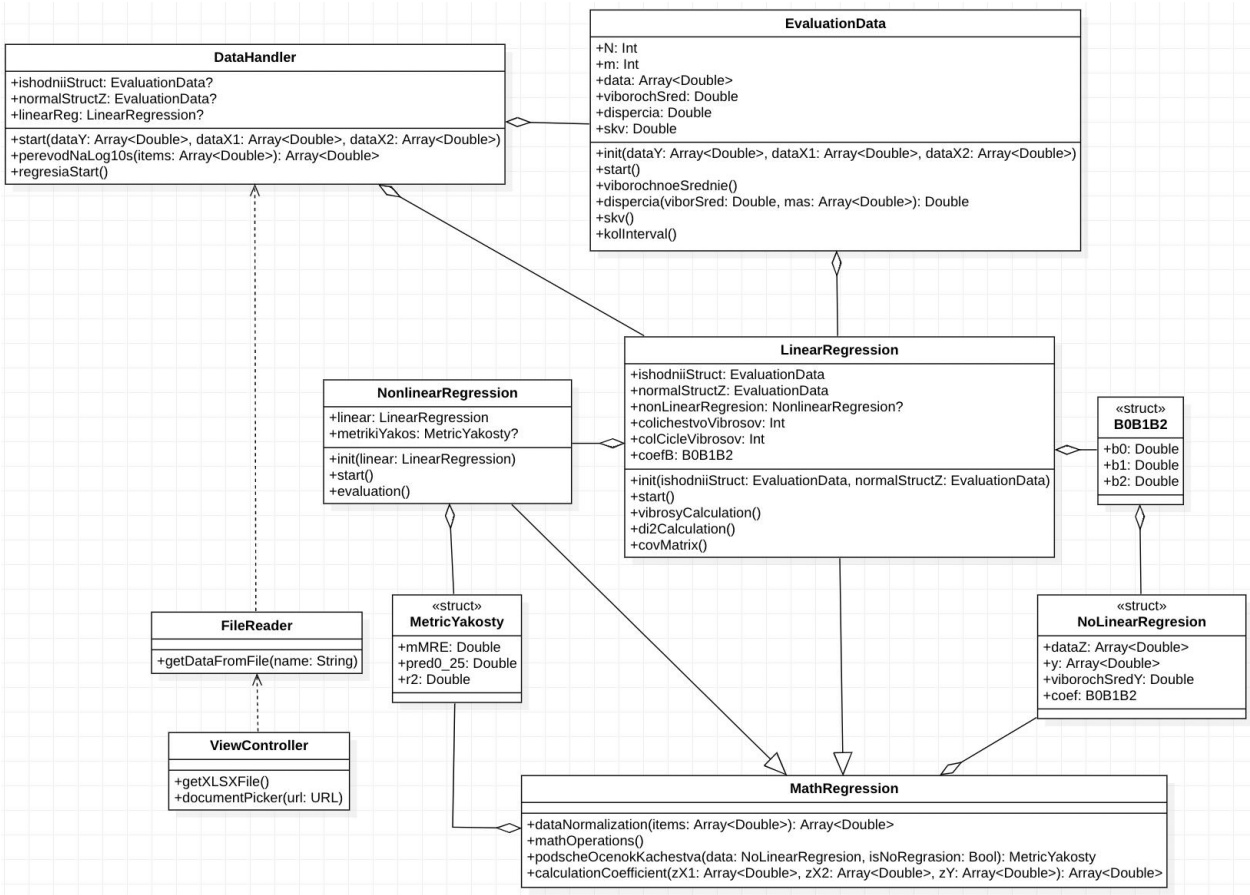

Рисунок 3.5 – Діаграма класів розроблюваної програми

Отже, у вигляді діаграми класів було представлено статичну модель розроблюваного програмного продукту, що демонструє класи, атрибути, та методи системи, а також взаємозв'язки між ними.

3.2.2 Динамічна модель програми

Простого моделювання статичної структури недостатньо для проектування системи, оскільки різні події впливають на стан системи і відповідно впливають на кінцеву структуру системи. Таким чином, краще розуміння динамічної поведінки системи допоможе в подальшому вдосконаленні дизайну.

Динамічна модель системи має на меті визначити, як змінюється стан різних об’єктів, коли відбуваються події. Подія – це те, що відбувається в певний момент часу. Для об'єкта подія, по суті, є запитом на виконання операції. Подія, як правило, є подією чогось і не має пов’язаної з нею тривалості. У кожної події є ініціатор і особа, яка відповідає. Події можуть бути внутрішніми для системи, і в цьому випадку ініціатор події, і особа, яка відповідає на події, знаходяться в системі. А також подія може бути зовнішньою подією, і в цьому випадку ініціатор події знаходиться поза системою.

На рисунку 3.6 зображено діаграму послідовності за описаними варіантами використання №1-3 для виводу отриманих з файлу даних та даних після проведення нормалізації із відповідними розрахованими параметрами отриманих вибірок для програми з оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework. Рисунок 3.7 відображає діаграму послідовності виконання перевірки даних на викиди, тобто варіант використання №7.

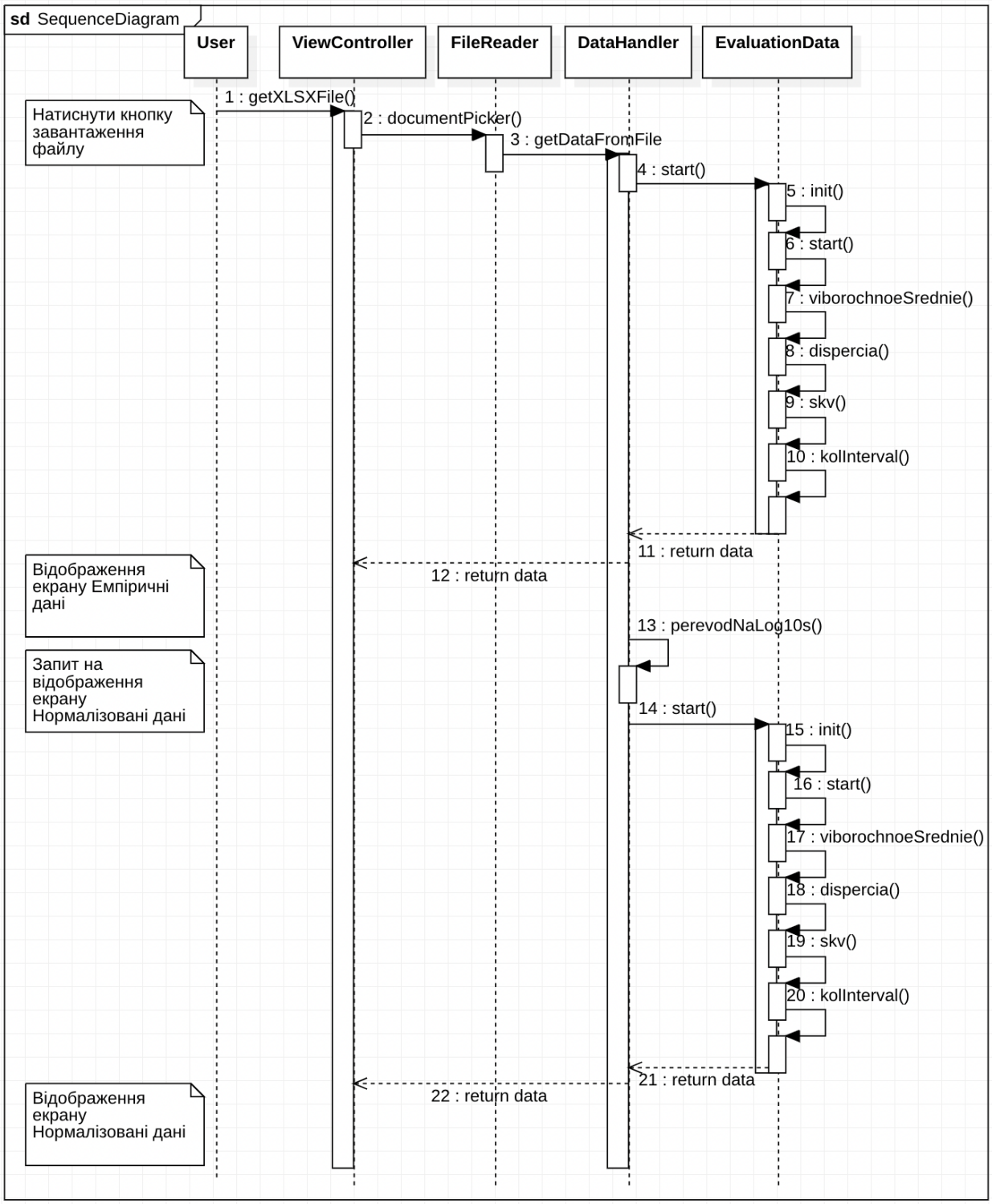

Рисунок 3.6 – Діаграма послідовності частини розроблюваної програми

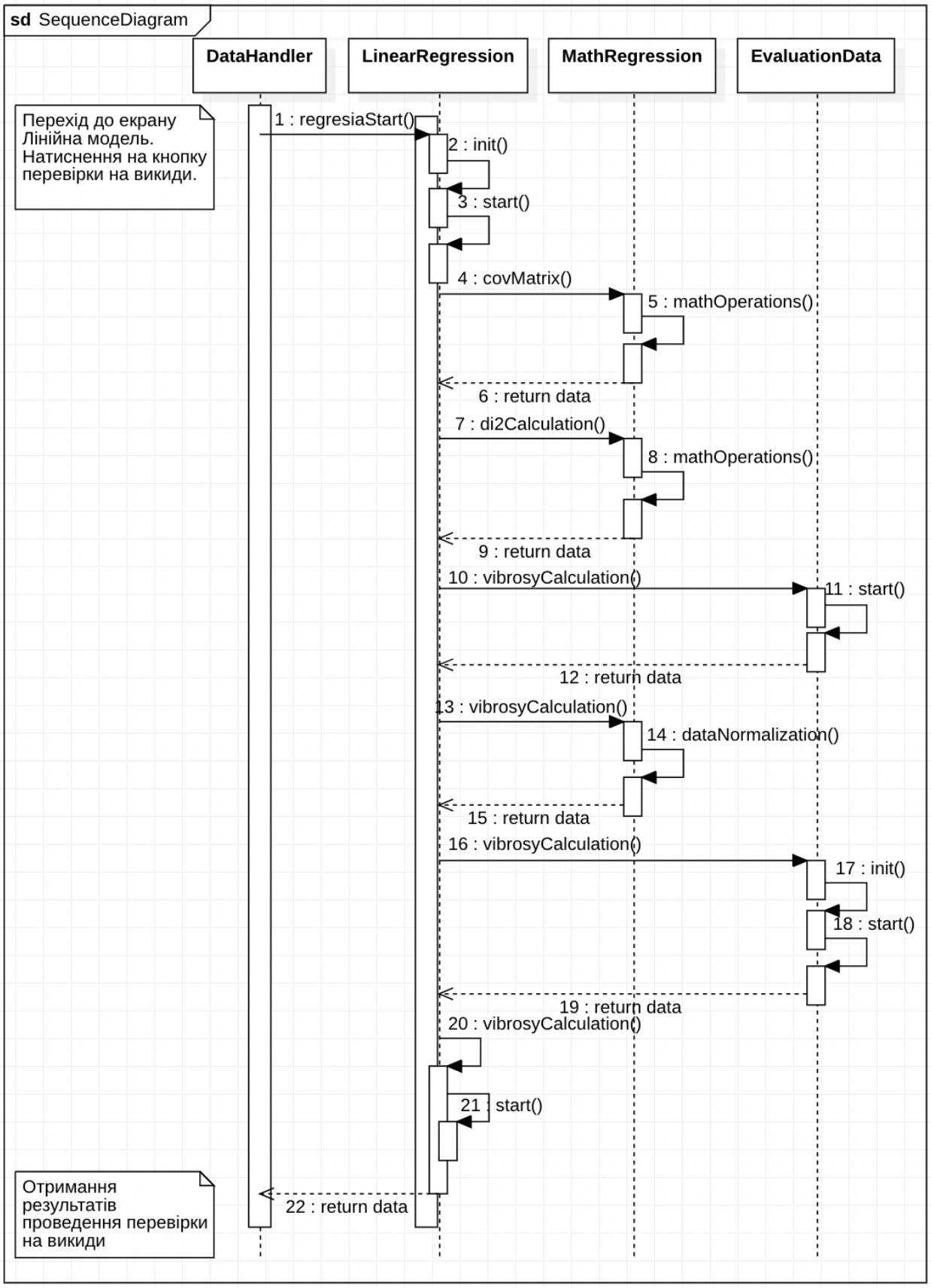

Рисунок 3.7 – Діаграма послідовності перевірки даних на викиди

Отже, у вигляді діаграм послідовностей було представлено динамічну модель частини розроблюваного програмного продукту, що демонструє поведінку системи при ініціюванні користувачем певних дій.

3.3 Робочий проект програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework

Для розробки програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, у вигляді мобільного застосунку було обрано мову програмування Swift 5 та середу розробки, що відповідає обраній мові, а саме Xcode.

3.3.1 Обґрунтування вибору мови та засобів розроблюваної програми

Swift – це молода мова програмування з відкритим вихідним кодом, розроблена Apple у 2014 році спеціально для пристроїв під керуванням macOS та iOS. Вона є мовою програмування загального призначення, створена з використанням унікального підходу до безпеки даних, шаблонів проектування та продуктивності. Досить проста у використанні, що благотворно позначається на швидкості розробки, та є першою мовою системного програмування промислової якості, яка є такою ж виразною та приємною, як мова сценаріїв [44].

Swift було створено як мову, яку можна застосовувати в найрізноманітніших сферах – від системного програмування до мобільних і настільних програм. Це також дає можливість масштабувати проекти до хмарних служб (неможливе завдання для більшості платформ, створених за допомогою програм).

Пам'яттю Swift можна керувати автоматично. Swift запозичив і вдосконалив ряд параметрів з інших мов. Серед цих функцій є іменовані параметри, які також є в Objective-C. Але в Swift вони представлені в чистому синтаксисі, що полегшує читання та підтримку API Swift.

Для реалізації поставленої задачі було обрано мову Swift через ряд її переваг, а саме:

- наявність бібліотек для розробки програм для Mac, iPhone та iPad;
- наявність спільноти Super Open Source Swift [45], що означає, що мова є максимально зручною та простою для розробників. Вони також мають доступ до вдосконалення мови та виправлення помилок;
- наявність спеціального інструменту Playground [46], завдяки якому одразу можна побачити результати роботи програми. Іноді це збільшує швидкість процесу розробки в кілька разів і допомагає швидко впоратися з проблемною ділянкою коду;
- легка підтримка коду, оскільки весь вміст файлів реалізації та заголовків об’єднано в один файл. Тобто розробники програм можуть працювати з логікою програми замість підтримки кількох файлів;
- сумісність надзвичайно важлива для будь-якої мови, і Swift справді добре справляється з цим. Swift добре працює з Objective-C завдяки використанню загальних методів і класів.

Xcode – це інтегроване середовище розробки (IDE) для розробки програм в екосистемі Apple [47]. Xcode містить усе необхідне для створення описаної програми, включаючи багатий набір інструментів для проектування інтерфейсів користувача, створення програми та тестування коду. Xcode містить і інтегрує багато потужних інструментів, які полегшують розробку ПЗ для програмістів, основні з яких: редактор вихідного коду; створення та візуалізація інтерфейсу користувача без коду; тестування, виявлення та виправлення проблем у коді; пошук помилок у коді та пропозиції щодо їх виправлення; редагування кількох файлів у Xcode поруч; тестування та запуск програми на симульованому iPhone на комп’ютері.

UIKit – це структура, яка використовується для розробки інтерфейсу користувача для програм Apple. За допомогою UIKit можна створити інтерфейси, не турбуючись про базовий код [48].

Xcode Device Simulator дозволяє тестувати свою програму на різноманітних пристроях. Симулятор необхідний для перевірки інтерфейсу і функціональності програми без необхідності інсталювати її на фізичному пристрої. Також він дає змогу перевірити реакцію програми на різні розміри та орієнтації екрана [49].

3.3.2 Діаграми компонентів розроблюваної програми

Діаграма компонентів використовується для розбиття великої об’єктно-орієнтованої системи на менші компоненти, щоб зробити їх більш керованими. Вона моделює фізичний вигляд системи, такий як виконувані файли, бібліотеки, тощо, які знаходяться всередині вузла, а також візуалізує зв’язки і організацію між компонентами, присутніми в системі, що допомагає у формуванні виконуваної системи [50].

Компонент – це окремий блок системи який можна замінити та виконати, тобто блок логічної одиниці системи, дещо вища абстракція ніж класи. Деталі реалізації компонента приховані, і для виконання функції потрібен інтерфейс. Це як чорний ящик, поведінка якого пояснюється наданими та необхідними інтерфейсами.

Діаграми компонентів часто малюються, щоб допомогти змоделювати деталі впровадження та ще раз перевірити, чи всі аспекти необхідних функцій системи охоплені плановою розробкою. У першій версії UML компоненти, включені в ці діаграми, були фізичними: документи, таблиця бази даних, файли та виконувані файли, усі фізичні елементи з розташуванням, тоді як в UML 2 ці компоненти є менш фізичними та більш концептуальними автономними елементами дизайну, такими як бізнес-процес, який надає або вимагає інтерфейсів для взаємодії з іншими

конструкціями в системі. Фізичні елементи, описані в UML 1, такі як файли та документи, тепер називаються артефактами, а компонент UML 2 може містити кілька фізичних артефактів, якщо вони природно належать один одному [51].

На рисунку 3.8 представлено діаграму компонентів програми з оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

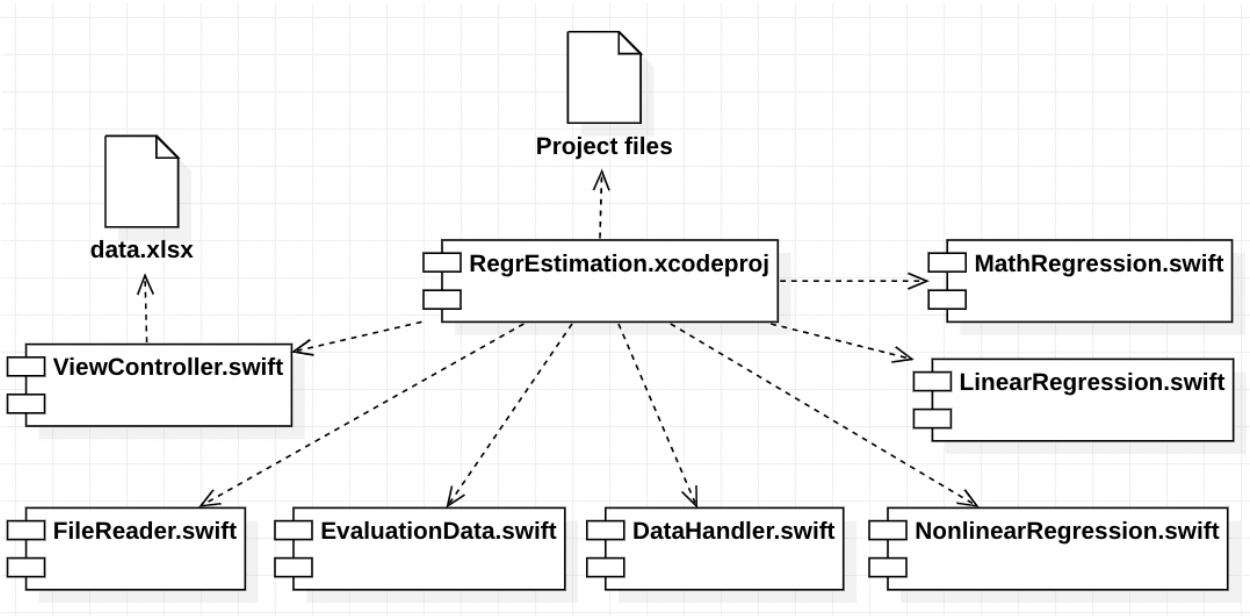

Рисунок 3.8 – Діаграма компонентів розроблюваної програми

Отже, у процесі створення робочого проекту для програмного продукту було розроблено діаграму компонентів розроблюваної програми.

3.3.3 Реалізація та тестування розроблюваної програми

Реалізація була виконана на операційній системі macOS мовою програмування Swift, де в якості середовища розробки було обрано Xcode. Для розробки інтерфейсу було використано фреймворк UIKit, а для відлагодження проекту та його тестування було використано Xcode Device Simulator.

В додатку Б наведено текст розроблюваної програми. Проведемо її тестування.

Тестування чорної скриньки – це техніка тестування ПЗ, яка перевіряє функціональність програмного забезпечення, не вдивляючись у його внутрішню структуру чи кодування. Основним джерелом тестування «чорної скриньки» є специфікація вимог, заявлених замовником [52].

У цьому методі тестувальник вибирає функцію та надає вхідне значення для перевірки її функціональності, а також перевіряє, чи функція видає очікуваний вихід чи ні. Якщо функція дає правильний вихід, то вона проходить тестування, інакше – невдача. Команда тестувальників повідомляє про результат групі розробників, а потім тестує наступну функцію. Після завершення тестування всіх функцій, якщо є серйозні проблеми, їх повертають групі розробників для виправлення.

В ході проведення тестування програмного продукту були протестовані всі компоненти. Нижче наведено тестування функцій завантаження даних в програму з файлу та введення користувачем даних для оцінювання розміру веб-застосунку методом «чорної скриньки».

В таблиці 3.9 представлені класи еквівалентності вхідних даних для завантаження даних з файлу, що описано у варіанту використання №1 «Завантаження даних».

Таблиця 3.9 – Класи еквівалентності вхідних даних для варіанту використання «Завантаження даних»

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
Файл із даними	1 Файл правильного формату	2 Файл пустий 3 Файл неправильного формату

В таблиці 3.10 представлено тестування правильних класів еквівалентності.

Таблиця 3.10 – Тестування правильних класів еквівалентності для варіанту використання «Завантаження даних»

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	1	Файл з розширенням типу .xslx	Файл успішно завантажено, виконано запис даних

Отже, за отриманими результатами з наведеної таблиці можна зробити висновок, що тестування правильних класів еквівалентності для варіанту використання «Завантаження даних» пройшло успішно.

В таблиці 3.11 представлено тестування неправильних класів еквівалентності.

Таблиця 3.11 – Тестування неправильних класів еквівалентності для варіанту використання «Завантаження даних»

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	2	Пустий файл	Повідомлення про помилку
2	3	Файл з розширенням типу .jpeg	Повідомлення про помилку

Отже, за отриманими результатами з наведеної таблиці можна зробити висновок, що тестування неправильних класів еквівалентності для варіанту використання «Завантаження даних» пройшло успішно.

В таблиці 3.12 представлені класи еквівалентності вхідних даних для виконання користувачем оцінювання розміру веб-застосунку, що описано у варіанті використання №4 «Оцінювання розміру веб-застосунку».

Таблиця 3.12 – Класи еквівалентності вхідних даних для варіанту використання «Оцінювання розміру веб-застосунку»

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
Кількість класів	1 Цілочисловий формат даних	2 Число дробове 3 Не числовий формат даних 4 Від'ємне число
Кількість методів	5 Цілочисловий формат даних	6 Число дробове 7 Не числовий формат даних 8 Від'ємне число

В таблиці 3.13 представлено тестування правильних класів еквівалентності.

Таблиця 3.13 – Тестування правильних класів еквівалентності для варіанту використання «Оцінювання розміру веб-застосунку»

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	1	В форму введено ціле додатне число	Дані прийнято
2	5	В форму введено ціле додатне число	Дані прийнято

Отже, за отриманими результатами з наведеної таблиці можна зробити висновок, що тестування правильних класів еквівалентності для варіанту використання «Оцінювання розміру веб-застосунку» пройшло успішно.

В таблиці 3.14 представлено тестування неправильних класів еквівалентності.

Таблиця 3.14 – Тестування неправильних класів еквівалентності для варіанту використання «Оцінювання розміру веб-застосунку»

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	2	25,555	Повідомлення про помилку
2	3	Двадцять п’ять	Повідомлення про помилку
3	4	-25	Повідомлення про помилку
4	6	67,8	Повідомлення про помилку
5	7	Шістдесят сім	Повідомлення про помилку
6	8	-67	Повідомлення про помилку

Отже, за отриманими результатами з наведеної таблиці можна зробити висновок, що тестування неправильних класів еквівалентності для варіанту використання «Оцінювання розміру веб-застосунку» пройшло успішно.

3.3.4 Випробування розроблюваної програми

Випробування розроблюваного програмного продукту виконано відповідно до програми та методики випробувань, що наведена в Додатку Д.

В процесі проведення випробувань було виконано повне функціональне тестування з метою виявлення параметрів, які забезпечують визначення причин збою, показників якості системи та її відповідності різним вимогам, перевірки та отримання проектних рішень, а також характеристики тривалість та періоду випробувань.

Фіксування в протоколах та усунення всіх виявлених недоліків розроблюваної програми були виконано до моменту впровадження програмного продукту в дію.

На початку роботи з програмою користувачу надається інтерфейс для вибору файлу з даними для проведення подальших розрахунків та маніпуляцій з ними, що наведено на рисунку 3.9. На цьому етапі може виникнути помилка некоректного вибору типу файлу (не .xlsx) і в цьому випадку користувач побачить відповідне повідомлення на екрані з можливістю повторного вибору файлу.

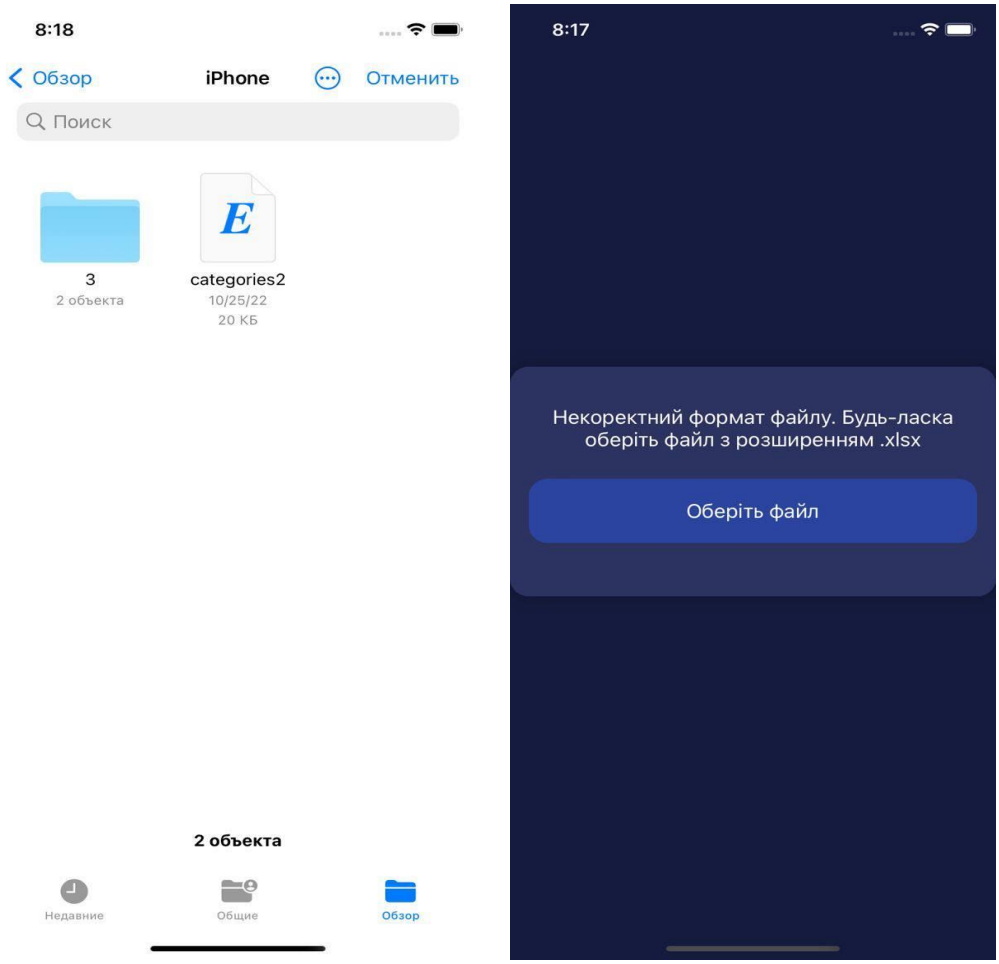

Рисунок 3.9 – Завантаження вхідних даних користувачем

Під час тестування для проведення якісного випробування програми було відтворено поведінку користувача і перевірено увесь розроблений функціонал. Так, наприклад, на рисунку 3.10 зображено що для виконання розрахунку параметрів якості прогнозування лінійної регресійної моделі було виконано перевірку на викиди та безпосередньо побудову лінійної регресійної моделі.

Рисунок 3.10 – Екран побудови лінійної регресійної моделі

Для оцінювання розміру програмного проекту необхідно перейти до відповідного екрану та натиснути кнопку «Оцінити розмір веб-застосунку». В результаті цих дій користувач отримає новий екран, що зображено на рисунку 3.11, в якому можна ввести кількість методів та класів проекту, що цікавить користувача, та ініціювати оцінку для отримання розрахованого значення кількості рядків коду.

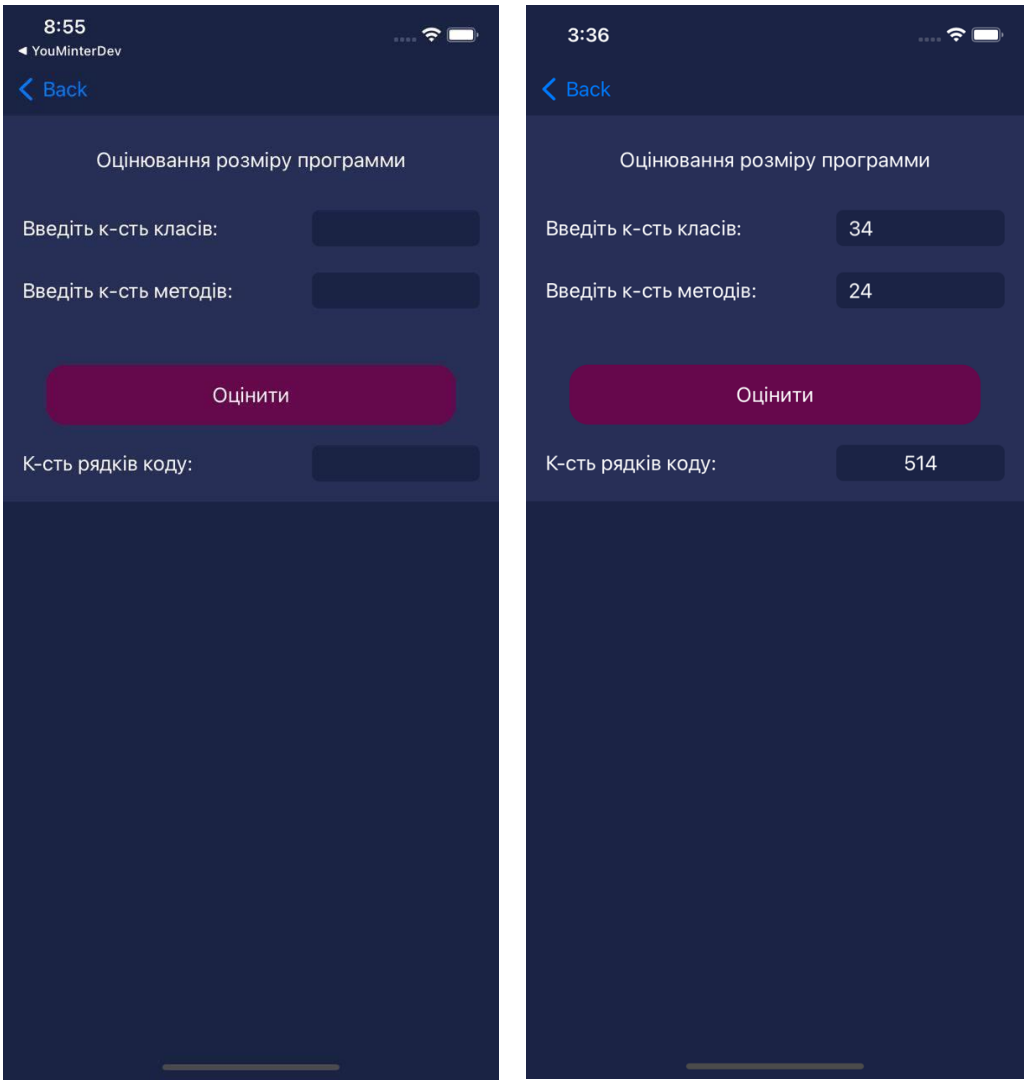

Рисунок 3.11 – Екран оцінювання розміру програмного проекту

Отже, було проведено випробування розроблюваного програмного продукту в результаті якого можемо зробити висновок, що система відповідає поставленим вимогам та передбачуваному рівню надійності.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ PYTHON З ВИКОРИСТАННЯМ DJANGO REST FRAMEWORK

В процесі виконання кваліфікаційної роботи було розроблено програму для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, який демонструє практичне значення одержаних результатів.

Розроблена програма була створена мовою програмування Swift в середовищі Xcode, а також протестована на відповідність описаному технічному завданню. Текст програми наведено в додатку Б.

Згідно з вимогами до програмного продукту, наведеними в технічному завданні (додаток А), програмний продукт надає можливість виконання наступного переліку функцій:

• внесення вхідних даних про проекти у вигляді файлу з кількістю рядків коду, класів та методів різних проектів;

• застосування до внесених даних нормалізуючого перетворення у вигляді десяткового логарифму;

• застосування для внесених даних перевірки на викиди;

• побудова лінійної регресійної моделі;

• розрахунок метрик якості лінійної регресійної моделі;

• побудова нелінійної регресійної моделі з використанням зворотного нормалізуючого перетворення;

• розрахунок метрик якості нелінійної регресійної моделі;

• розрахунок довірчого інтервалу та інтервалу передбачення для нелінійної регресії;

• розрахунок оцінювання розміру веб-застосунку;

• виведення отриманих розрахунків на екран.

Також було розроблено наступну документацію:

- технічне завдання (додаток А);
- текст програми (додаток Б);
- опис програми (додаток В);
- інструкція користувача (додаток Г);
- програма і методика випробувань (додаток Д).

Оскільки створюваний програмний продукт повністю відповідає вимогам до організації вхідних та вихідних даних, вимогам до надійності та іншим вимогам, зазначеним в технічному завданні (додаток А), можемо дійти висновку, що було досягнуто поставленої мети щодо розробки програми для оцінювання розміру веб-застосунків.

Перевагою розробленого програмного продукту є те, що інтерфейс виконано у вигляді мобільного додатку, що може бути зручнішим в користуванні завдяки своїй мобільності та може бути застосовано в любому місці та в любий момент без необхідності користування стаціонарними ПК.

Під час тестування програмний продукт показав повну працездатність і завдяки інтуїтивно зрозумілому інтерфейсу його можуть використовувати користувачі з різним рівнем навичок володіння технічними засобами, а також він не потребує від користувача знань з програмування.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ PYTHON З ВИКОРИСТАННЯМ DJANGO REST FRAMEWORK

Дана кваліфікаційна робота присвячена розробці програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

Ефективність інформаційних систем визначається ступенем їх позитивного впливу на підприємство, що управляється або процес в результаті використання засобів обчислювальної техніки, програмного забезпечення, зміни інформаційних потоків та організаційної структури управління.

Створення програмного продукту вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

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

5.1 Розрахунок витрат на створення й експлуатацію розроблюваної програми

Витрати на розробку системи складаються з витрат на заробітну платню розробника; на амортизацію ЕОМ, на якій виконується розробка; на експлуатацію ЕОМ; на засоби розробки; на матеріали і комплектуючі.

Розробка програмного продукту виконується програмістом, місячна заробітна плата якого складає 25000 грн. Вартість сучасного комп'ютера складає 40000 грн. (наведено вартість macbook на базі процесора M1).

Витрати на допоміжні матеріали наведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Доступ до мережі Internet	305
2	Папір А4 (500 арк, упаковка, клас А)	220
3	Заправлення порошкового картриджа до принтера	150
4	Непередбачувані витрати	250
	Разом	925

Вартість програмного продукту знаходиться за формулою:

$$К_{пр} = (В_{зп} + В_{есв} + В_{зг} + В_{е}) * Т + В_{дм} \quad (5.1)$$

де: $В_{зп}$ – витрати на оплату праці, встановлені зарплатнею працівника, грн.;

$В_{есв}$ – єдиний соціальний внесок, що складає 22% від витрат на заробітну плату, грн.;

$В_{зг}$ – загальногосподарські витрати, що складають 10% від витрат на оплату праці без урахування податкового навантаження, грн.;

$В_е$ – витрати за електроенергію, грн.;

Т – тривалість розробки, міс.;

$В_{дм}$ – витрати на допоміжні матеріали, грн.

Розрахунок вартості розробки програми виконується при вартості кіловат-години електроенергії 1,68 грн. За умови споживання потужності 0,2 кВт та тривалості розробки на місяць, а саме 22×8=176 годин розрахуємо значення $З_е$:

$$В_е = 176 \cdot 0{,}2 \cdot 1{,}68 = 59{,}14 \text{ грн.}$$

Витрати на розробку програмного продукту наведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку програмного продукту

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки	міс.	1
2	Заробітна плата	грн.	25000
3	Витрати на єдиний соціальний внесок	грн.	5500
4	Загальногосподарські витрати	грн.	2500
5	Витрати на електроенергію	грн.	59,14
6	Загальні витрати на допоміжні матеріали та засоби розробки	грн.	925

Оскільки за методом експертної оцінки на основі розробки аналогічних систем визначено, що загальна тривалість розробки Т складає 1 місяць, а кількість необхідних спеціалістів дорівнює 1 людині, то вартість розробки системи розрахуємо за формулою (5.1) і вона складає:

$$К_{пр} = (25000 + 5500 + 2500 + 59{,}14) * 1 + 925 = 33984{,}14 \text{грн.}$$

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

$$А_{уст} = 40000 \cdot 0{,}6 = 24000 \text{ грн.}$$

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

$$В_{м} = 40000 \cdot 0{,}05 = 2000 \text{ грн.}$$

Оскільки в середньому устаткування споживає 0,07 кВт електроенергії і навантажується близько 6,5 годин на день, а в році маємо 256 робочих днів, можемо розрахувати річне споживання електроенергії устаткуванням:

$$В_{е} = 256 \cdot 6{,}5 \cdot 1{,}68 \cdot 0{,}07 = 195{,}70 \text{ грн.}$$

Експлуатаційні витрати на устаткування на рік складуть:

$$В_{уст} = 24000 + 2000 + 195{,}70 = 26195{,}70 \text{ грн.}$$

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

$$В_{р} = 33984{,}14 + 26195{,}70 = 60179{,}84 \text{ грн.}$$

5.2 Економічна ефективність розробки і впровадження

Основним показником економічної ефективності програмного продукту є зменшення трудових витрат та часу на виконання оцінювання розміру веб-застосунку, а також ефективне розподілення трудових ресурсів на основі отриманих результатів.

Ефективний розподіл ресурсів дає змогу зменшити час тестування, і, відповідно, грошові витрати на роботу спеціалістів відділу забезпечення якості ПЗ.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 1 працівника (за експертною

оцінкою фахівців підприємства). Заробітна плата 1 працівника на рік становить $25000 \cdot 12 \cdot 1 = 300\,000$грн.

Річний економічний ефект розраховується за формулою:

$$\mathrm{E}_{\text{рік}} = \Delta C_n - E_n \cdot \mathrm{k} \qquad (5.2)$$

де: ΔC_n – вивільнені кошти після впровадження продукту віднявши значення експлуатаційних витрат, тобто маємо розрахунок $\Delta C_n = 300000 - 60179{,}84 = 239820{,}16$ грн.;

$\mathrm{E_n}$ – коефіцієнт ефективності (дорівнює коефіцієнту амортизації 0,6);

k – одноразові витрати на впровадження продукту (60179,84 грн.).

Таким чином, за формулою (5.2) можемо виконати розрахунок річного економічного ефекту:

$$\mathrm{E}_{\text{рік}} = 239820{,}16 - 0{,}6 \cdot 60179{,}84 = 203\,712{,}26\text{грн.}$$

Термін окупності програмного продукту розраховується за формулою:

$$\mathrm{T} = \frac{k}{\Delta C_n} = \frac{60179{,}84}{239\,820{,}16} = 0{,}25 \text{ року}$$

Отже, термін окупності програмного продукту становить приблизно 3 місяці.

6 ОХОРОНА ПРАЦІ

Охорона праці є важливим кластером та соціально-економічним напрямом розвитку країни. Головне завдання – це система технічних, економічних, правових, санітарних заходів, спрямованих на забезпечення належних умов праці. Працюючи, людина взаємодіє з виробничим середовищем, як соціальним явищем, з елементами технічного та природного походження. З охороною праці тісно пов'язане питання пожежної безпеки. стан навколишнього повітряного середовища приміщень різного типу, їх цільове освітлення, вентиляційні системи , вібрація, електромагнітно-радіоактивні випромінювання, шум. Метою є дослідження трудового процесу з позиції забезпечити здоров’я та життя людини, зведення до мінімуму ймовірність нещасних випадків. Всі норми пов’язані з охороною праці закріплені в Конституції України і в Законі України «Про охорону праці» [53] та мають законодавчий характер.

На сьогодні в країні затверджено більше 1000 правил та інструкцій, близько 39 державних та 344 міжнародні стандарти, 2014 нормативних галузевих актів, 235 міжгалузевих стандартів.

6.1 Загальні вимоги з охорони праці при роботі на персональному комп'ютері

При роботі на ПК працівник зобов'язаний виконувати роботу, яка визначена його посадовою інструкцією та дотримуватись правил внутрішнього трудового розпорядку праці і відпочинку в залежності від тривалості, виду і категорії трудової діяльності. Також необхідно застосовувати засоби індивідуального або колективного захисту до вимог охорони праці в конкретній ситуації.

Необхідно навчитися методам надання першої допомоги потерпілим на робочому місці, а також знати як надати першу допомогу постраждалим від електричного струму і при інших нещасних випадках.

Окремо пройти інструктаж по протипожежним заходам та вміти застосовувати первинні засоби пожежогасіння.

Загальні вимоги з охорони праці при роботі на ПК:

1. Для роботи на персональному комп'ютері допускаються особи, що проходять спеціальний інструктаж, навчання та стажування на робочому місці, перевірку знань вимог охорони праці, дотримання норм, які мають I групу з електробезпеки.

2. Небезпечні та шкідливі фактори при експлуатації персонального комп'ютера, які можуть діяти на людину:

- надлишок електромагнітних випромінювань;
- великий рівень накопиченої статичної електрики;
- забруднення повітря;
- фізичне перевантаження в статичному положенні;
- перенавантаження на зір;
- мала освітленість місця роботи.

3. Бажано, щоби основні елементи корпусу , клавіатура та інші блоки і пристрої мали матову поверхню, яка не створюватиме відблиски, а приміщення, де розміщуються робочі місця з ПК, повинні бути обладнані захисним заземленням відповідно до технічних вимог з експлуатації. Робочі місця з комп'ютерами повинні розміщуватися таким чином, щоб відстань від екрана одного до тилу іншого було не менше 2м, а відстань між бічними поверхнями не менше 1,2 м. Робочі столи слід розміщувати таким чином, щоб природне світло падало переважно ліворуч.

4. Вікна в приміщеннях, де використовуються персональні комп'ютери, повинні бути обладнані регульованими елементами типу: жалюзі та інші.

5. Освітлення в приміщеннях де розташовані ПК повинне здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративних приміщеннях, у разі переважної роботи з документами, слід застосовувати системи комбінованого освітлення (до загального освітлення додатково встановлюються світильники місцевого освітлення, призначені для освітлення зони розташування документів).

6. Екран монітора повинен знаходитися від очей користувача на відстані 600 - 700 мм, але не ближче 500 мм.

7. У приміщеннях, обладнаних ПК, проводиться щоденне вологе прибирання і систематичне провітрювання .

6.2 Вимоги охорони праці під час роботи та в аварійних ситуаціях

Перед початком роботи за ПК необхідно дотримуватись наступних правил:

1. Згідно вимог треба підготувати робоче місце.

2. Освітлення на робочому місці повинно буди відрегульовано з дотриманням відсутності відблисків на екрані.

3. Перевірити правила підключення обладнання до електромережі.

4. Перевірити цілісність проводів та відсутність оголених на них ділянок.

5. Переконатись в наявності заземлення системного блоку і монітору.

6. Протерти антистатичною серветкою всі поверхні монітора.

7. Перевірити правильність розташування столу, стільця, кута нахилу екрану, положення клавіатури, положення «миші» на спеціальному килимку, при необхідності провести регулювання робочого столу і крісла, а також розташування елементів ПК відповідно до вимог ергономіки та з метою виключення незручних поз, і тривалих напруг тіла.

Працівникові при роботі на ПК забороняється:

- торкатися задньої панелі системного блоку (процесора) при включеному живленні;
- переключати елементи кабелів периферійних пристроїв при включеному живленні;
- допускати потрапляння вологи на поверхню системного блоку (процесора), монітора, робочу поверхню клавіатури, дисководів, принтерів і інших пристроїв;
- проводити самостійне розкриття і ремонт обладнання;
- працювати на комп'ютері при знятих кожухах.

У випадку виникнення аварійної ситуації необхідно дотримуватись наступних правил:

1. У всіх випадках обриву проводів живлення, несправності заземлення та інших пошкоджень, появи горілого, негайно вимкнути живлення і повідомити про аварійну ситуацію керівнику.

2. Не приступати до роботи до усунення несправностей.

3. При виникненні пожежі, задимлення:

- негайно повідомити по телефону «01» в пожежну охорону, сповістити працюючих, довести до відома керівника підрозділу, повідомити про пожежу на пост охорони;
- відкрити запасні виходи з будівлі, знеструмити електроживлення, закрити вікна і прикрити двері;
- приступити до гасіння пожежі первинними засобами пожежогасіння, якщо це не пов'язано з ризиком для життя;
- якщо є ризик для життя , покинути приміщення та бути на безпечній відстані.

4. Дії по закінченню роботи на ПК:

- відключити живлення комп'ютера та пристроїв;
- прибратися на робочому місці.

6.3 Техніка безпеки при роботі з електроприладами

Для роботи обладнання слід використовувати заземлення розетки. Не можна братися за дроти і працюючу техніку мокрими руками. Не слід висмикувати вилку з розетки за шнур. Забороняється користуватися несправними приладами. Не можна розбирати або ремонтувати техніку, включену в розетку. Забороняється включати одночасно кількість електроприладів, що перевищує допустиму кількість для даного споживача. Небезпечно торкатися металевих поверхонь, тримаючи в руках працює електричний пристрій. При включенні обладнання на шнурі не повинно бути вузлів і заломів, щоб уникнути порушення ізоляції. Ні в якому разі не можна торкатися до оголених проводів, світлотехнічної фурнітурі під час монтажу. Кабель слід підключати спочатку до електроприлади, а потім до джерела живлення. Проводи повинні знаходитися на віддаленій відстані від газових труб, радіаторів опалення, обігрівачів, електричних плит, і інших нагріваються пристроїв. Перед підключенням нового електроприладу необхідно уважно вивчити інструкцію.

При влаштуванні на роботу, кожен співробітник проходить інструктаж з техніки безпеки. Вся документація по електробезпеки зібрана в «Правилах з охорони праці при експлуатації електроустановок». Після ознайомлення з інструкцією, працівник розписується в журналі. Це покладає на нього відповідальність за неправильне поводження з обладнанням та електроприладами. У разі недотримання регламенту на порушника може бути покладена адміністративна і навіть кримінальна відповідальність.

На виробництві до роботи з електричним струмом повинні виконувати тільки з відповідною освітою. А до виконання обов'язків на електроустановках дозволяється ставити робочих пройшли навчання і атестацію, з отриманням посвідчення. Процедуру так само проходять начальники, майстри, які не мають прямого контакту з електрикою, але відповідають за своїх підлеглих.

Техніка безпеки на виробництві та в офісі передбачає дотримання таких вимог: Виконання робіт дозволяється в спеціальному одязі. Обладнання повинно експлуатуватися в справному стані і строго за призначенням. Розміщення електроінструменту дозволено в приміщеннях відповідного класу захисту. В умовах роботи у вологому середовищі необхідно використовувати інструмент відповідного класу захисту. Не можна допускати сторонніх під час роботи в лабораторії, цеху, і інші небезпечні ділянки. Перед запуском обладнання слід перевірити цілісність проводів, справність розеток.

При загорянні необхідно знеструмити приміщення і висмикнути дріт з джерела живлення. Тільки після цього накрити щільною повітронепроникної тканиною (ковдрою, покривалом, пледом). Відсутність кисню в осередку загоряння припинить процес. Вогонь не перекинеться на інші предмети. Якщо можливості знеструмити загорівся прилад немає, полум'я можна загасити, засипавши його сухою землею з квіткового горщика, содою, піском. Такий спосіб також перекриє доступ повітря і зупинить пожежа. При загорянні електропроводки технологія аналогічна. Слід спочатку знеструмити приміщення, а потім гасити вогнище загоряння. В офісах, на виробництві по техніці безпеки обов'язковою нормою є наявність вогнегасника. Важливо! Вогнегасник дозволяється розпилювати тільки в провітрюваних приміщеннях. При відключенні електрики допускається застосовувати пінні або водні вогнегасники.

Допомога потерпілому, проблема полягає в тому, що, схопившись за джерело небезпеки руками, людина не здатна відпустити провід через спазм м'язів. Тому потрібно зупинити вплив електрики на потерпілого. Необхідно висмикнути дріт з розетки використовуючи не проводять струм предмети (дерево, гума, тканина) або перекусити його пассатижами. Якщо знеструмити прилад неможливо, потрібно відтягнути від нього потерпілого на віддалене відстань. Не можна торкатися голими руками до шкіри людини, що знаходиться під впливом електричного струму!

Після виконаних заходів необхідно оцінити стан людини і приступити до надання долікарської допомоги, враховуючи ступінь ураження електрикою: У свідомості – перекласти його на жорстку поверхню, обробити шкіру навколо уражених ділянок йодом або марганцівкою, накласти на рани суху пов'язку. В непритомності – звільнити його від одягу, що здавлює, спробувати привести до тями за допомогою нашатирного спирту і обробити опіки. У стані клінічної смерті - реанімувати за допомогою непрямого масажу серця і штучної вентиляції легенів. Якщо м'язи рота спазмовані, то дихання треба робити «рот в ніс». Процедуру слід виконувати до приїзду швидкої допомоги. Наслідки від отриманої електротравми можуть бути різними - від легкого запаморочення до зупинки серця. Надана вчасно долікарська допомога може врятувати життя потерпілому.

Недбале ставлення і зневага технікою безпеки при роботі з електроприладами може призвести до серйозних наслідків. Для збереження життя та здоров'я себе та своїх близьких слід запам'ятати і виконувати елементарні правила. Електрика і вода - поняття не сумісні. Не можна наближатися до джерел вологості з включеними електроприладами. Не слід при роботі обладнання торкатися до матеріалів, які проводять електричний струм. При загорянні проводки або техніки в першу чергу слід знеструмити вогнище загоряння. Не можна гасити водою пристрій під напругою. Щоб зупинити пожежу, слід перекрити доступ кисню за допомогою повітронепроникної тканини або сипучих речовин. При наданні першої допомоги потерпілому, слід запам'ятати: він сам є провідником поки не звільнений від джерела ураження електричним струмом. До приїзду швидкої можна самостійно спробувати врятувати життя людини.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Наука і техніка розвивається в темпах геометричної прогресії, де промисловість не є виключенням як одна із наймасштабніших сфер діяльності людини. У зв'язку з не бездоганністю технологічних процесів на даному етапі неминучий негативний вплив промисловості на навколишнє середовище і промислових відходів як компонента даного впливу. Щорічно в усьому світі й у нашій країні мільярди тонн твердих, рідких, газоподібних відходів потрапляють у біосферу, завдаючи тим самим непоправної шкоди як живій, так і неживій природі.

Зростаючі темпи змін навколишнього середовища приводить до порушення взаємозв'язку між природою і людиною, зниженню адаптаційних можливостей організму людини. Середовище може містити такі речовини, з якими організм в ході еволюції не стикався і тому не має відповідних аналізаторних систем, що сигналізують про їх існування. У зв'язку з цим оцінити стан здоров'я людини та зрозуміти характер патології не проаналізувавши зміни що відбуваються у навколишньому середовищі неможливо.

Велике значення тому має організація інформаційної системи «здоров'я населення – навколишнє середовище», дані для якої можуть збиратися через державну статистичну звітність. Задача державної інформаційної системи полягає у зборі даних про забруднення навколишнього середовища, стан здоров'я населення.

7.1 Забруднення навколишнього середовища в процесі використання комп'ютеризованої техніки

Забруднення – це наявність у навколишньому середовищі шкідливих речовин, що порушують функціонування екологічних систем або інших окремих елементів і знижують якість середовища з огляду проживання людини або ведення її господарської діяльності. Цим терміном характеризують всі тіла, речовини, явища, процеси, що з'являються в даному місці навколишнього середовища, але не в той час і не в тій кількості, які характерні для природи і можуть виводити її системи зі стану рівноваги.

На сучасному етапі розвитку суспільства поводження з відходами водночас з іншими екологічними проблемами посідає одне з чільних місць у системі екологічної безпеки і забезпеченні сталого розвитку країни. Їхнє вирішення пов'язано з необхідністю узгодити комплекс екологічних, економічних і соціальних завдань і вимагає постійних системних зусиль з боку органів управління, науковців та громадськості.

Згідно із Законодавством України про відходи − відходи є об'єктом права власності. Суб'єктами права власності на відходи є громадяни України, іноземці, особи без громадянства, підприємства, установи та організації усіх форм власності, територіальні громади і держава. Суб'єкти права власності володіють, користуються і розпоряджаються відходами в межах, визначених законом.

Відповідно до Закону України «Про відходи» [54], поводження з відходами − дії, спрямовані на запобігання утворенню відходів, їх збирання, перевезення, зберігання, оброблення, утилізацію, видалення, знешкодження і захоронення, втому числі контроль за цими операціями та нагляд за місцями видалення.

Законом передбачено два альтернативних напрями поводження з відходами: утилізація (використання відходів як вторинних матеріальних чи

енергетичних ресурсів) та видалення (в основному шляхом захоронення та знищення або знешкодження).

Використання комп'ютера відіграє велику роль у забрудненні навколишнього середовища. Підраховано, що з 250 мільярдів доларів США на рік, що витрачаються на живлення комп'ютерів у всьому світі, лише близько 15% цієї енергії витрачається на реальне використання або експлуатацію техніки, а решта витрачається в простої (тобто споживається комп'ютерами, які не використовуються, але все ще увімкнені) [55]. Ця споживана енергія є основною причиною викидів CO2, тому енергія, збережена на комп'ютерному обладнанні та обчислювальних машинах, прирівнюватиметься до тонн викидів вуглецю, збережених на рік.

Це почалося ще в 1992 році, коли Агентство з охорони навколишнього середовища США (EPA) запустило програму Energy Star, контрольовану програму маркування для сприяння та визнання енергоефективності. Знак Energy Star наразі сертифікував понад 75 різних категорій продуктів, будинків, комерційних будівель і промислових підприємств. Програма також призвела до широкого впровадження режиму сну серед споживачів електроніки.

7.2 Розробка заходів щодо зменшення забруднення навколишнього середовища

Уряди, промисловість і екологічні неурядові організації започаткували велику кількість ініціатив для сприяння екологічному використанню техніки (Green Computing), в яких IT-індустрія докладає зусиль в усіх своїх секторах. Переробка обладнання, скорочення використання паперу, віртуалізація, хмарні обчислення, керування живленням, екологічне виробництво є ключовими ініціативами щодо Green Computing [56].

"Going Green" – це тенденція, що набирає популярність та зарекомендувала себе як спосіб, за яким краще робити речі, одночасно зберігаючи навколишнє середовище. Зараз це проявляється у багатьох аспектах нашого життя, таких як переробка, енергоефективні пристрої, чисті джерела енергії, екологічно чисті транспортні засоби, зелені будівлі.

Комп'ютеризація також взяла участь у збереженні навколишнього середовища в рамках концепції Green Computing. Green Computing – це екологічно відповідальне та безпечне використання комп'ютерів та їхніх ресурсів. У більш широкому сенсі це також визначається як вивчення проектування, машинобудування, виробництва, використання та утилізації комп'ютерних пристроїв таким чином, щоб зменшити їхній вплив на навколишнє середовище. Green Computing, також відомі як Green Technology або Green IT, швидко стали найефективнішим засобом використання технологій.

В основному Green Computing передбачають зменшення впливу технологій на навколишнє середовище. Це означає використання меншої кількості енергії, зменшення відходів і сприяння сталому розвитку. Green Computing спрямовані на зменшення вуглецевого сліду, створюваного бізнесом інформаційних технологій та систем, і суміжними галузями. Енергоефективність і електронні відходи є двома основними техніками, задіяними в Green Computing. Енергоефективність передбачає впровадження енергоефективних центральних процесорів (CPU), серверів та периферійних пристроїв, а також зниження споживання ресурсів, а електронні відходи – це належна утилізація електронних відходів.

Гарним прикладом є стратегія Intel до 2030 року. Корпорація Intel прагне до постійного прогресу в досягненні чистого позитивного використання води, 100% екологічної енергії, та відсутності відходів на сміттєзвалищах у всіх виробничих підрозділах Intel. Крім того, Intel також включила один надзвичайно унікальний компонент: «спільне користування» кліматом та соціальні цілі, які вимагають співпраці з промисловістю, урядами

та громадами: революціонізувати здоров'я та безпеку за допомогою технологій; зробити технологію повністю інклюзивною та розширити цифрову готовність; досягти вуглецево-нейтрального використання техніки для вирішення проблеми зміни клімату [57].

Екологізація технологій відіграє потенційну роль у підвищенні екологічної стійкості, роблячи екологічнішим весь життєвий цикл технологій і продуктів, включаючи дослідження, виробництво, використання та утилізацію.

Екологічний дизайн – проектування енергоефективних комп’ютерів, серверів, принтерів, проекторів та інших цифрових пристроїв.

Зелене виробництво – мінімізація відходів під час виробництва комп’ютерів та інших підсистем для зменшення впливу цієї діяльності на навколишнє середовище.

Екологічне використання – мінімізація споживання електроенергії комп’ютерами та їхніми периферійними пристроями та використання їх у екологічний спосіб.

Екологічна утилізація – перепрофілювання наявного обладнання або відповідна утилізація чи переробка непотрібного електронного обладнання.

ВИСНОВКИ

Кваліфікаційна робота присвячена удосконаленню нелінійної регресійної моделі для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії, завдяки чому було підвищено достовірність оцінювання розміру веб-застосунків.

Для досягнення поставленої мети всі завдання було виконано у повному обсязі. Використовуючи методи теорії ймовірностей, математичної статистики, математичного моделювання та регресійного аналізу одержано наступні результати: виконано аналіз та порівняно моделі оцінювання розміру ПЗ, що вже існують; із джерела GitHub зібрано проекти з відкритим вихідним кодом зі створення веб-застосунків мовою Python з використанням Django Rest Framework, та обрано ті з них, що можуть бути використані для створення регресійної моделі; визначено необхідні метрики з кожного проекту та перевірено отримані дані на викиди; виконано нормалізацію даних; на основі лінійної моделі та зворотного нормалізуючого перетворення виконано побудову нелінійної регресійної моделі, розроблено програму для реалізації побудованої моделі.

За рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії, удосконалено нелінійну регресійну модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, що дозволило підвищити достовірність оцінювання та отримати кращі параметри *MMRE* та *PRED*(0,25) у порівнянні з побудованою однофакторною нелінійною регресійною моделлю та існуючою однофакторною нелінійною регресійною моделлю з використанням мови Java.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Разработка программного обеспечения – Краткое руководство. URL: https://coderlessons.com/tutorials/akademicheskii/programmnaia-inzheneriia/razrabotka-programmnogo-obespecheniia-kratkoe-rukovodstvo (дата звернення: 06.08.2022 р.).

2. Павленко П.М. Основи математичного моделювання систем і процесів : навч. посіб. Київ : Книжкове вид-во НАУ, 2013. 201 с.

3. Adam Hayes. Nonlinearity. URL: https://www.investopedia.com/terms/n/nonlinearity.asp (дата звернення: 01.08.2022 р.).

4. The software quality company TIOBE. TIOBE Index for September 2022. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 05.09.2022 р.).

5. The software quality company TIOBE. The Python Programming Language. URL: https://www.tiobe.com/tiobe-index/python/ (дата звернення: 05.09.2022 р.).

6. Приходько С.Б., Приходько Н.В., Фаріонова Т.А., Ворона М.В. Трьохфакторна нелінійна регресійна модель для оцінювання розміру PHP-застосунків із відкритим кодом. Вчені записки Таврійського національного університету імені В.І. Вернадського, Технічні науки. Київ, 2020. № 1(70) Ч.1. С. 124–131.

7. Приходько С.Б., Приходько Н.В., Ворона М.В., Бєловол І.О. Нелінійна регресійна модель для оцінювання розміру web-застосунків, що створюються з використанням фреймворку Laravel. Науково-технічний журнал «Інформаційні технології та комп’ютерна інженерія». Вінниця : ВНТУ, 2021. № 1(50). С. 115–121.

8. Prykhodko N.V., Prykhodko S.B. The non-linear regression model to estimate the software size of open-source Java-based systems. Науковий журнал

«Радіоелектроніка, інформатика, управління». Запоріжжя : ЗНТУ, 2018. № 3(46). С. 158–166.

9. Давлатова Д.Х., Макарова Л.М. Регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework. Інформаційні технології: моделі, алгоритми, системи (ITMAS-2022): Матеріали III Всеукраїнської науково-практичної інтернет-конференції (26-28 жовтня 2022 р.). Миколаїв: НУК імені адмірала Макарова, 2022. С.24-26. URL: http://itconf.nuos.edu.ua/2022/proceedings/

10. The official site of the Python Programming Language. URL: https://www.python.org/ (дата звернення: 14.08.2022 р.).

11. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. СПб. : Питер, 2017. 496 с.

12. Руководство по языку программирования Python. URL: https://metanit.com/python/tutorial/ (дата звернення: 09.10.2022 р.).

13. Get started with Django. URL: https://www.djangoproject.com/ (дата звернення: 15.08.2022 р.).

14. Руководство по веб-фреймворку Django. URL: https://metanit.com/python/django/ (дата звернення: 16.08.2022 р.).

15. Django Architecture – 3 Major Components of MVC Pattern. URL: https://metanit.com/python/django/ (дата звернення: 16.08.2022 р.).

16. Керівництво з Django REST framework. URL: https://data-flair.training/blogs/django-architecture/ (дата звернення: 18.08.2022 р.).

17. Что такое REST API и чем оно отличается от другого API. URL: https://appmaster.io/ru/blog/chto-takoe-rest-api-i-chem-ono-otlichaetsya-ot-drugogo-api (дата звернення: 16.08.2022 р.).

18. Исенко А.И. Понятия модели и моделирования в человеческой деятельности. Научно-методический электронный журнал «Концепт». 2015. № 04 (апрель). С. 31–35. URL: http://e-koncept.ru/2015/15095.htm.

19. Boehm B., Abts C. and Chulani S. Software development cost estimation approaches – A survey. Annals of Software Engineering. 2000. Vol. 10. P. 177-205.

20. Kemerer C.F. An empirical validation of software cost estimation models. Communications of the ACM. 1987. Vol. 30, № 5. P. 416-429.

21. Reifer D.J. Estimating Web Development Costs : There Are Differences. CROSSTALK, The Journal of Defense Software Engineering. 2002. P. 13-17.

22. Boehm B.W. Software engineering economicsю. New Jersey : Prentice Hall, 1981. 767 p.

23. Reifer D.J. Web development: estimating quick-to-market software. IEEE Software. 2000. Vol. 17, № 6. P. 57-64.

24. Kitchenham B., Mendes E. Why Comparative Effort Prediction Studies may be Invalid. International Conference on Predictor Models in Software Engineering (PROMISE). New York, 2009. P. 1-5.

25. Ключевые метрики для оценки эффективности вашего сайта. URL: https://outcode.ru/blog/klyuchevye-metriki-dlya-ocenki-effektivnosti-vashego-sayta (дата звернення: 05.09.2022 р.).

26. Tan H.B.K., Zhao Y., Zhang H. Estimating LOC for information systems from their conceptual data models. Proceedings of the 28th International Conference on Software Engineering (ICSE '06). Shanghai, 2006. P. 321-330.

27. Zaw T., Hlaing S.Z., Lwin M., Ochimizu K. The Measurement of Software Size based on Generation Model using COSMIC FSM. Proceedings from ICSEC’19: 23rd International Computer Science and Engineering Conference, Phuket, Thailand, 2019. P. 373-378.

28. Albrecht A.J., Gaffney J.E. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering. 1983. Vol. SE-9, № 6. P. 639-648.

29. Marco L.D., Ferrucci F., Gravino C. Approximate COSMIC Size to Early Estimate Web Application Development Effort, Software Engineering and Advanced Applications (SEAA). 39th EUROMICRO Conference, 2013. P. 349-356.

30. Meli R., Abran A., Ho V. T., Oligny S. On the applicability of COSMIC-FFP for measuring software throughout its life cycle. Proceedings of the 11th European Software Control and Metrics Conference, 2000. P. 18–20.

31. Aggarwal C.C. Outlier analysis. 2nd ed. New York : Springer International Publishing AG, 2017. 481 p.

32. Prykhodko S., Prykhodko N., Makarova L., Pukhalevych A. Application of the Squared Mahalanobis Distance for Detecting Outliers in Multivariate Non-Gaussian Data. Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). Lviv-Slavske, 2018. P. 962-965.a

33. Bates D.M., Watts D.G. Nonlinear Regression Analysis and Its Applications. Wiley, 2007. 392 p.

34. Box G.E.P., Cox D.R. An analysis of transformations. Journal of the Royal Statistical Society (B). 1964. Vol. 26, № 2. P. 211-252.

35. Yeo I., Johnson R.A. A new family of power transformations to improve normality or symmetry. Biometrika. 2000. Vol. 87, № 4. P. 954–959.

36. A code hosting platform for version control and collaboration GitHub. URL: https://github.com/ (дата звернення: 24.08.2022 р.).

37. Plugin Statistic for PyCharm. URL: https://plugins.jetbrains.com/plugin/4509-statistic (дата звернення: 24.08.2022 р.).

38. Інструкція по початку роботи з PyCharm. URL: https://py-charm.blogspot.com/2017/09/blog-post.html (дата звернення: 14.08.2022 р.).

39. Chatterjee S., Hadi A. Regression Analysis by Example. Fifth edition. New York: A John Wiley & Sons, Inc., Publication, 2012. – 421 p.

40. Макарова Л.М., Приходько Н.В., Кудін О.О. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою Java. Вісник Херсонського національного технічного університету. Херсон, 2019. № 2(69). С. 145-153.

41. Unified Modeling Language. URL: https://www.uml.org/ (дата звернення: 27.10.2022 р.).

42. Каграманова Ю. Як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 27.10.2022 р.).

43. Що таке UI та UX дизайн. URL: https://te.itstep.org/blog/ui-and-ux-design (дата звернення: 29.09.2022).

44. Язык Swift и платформы iOS и Mac OS. URL: https://metanit.com/swift/tutorial/1.1.php (дата звернення: 16.09.2022).

45. Community Overview. URL: https://www.swift.org/community/ (дата звернення: 02.10.2022 р.).

46. Swift Playgrounds. URL: https://apps.apple.com/ru/app/swift-playgrounds/id908519492 (дата звернення: 13.10.2022 р.).

47. Xcode. URL: https://apps.apple.com/ua/app/xcode/id497799835?l=ru&mt=12 (дата звернення: 28.09.2022 р.).

48. Framework UIKit. Overview. URL: https://developer.apple.com/documentation/uikit (дата звернення: 28.09.2022 р.).

49. Running your app in Simulator or on a device. URL: https://developer.apple.com/documentation/xcode/running-your-app-in-simulator-or-on-a-device (дата звернення: 04.10.2022 р.).

50. Рыбалка С.А. Диаграмма компонентов (component diagram). URL: https://portal.tpu.ru/SHARED/r/RYBALKA/academic/mct/TabMCTLect/MCTLect6.pdf (дата звернення: 28.10.2022 р.).

51. Различия между UML 1.x и UML 2.0. URL: http://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/supportingmaterials/differences_between_uml_1_x_and_uml_2_0_CA70F2E6.html (дата звернення: 27.10.2022 р.).

52. White/black/grey box-тестування. URL: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/ (дата звернення: 29.10.2022 р.).

53. Закон України «Про охорону праці» від 14.10.1992р № 2694-XII. (зі змінами від 19.08.2022). URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text.

54. Закон України «Про відходи» від 05.03.1998 № 187/98-ВР. (зі змінами від 16.10.2020). URL: https://zakon.rada.gov.ua/laws/show/187/98-%D0%B2%D1%80#Text.

55. Priyadarsini T.L., Yasmeen MD, Bharadwaj A. Green Information Technology. International Research Journal of Engineering and Technology (IRJET). 2016. Vol. 3, № 6. P. 2777–2782.

56. Prof. Gurpreet Singh and Er.Gurdeep Singh. Green computing initiatives for environmental issues. The WEI International Academic Conference Proceedings. Harvard, USA, 2015. P. 87–90.

57. 2030 RISE Strategy and Goals. URL: https://www.intel.com/content/www/us/en/corporate-responsibility/2030-goals.html (дата звернення: 09.11.2022 р.).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Повне найменування розроблюваного проекту: «Програма для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework», далі програма.

Коротка характеристика галузі застосування: програма застосовується для автоматизації розрахунків з оцінювання розміру розроблюваних мовою Python з використанням бібліотеки Django Rest Framework веб-застосунків, що дозволить підвищити ефективність роботи аналітика та менеджера проекту на початкових стадіях створення програми.

1 Підстави для розробки

Розробка виконується на підставі завдання на кваліфікаційну роботу з теми «Регресійна модель для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, та розробка програми для її реалізації», видане кафедрою ПЗАС ННІКНУП НУК ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програми є побудова лінійної та нелінійної регресійних моделей, проведення розрахунку необхідних обчислень та вивід результатів на екран на базі завантажених даних користувачем.

2.2 Експлуатаційне призначення

Програма призначена для некомерційного використання та повинна використовуватись користувачами на смартфонах з операційною системою iOS.

3. Вимоги до програмного продукту

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмний продукт має надавати можливість виконувати наступний перелік функцій:

• внесення вхідних даних про проекти у вигляді файлу з кількістю рядків коду, класів та методів різних проектів;

• застосування до внесених даних нормалізуючого перетворення у вигляді десяткового логарифму;

• застосування для внесених даних перевірки на викиди;

• побудова лінійної регресійної моделі;

• розрахунок метрик якості лінійної регресійної моделі;

• побудова нелінійної регресійної моделі з використанням зворотного нормалізуючого перетворення;

• розрахунок метрик якості нелінійної регресійної моделі;

• розрахунок довірчого інтервалу та інтервалу передбачення для нелінійної регресії;

• розрахунок оцінювання розміру веб-застосунку;

• виведення отриманих розрахунків на екран.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані мають бути організовані у вигляді файлів з розширенням .xlsx та вводяться за допомогою вибору шляху до відповідного файлу у вікні відкриття файлу операційної системи.

Вихідні дані, а саме нормалізовані дані, параметри вибірок, результати проведення викидів, побудова лінійної та нелінійної моделі, метрики якості та результат розрахунку оцінювання розміру веб-застосунку мають виводитись на екран користувача.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного функціонування програмного продукту

Стійке функціонування програми повинно забезпечуватись шляхом застосування технічних засобів та системного програмного забезпечення, що відповідають класу виконуваних завдань, а також за умови стабільного функціонування операційної системи.

Надійність програмного продукту має забезпечуватися за рахунок: надійності загальносистемного програмного забезпечення та програмного забезпечення, що розробляється Розробником; проведення комплексу заходів налагодження, пошуку та виключення помилок; веденням журналів системних повідомлень і помилок підсистем для подальшого аналізу та зміни конфігурації.

У випадку виникненні збоїв в роботі програмного продукту, для відновлення нормальної роботи необхідно здійснити перезавантаження операційної системи та самої програми.

3.2.2 Вимоги до контролю вхідної та вихідної інформації

Програмне забезпечення має перевіряти вхідні дані на коректність. У випадку завантаження користувачем файлу неправильного типу користувач має отримати повідомлення про помилку.

3.2.3 Вимоги до часу відновлення після відмови

Для відновлення після відмови потрібен час, який складається з часу перезавантаження операційної системи, перезавантаження самої програми, та повторного введення вхідних даних.

3.3 Вимоги до умов експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації повинні задовольняти вимогам до технічних засобів:

• температура повітря у приміщенні: від +10°C до +35°C;

• відносна вологість повітря: 40-60%;

• атмосферний тиск: 630-800 мм ртутного стовпчика;

• запиленість повітря: не більше ніж 0,75 мг/м3;

3.3.2 Вимоги до видів обслуговування

Програма не потребує обслуговування.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

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

Користувач повинен мати навички роботи зі смартфоном, та мати змогу працювати з інтерфейсами мобільних додатків.

3.4 Вимоги до складу і параметрів технічних засобів

Для реалізації мобільного додатку було обрано такі компоненти:

- macOS – в якості операційної системи;
- Xcode – в якості середовища розробки;
- Swift 5 – в якості мови програмування;

Даний програмний продукт вимагає технічний засіб – персональний комп’ютер, ноутбук та мобільний телефон на iOS з версією починаючи від 13.0 та новіші з можливістю виходу в мережу інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Розроблений програмний продукт орієнтований на роботу на смартфонах з операційною системою iOS починаючи від версії 13.0 та новіші. Тому для коректної роботи програми необхідне стабільне функціонування операційної системи.

3.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4 Вимоги до програмної документації

Програмна документація повинна містити: технічне завдання на розробку програми; текст програми; опис програми; інструкція користувача; програма і методика випробувань.

5 Техніко-економічні показники

Економічна ефективність впровадження програми розрахована у розділі 5. Згідно з отриманими результатами, розробка та впровадження розроблюваної програми є доцільною, а термін її окупності становить приблизно 3 місяці.

6 Стадії та етапи розробки

Стадії та етапи розробки представлені в таблиці А.1.

7 Порядок прийому та контролю

Система піддається випробуванням таких видів: попередні випробування; дослідна експлуатація; приймальні випробування.

Склад, обсяг та методи попередніх випробувань системи визначаються документом «Програма та методика випробувань», що розробляється на стадії «Робоча документація».

Склад, обсяг та методи дослідної експлуатації системи визначаються документом «Програма дослідної експлуатації», що розробляється на стадії «Введення в дію».

Склад, обсяг та методи приймальних випробувань системи визначаються документом «Програма та методика випробувань», що розробляється на стадії «Введення в дію» з урахуванням результатів проведення попередніх випробувань та дослідної експлуатації.

Таблиця А.1 – Стадії та етапи розробки

№	Стадії розробки	Етапи розробки	Зміст роботи по етапах	Термін виконання
1	2	3	4	5
1	Технічне завдання	Розробка, узгодження та затвердження технічного завдання	Постановка задачі; обґрунтування необхідності проведення досліджень; визначення вимог до програми, стадій, етапів і термінів розробки програми; розробка технічного завдання та його затвердження	Початок: 09.09.22 Кінець: 28.09.22
2	Ескізний проект	Розробка та затвердження ескізного проекту	Виявлення основних вимог, розробка структури та алгоритму поведінки майбутньої системи, розробка ескізного проекту програмного продукту та його затвердження	Початок: 30.09.22 Кінець: 07.10.22
3	Технічний проект	Розробка та затвердження технічного проекту	Уточнення алгоритму рішення задачі та структури програми, розробка технічного проекту та його затвердження	Початок: 10.10.22 Кінець: 19.10.22
4	Робочий проект	Розробка та затвердження робочого проекту	Розробка програмного продукту та його документації, розробка інструкції користувача, розробка робочого проекту та його затвердження	Початок: 20.10.22 Кінець: 01.11.22

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

DataHandler.swift

```
import UIKit
import Foundation
import simd
import Accelerate

class DataHandler {
  static let shared = DataHandler()

  var ishodniiStruct: EvaluationData?
  var normalStructZ: EvaluationData?
  var linearReg: LinearRegr?

  init() {

  }

  func start(dataY: [Double], dataX1: [Double], dataX2: [Double]) {
    self.ishodniiStruct = EvaluationData(dataY: dataY, dataX1: dataX1, dataX2: dataX2)
    self.ishodniiStruct?.start()

    if let ishodniiStruct = self.ishodniiStruct {
      let dataZY = perevodNaLog10s(items: ishodniiStruct.dataY)
      let dataZX1 = perevodNaLog10s(items: ishodniiStruct.dataX1)
      let dataZX2 = perevodNaLog10s(items: ishodniiStruct.dataX2)
      normalStructZ = EvaluationData(dataY: dataZY, dataX1: dataZX1, dataX2: dataZX2)
      normalStructZ?.start()
      regresiaStart()
    }
  }

  func perevodNaLog10s(items: [Double]) -> [Double] {
    var newMass = [Double]()
    for item in items {
      let log10 = log10(item)
      newMass.append(log10)
    }
    return newMass
  }

  func regresiaStart() {
    if let ishodniiStruct = self.ishodniiStruct, let normalStructZ = self.normalStructZ {
      let ishodniiSt = EvaluationData(dataY: ishodniiStruct.dataY, dataX1:
ishodniiStruct.dataX1, dataX2: ishodniiStruct.dataX2)
      let normalStZ = EvaluationData(dataY: normalStructZ.dataY, dataX1:
normalStructZ.dataX1, dataX2: normalStructZ.dataX2)
```

```
      linearReg = LinearRegr(ishodniiStruct: ishodniiSt, normalStructZ: normalStZ)
      linearReg?.start()
    }
  }
}
```

EvaluationData.swift

```
import Foundation
import UIKit

class EvaluationData {

  var N: Int = 0
  var m: Int = 0

  var dataY: [Double] = []
  var dataX1: [Double] = []
  var dataX2: [Double] = []

  var viborochSredY: Double = 0.0
  var viborochSredX1: Double = 0.0
  var viborochSredX2: Double = 0.0

  var disperciaY: Double = 0.0
  var disperciaX1: Double = 0.0
  var disperciaX2: Double = 0.0

  var skvY: Double = 0.0
  var skvX1: Double = 0.0
  var skvX2: Double = 0.0

  var maxY: Double = 0.0
  var maxX1: Double = 0.0
  var maxX2: Double = 0.0

  var minY: Double = 0.0
  var minX1: Double = 0.0
  var minX2: Double = 0.0

  var stepY: Double = 0.0
  var stepX1: Double = 0.0
  var stepX2: Double = 0.0

  init(dataY: [Double], dataX1: [Double], dataX2: [Double]) {
    self.dataY = dataY
    self.dataX1 = dataX1
    self.dataX2 = dataX2
  }

  func start() {
```

```
  viborochnoeSrednie()
}

func viborochnoeSrednie() {
  N = dataY.count
  viborochSredY = dataY.reduce(0.0, +) / Double(N)
  viborochSredX1 = dataX1.reduce(0.0, +) / Double(N)
  viborochSredX2 = dataX2.reduce(0.0, +) / Double(N)

  dispercia()
}

func dispercia() {
  disperciaY = disperciaS(viborSred: viborochSredY, mas: dataY)
  disperciaX1 = disperciaS(viborSred: viborochSredX1, mas: dataX1)
  disperciaX2 = disperciaS(viborSred: viborochSredX2, mas: dataX2)

  skv()
}

func disperciaS(viborSred: Double, mas: [Double]) -> Double {
  var test:Double = 0.0
  for item in mas {
    let shag1 = pow(item - viborSred, 2)
    test += shag1
  }
  return test / Double(N - 1)
}

func skv() {
  skvY = sqrt(disperciaY)
  skvX1 = sqrt(disperciaX1)
  skvX2 = sqrt(disperciaX2)

  kolInterval()
}

func kolInterval() {
  m = Int(3.3 * log10(Double(N)) + 1)

  searchMinAndMax()
}

func searchMinAndMax() {
  maxY = dataY.max()!
  maxX1 = dataX1.max()!
  maxX2 = dataX2.max()!

  minY = dataY.min()!
  minX1 = dataX1.min()!
  minX2 = dataX2.min()!
  shirinaPodintervala()
```

```
  }

  func shirinaPodintervala() {
    stepY = (maxY - minY) / Double(m)
    stepX1 = (maxX1 - minX1) / Double(m)
    stepX2 = (maxX2 - minX2) / Double(m)
  }
}
```

FileReader.swift

```
import Foundation
import CoreXLSX

class FileReader {

  init() {
  }

  func getDataFromFile(name: String, onSuccess: @escaping ()->()) {
    if let file = XLSXFile(filepath: name) {
      do {
        for wbk in try file.parseWorkbooks() {
          for (name, path) in try file.parseWorksheetPathsAndNames(workbook: wbk) {
            if let worksheetName = name {
              print("This worksheet has a name: \(worksheetName)")
            }

            let worksheet = try file.parseWorksheet(at: path)
            if let sharedStrings = try file.parseSharedStrings() {
              var columnBtrings = worksheet.cells(atColumns: [ColumnReference("B")!])
                .compactMap { $0.stringValue(sharedStrings) }

              var dataY: [Double] = []
              columnBtrings.removeFirst()
              for item in columnBtrings {
                if let itemDouble = Double(item) {
                  let privedenni = itemDouble / 1000
                  dataY.append(privedenni)
                }
              }

              var columnCtrings = worksheet.cells(atColumns: [ColumnReference("C")!])
                .compactMap { $0.stringValue(sharedStrings) }

              columnCtrings.removeFirst()
              var dataX1: [Double] = []
              for item in columnCtrings {
                if let itemDouble = Double(item) {
                  dataX1.append(itemDouble)
                }
```

```
            }

            var columnDtrings = worksheet.cells(atColumns: [ColumnReference("D")!])
              .compactMap { $0.stringValue(sharedStrings) }

            columnDtrings.removeFirst()
            var dataX2: [Double] = []
            for item in columnDtrings {
              if let itemDouble = Double(item) {
                dataX2.append(itemDouble)
              }
            }

            DataHandler.shared.start(dataY: dataY, dataX1: dataX1, dataX2: dataX2)

            onSuccess()
          }
        }
      }
    } catch {
    }
  }
 }
}
```

LinearRegr.swift

```
import UIKit
import Foundation
import simd
import Accelerate

class LinearRegr {

  var ishodniiStruct: EvaluationData
  var normalStructZ: EvaluationData
  var nonLinearRegresion: NonLinearRegresia?

  var zyZySred: [Double] = []
  var zx1Zx1Sred: [Double] = []
  var zx2Zx2Sred: [Double] = []

  var zMinusZsread: [[Double]] = [[Double]]()
  var zMinusZsreadSquare: [[Double]] = [[Double]]()
  var shag3Matrix: [[Double]] = [[Double]]()
  var covMatrix: [[Double]] = [[Double]]()
  var covObrMatrix: [[Double]] = [[Double]]()
  var zMinusZsreadUmnozCovObr: [[Double]] = [[Double]]()
  var di2: [Double] = [Double]()
  var tsWithDi2: [Double] = [Double]()
```

```
var xi2 = 7.81472790325118
var f = 2.73950230196601
var colichestvoVibrosov = -1
var colCicleVibrosov = 0

var coefB = B0B1B2()

var metrikiYakos = MetricYakosty()

init(ishodniiStruct: EvaluationData, normalStructZ: EvaluationData) {
  self.ishodniiStruct = ishodniiStruct
  self.normalStructZ = normalStructZ
}

func start() {
  zYminusZySred()
  zMinusZsreads()
  zMinusZsreadSquares()
  shag3Matrixs()
  covMatrixs()
  vibrosyCalculation()
}

func vibrosyCalculation() {
  var di2MaxIndex: Set<Int> = []
  var di2Index = 0
  for item in di2 {
    if item >= xi2 {
      di2MaxIndex.insert(di2Index)
    }
    di2Index += 1
  }

  di2Index = 0
  for item in tsWithDi2 {
    if item >= f {
      di2MaxIndex.insert(di2Index)
    }
    di2Index += 1
  }

  colichestvoVibrosov = di2MaxIndex.count

  let masReserved = di2MaxIndex.sorted().reversed()
  for index in masReserved {

    ishodniiStruct.dataY.remove(at: index)
    ishodniiStruct.dataX1.remove(at: index)
    ishodniiStruct.dataX2.remove(at: index)
  }

  if masReserved.count != 0 {
```

```
            self.ishodniiStruct.start()
            let ishodniiStruct = self.ishodniiStruct
            let dataZY = MathRegression.shared.perevodNaLog10s(items: ishodniiStruct.dataY)
            let dataZX1 = MathRegression.shared.perevodNaLog10s(items:
ishodniiStruct.dataX1)
            let dataZX2 = MathRegression.shared.perevodNaLog10s(items:
ishodniiStruct.dataX2)
            normalStructZ = EvaluationData(dataY: dataZY, dataX1: dataZX1, dataX2: dataZX2)
            normalStructZ.start()

            colCicleVibrosov += 1

            start()
           } else {
            let b = MathRegression.shared.calculationCoefficient(zX1: normalStructZ.dataX1,
zX2: normalStructZ.dataX2, zY: normalStructZ.dataY)
            coefB.b0 = b[0]
            coefB.b1 = b[1]
            coefB.b2 = b[2]

            let noLinearRegresion = NoLinearRegresion(zY: normalStructZ.dataY,
                                      zX1: normalStructZ.dataX1,
                                      zX2: normalStructZ.dataX2,
                                      y: ishodniiStruct.dataY,
                                      b0: coefB.b0,
                                      b1: coefB.b1,
                                      b2: coefB.b2,
                                      viborochSredY: normalStructZ.viborochSredY)

            metrikiYakos = MathRegression.shared.podscheOcenokKachestva(data:
noLinearRegresion)

            nonLinearRegresion = NonLinearRegresia(linear: self)
            nonLinearRegresion?.start()
           }
          }

          func zYminusZySred() {

           let viborochSredY = normalStructZ.viborochSredY
           let viborochSredX1 = normalStructZ.viborochSredX1
           let viborochSredX2 = normalStructZ.viborochSredX2

            zyZySred = normalStructZ.dataY.map({ $0 - viborochSredY })
            zx1Zx1Sred = normalStructZ.dataX1.map({ $0 - viborochSredX1 })
            zx2Zx2Sred = normalStructZ.dataX2.map({ $0 - viborochSredX2 })
          }

          func zMinusZsreads() {
           let size = normalStructZ.dataY.count
           var result = [[Double]](
            repeating: [Double] (repeating: 0, count: size),
```

```
    count: 3
  )

  for k in 0..<normalStructZ.dataY.count  {
    result[0][k] = normalStructZ.dataY[k] - normalStructZ.viborochSredY
  }
  for k in 0..<normalStructZ.dataX1.count {
    result[1][k] = normalStructZ.dataX1[k] - normalStructZ.viborochSredX1
  }
  for k in 0..<normalStructZ.dataX2.count  {
    result[2][k] = normalStructZ.dataX2[k] - normalStructZ.viborochSredX2
  }
  zMinusZsread = result
}

func zMinusZsreadSquares() {
  let size = zMinusZsread[0].count

  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: size),
    count: size
  )

  for k in 0..<3 {
    for i in 0..<size {
      result[k][i] = pow(zMinusZsread[k][i], 2)
    }
  }
  zMinusZsreadSquare = result
}

func shag3Matrixs() {
  let size = zMinusZsread[0].count

  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: size),
    count: size
  )
  for i in 0..<size {
    result[0][i] = zMinusZsread[0][i] * zMinusZsread[1][i]
  }
  for i in 0..<size {
    result[1][i] = zMinusZsread[0][i] * zMinusZsread[2][i]
  }

  for i in 0..<size {
    result[2][i] = zMinusZsread[1][i] * zMinusZsread[2][i]
  }

  shag3Matrix = result
}
```

```
        func covMatrixs() {
          covMatrix = zMinusZsreadSquareCalculation()

          covObrMatrix = MathRegression.shared.invert(matrix3x3: covMatrix)

          zMinusZsreadUmnozCovObr = MathRegression.shared.multiplyDiana(zMinusZsread,
covObrMatrix)

          di2 = di2Calculation()

          let N = Double(zMinusZsreadSquare.count)
          var result = [Double]()
          for item in di2 {
            let ts = (N - 3.0) * N * item / ((pow(N, 2) - 1.0) * 3.0)
            result.append(ts)
          }

          tsWithDi2 = result
        }

        func di2Calculation() -> [Double] {
          var viborkaDi2: [Double] = [Double]()
          for i in 0..<zMinusZsreadUmnozCovObr[0].count {
            var maszMinusZsreadUmnozCovObr = [Double]()
            for j in 0..<3 {
              maszMinusZsreadUmnozCovObr.append(zMinusZsreadUmnozCovObr[j][i])
            }

            var maszMinusZsread = [Double]()
            for j in 0..<3 {
              maszMinusZsread.append(zMinusZsread[j][i])
            }

            let transp = MathRegression.shared.transposeMatrix([maszMinusZsread])
            let multi = MathRegression.shared.multiply([maszMinusZsreadUmnozCovObr],
transp)
            viborkaDi2.append(multi[0][0])
          }

          return viborkaDi2
        }

        func zMinusZsreadSquareCalculation() -> [[Double]] {
          var result = [[Double]](
            repeating: [Double] (repeating: 0, count: 3),
            count: 3
          )

          let N = zMinusZsreadSquare.count

          result[0][0] = zMinusZsreadSquare[0].reduce(0.0, +) / Double(N)
          result[0][1] = shag3Matrix[0].reduce(0.0, +) / Double(N)
```

```
    result[0][2] = shag3Matrix[1].reduce(0.0, +) / Double(N)

    result[1][0] = shag3Matrix[0].reduce(0.0, +) / Double(N)
    result[1][1] = zMinusZsreadSquare[1].reduce(0.0, +) / Double(N)
    result[1][2] = shag3Matrix[2].reduce(0.0, +) / Double(N)

    result[2][0] = shag3Matrix[1].reduce(0.0, +) / Double(N)
    result[2][1] = shag3Matrix[2].reduce(0.0, +) / Double(N)
    result[2][2] = zMinusZsreadSquare[2].reduce(0.0, +) / Double(N)

    return result
  }
}
```

MathRegression.swift

```
import UIKit
import Foundation
import simd
import Accelerate

struct NoLinearRegresion {
  var zY: [Double]
  var zX1: [Double]
  var zX2: [Double]

  var y: [Double]

  var b0: Double
  var b1: Double
  var b2: Double

  var viborochSredY: Double
}

struct B0B1B2 {
  var b0: Double = 0.0
  var b1: Double = 0.0
  var b2: Double = 0.0
}

struct MetricYakosty {
  var mMRE: Double = 0.0
  var pred0_25: Double = 0.0
  var r2: Double = 0.0
}

class MathRegression {
  static let shared = MathRegression()
```

```
        func podscheOcenokKachestva(data: NoLinearRegresion, isNoRegrasion: Bool = false)
-> MetricYakosty {
        let zY = isNoRegrasion ? data.y : data.zY

        var zYRashitaniy = [Double]()
        var reznicaZy = [Double]()
        var mRE = [Double]()

        var reznicaZy2 = [Double]()
        var zMinusZsred2 = [Double]()

        for i in 0..<zY.count {
          var result = data.b0 + data.b1 * data.zX1[i] + data.b2 * data.zX2[i]
          if isNoRegrasion {
            result = pow(10,result)
            zYRashitaniy.append(pow(10,result))
          } else {
            zYRashitaniy.append(result)
          }

          let result2 = zY[i] - result
          reznicaZy.append(result2)
          reznicaZy2.append(pow(result2, 2))

          let result3 = abs(result2 / zY[i])
          mRE.append(result3)

          zMinusZsred2.append(pow(zY[i] - data.viborochSredY,2))
        }

        let N = mRE.count
        let mMRE = mRE.reduce(0.0, +) / Double(N)

        var pred = [Double]()

        for i in 0..<mRE.count {
          if mRE[i] <= 0.25 {
            pred.append(1)
          } else {
            pred.append(0)
          }
        }

        let Ns = pred.count
        let pred0_25 = pred.reduce(0.0, +) / Double(Ns)

        let sumReznicaZy2 = reznicaZy2.reduce(0.0, +)
        let sumZMinusZsred2 = zMinusZsred2.reduce(0.0, +)

        let r2 = 1.0 - (sumReznicaZy2 / sumZMinusZsred2)

        return MetricYakosty(mMRE: mMRE, pred0_25: pred0_25, r2: r2)
```

```
        }

        func calculationCoefficient(zX1: [Double], zX2: [Double], zY: [Double]) -> [Double] {
          let size = zX1.count

          var matrixTranspo = [[Double]](
            repeating: [Double] (repeating: 0, count: size),
            count: 3
          )

          for k in 0..<size  {
            matrixTranspo[0][k] = 1
          }
          for k in 0..<size {
            matrixTranspo[1][k] = zX1[k]
          }
          for k in 0..<size  {
            matrixTranspo[2][k] = zX2[k]
          }

          let resultOriginal = MathRegression.shared.transposeMatrix(matrixTranspo)

          let multyplayMatrix = MathRegression.shared.multiply(matrixTranspo, resultOriginal)

          let obrMultyMatrix = MathRegression.shared.invert(matrix3x3: multyplayMatrix)

          let zyNaTranspo = MathRegression.shared.multiplyDi2(matrixTranspo, zY)
          var vectorZyNaTranspo: [Double] = [Double]()
          vectorZyNaTranspo.append(zyNaTranspo[0][0])
          vectorZyNaTranspo.append(zyNaTranspo[1][0])
          vectorZyNaTranspo.append(zyNaTranspo[2][0])

          let reschetKoef = MathRegression.shared.multiplyDi2(obrMultyMatrix,
vectorZyNaTranspo)

          let b0 = reschetKoef[0][0]
          let b1 = reschetKoef[1][0]
          let b2 = reschetKoef[2][0]

          return [b0, b1, b2]
        }

        func multiplyDiana(_ matrixA:[[Double]], _ matrixB:[[Double]]) -> [[Double]] {

          if matrixA.count != matrixB.count {
            print("Fail")
            return [[]]
          }

          let size = matrixA[0].count

          var result = [[Double]](
```

```
    repeating: [Double] (repeating: 0, count: size),
    count: matrixA.count
  )

  for i in 0 ..< result.count {
    for j in 0 ..< matrixA[0].count {
      for k in 0 ..< matrixB.count {
        let teste = matrixA[k][j]
        let trst = matrixB[i][k]
        let finish = teste * trst
        result[i][j] += finish
      }
    }
  }

  return result
}

func transposeMatrix(_ matrix: [[Double]]) -> [[Double]] {
  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: matrix.count),
    count: matrix[0].count
  )
  for i in 0..<matrix.count {
    for k in 0..<matrix[0].count {
      result[k][i] = matrix[i][k]
    }
  }
  return result
}

func multiply(_ matrixA:[[Double]], _ matrixB:[[Double]]) -> [[Double]] {

  if matrixA[0].count != matrixB.count {
    print("Fail")
    return [[]]
  }

  let size = matrixA.count

  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: size),
    count: size
  )

  for i in 0 ..< result.count {
    for j in 0 ..< matrixA.count {
      for k in 0 ..< matrixB.count {
        result[i][j] += matrixA[i][k] * matrixB[k][j]
      }
    }
  }
```

```
  return result
}

func multiplyDi2(_ matrixA:[[Double]], _ matrixB:[Double]) -> [[Double]] {

  if matrixA[0].count != matrixB.count {
    print("Fail")
    return [[]]
  }

  let size = matrixA.count

  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: 1),
    count: size
  )

  for i in 0 ..< result.count {
      for k in 0 ..< matrixB.count {
        result[i][0] += matrixA[i][k] * matrixB[k]
      }
  }

  return result
}

func invert(matrix3x3 :[[Double]]) -> [[Double]] {

  let matrix: [Double] = matrix3x3[0] + matrix3x3[1] + matrix3x3[2]
  var inMatrix = matrix
  var N = __CLPK_integer(sqrt(Double(matrix.count)))
  var pivots = [__CLPK_integer](repeating: 0, count: Int(N))
  var workspace = [Double](repeating: 0.0, count: Int(N))
  var error : __CLPK_integer = 0

  withUnsafeMutablePointer(to: &N) {
    dgetrf_($0, $0, &inMatrix, $0, &pivots, &error)
    dgetri_($0, &inMatrix, $0, &pivots, &workspace, $0, &error)
  }

  return transposeMatrix22222(inMatrix)
}

func transposeMatrix22222(_ matrix: [Double]) -> [[Double]] {
  var result = [[Double]](
    repeating: [Double] (repeating: 0, count: 3),
    count: 3
  )
  var index = 0
  for i in 0..<3 {
    for k in 0..<3 {
```

```
            result[i][k] = matrix[index]
            index += 1
          }
        }
        return result
      }

      func perevodNaLog10s(items: [Double]) -> [Double] {
        var newMass = [Double]()
        for item in items {
          let log10 = log10(item)
          newMass.append(log10)
        }
        return newMass
      }
    }
```

NonLinearRegresia.swift

```
    import Foundation

    class NonLinearRegresia {

      var linear: LinearRegr

      var metrikiYakos: MetricYakosty?

      init(linear: LinearRegr) {
        self.linear = linear
      }

      func start() {
        evaluation()
      }

      func evaluation() {
        let noLinearRegresion = NoLinearRegresion(zY: linear.normalStructZ.dataY,
                                                  zX1: linear.normalStructZ.dataX1,
                                                  zX2: linear.normalStructZ.dataX2,
                                                  y: linear.ishodniiStruct.dataY,
                                                  b0: linear.coefB.b0,
                                                  b1: linear.coefB.b1,
                                                  b2: linear.coefB.b2,
                                                  viborochSredY: linear.ishodniiStruct.viborochSredY)
        print(noLinearRegresion)
        metrikiYakos = MathRegression.shared.podscheOcenokKachestva(data:
noLinearRegresion, isNoRegrasion: true)
      }
    }
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1 Загальні відомості

Програма призначена для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework за рахунок побудови лінійної та нелінійної регресійних моделей. Даний програмний продукт забезпечує оптимізацію проведення розрахунків та виконує оцінювання розміру програмного проекту.

Програмний продукт реалізовано з використанням мови програмування Swift 5.

Надійне функціонування програми має бути забезпечено стійким функціонуванням мобільного пристрою.

2 Функціональне призначення

Даний програмний продукт призначений для забезпечення можливість виконання нижчезазначених функцій:

- внесення користувачем даних про програмні проекти;
- виконання нормалізації внесених даних;
- виконання перевірки на викиди для нормалізованих даних;
- визначення коефіцієнтів для побудови рівняння регресії;
- побудови лінійної та нелінійної регресійних моделей;
- розрахунок метрик якості лінійної та нелінійної регресійних моделей;
- розрахунок довірчого інтервалу та інтервалу передбачення для нелінійної регресії;
- оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.

3 Опис логічної структури

Програмний продукт розроблено із використанням шаблону проектування MVC (Model-View-Controller). У рамках цього шаблону проектування програми поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами – модель, представлення та контроллер.

4 Використання технічних засобів

Система повинна відповідати вимогам до інформаційної та програмної сумісності. Для повноцінного функціонування програмного продукту необхідно використовувати операційну систему iOS не нижче версії 13.0.

Для можливості виконання модифікації або подальшого покращення реалізованого програмного продукту необхідно використовувати наступні технічні та програмні засоби:

- Мікропроцесор класу M1 та вище;
- Оперативна пам'ять RAM 8 Гб та більше;
- Жорсткий диск з вільним місцем 50 Гб та більше;

Вимоги до програмного забезпечення ПК: macOS Monterey 12.3 та вище.

5 Виклик та завантаження

Для запуску та роботи з програмним продуктом необхідна операційна система iOS 13.0 та вище.

Програмний продукт запускається за допомогою відповідної іконки, що знаходиться на головному екрані смартфону. Програма має один режим використання.

6 Вхідні дані

Вхідні дані для програми мають вводитись у вигляді файлу, який містить наступні стовпці значень: номер проекту, кількість рядків коду, кількість класів, кількість методів.

Вхідними даними для оцінювання розміру веб-застосунку слугують кількість класів та кількість методів, які користувач вносить у відповідні поля форми.

7 Вихідні дані

Вихідними даними є результат обчислень у вигляді числових значень, що відображаються на екрані користувача.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА

Для запуску та роботи з програмним продуктом необхідно запустити програму за допомогою відповідної іконки, що знаходиться на головному екрані смартфону. Після запуску користувач має вибрати файл з даними для завантаження, що наведено на рисунку Г.1, і у випадку коректності формату вибраного файлу з’являється інтерфейс, що наведений на рисунку Г.2, із обробленими з файлу даними. Також на рисунку Г.1 наведено повідомлення щодо некоректності обраного користувачем формату файлу, у разі коли користувач намагається обрати формат відмінний від формату .xlsx.

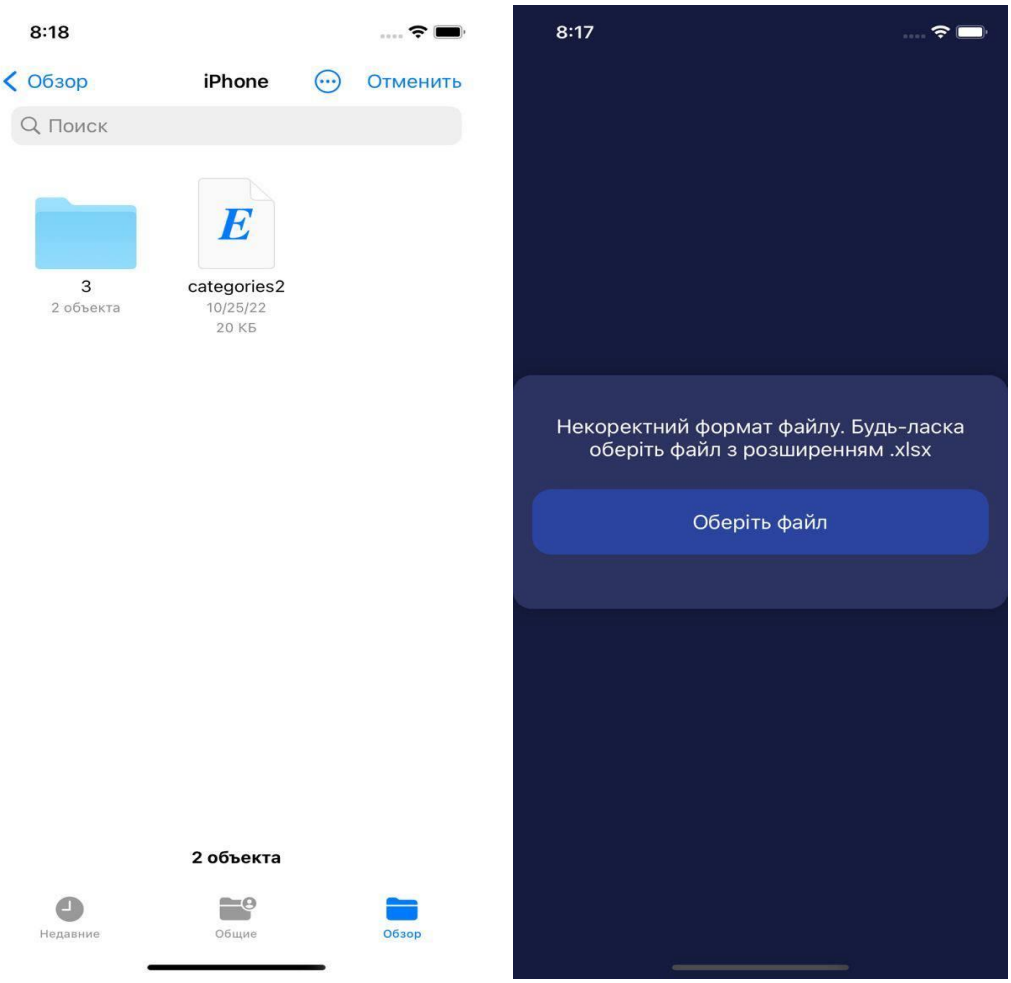

Рисунок Г.1 – Завантаження вхідних даних користувачем

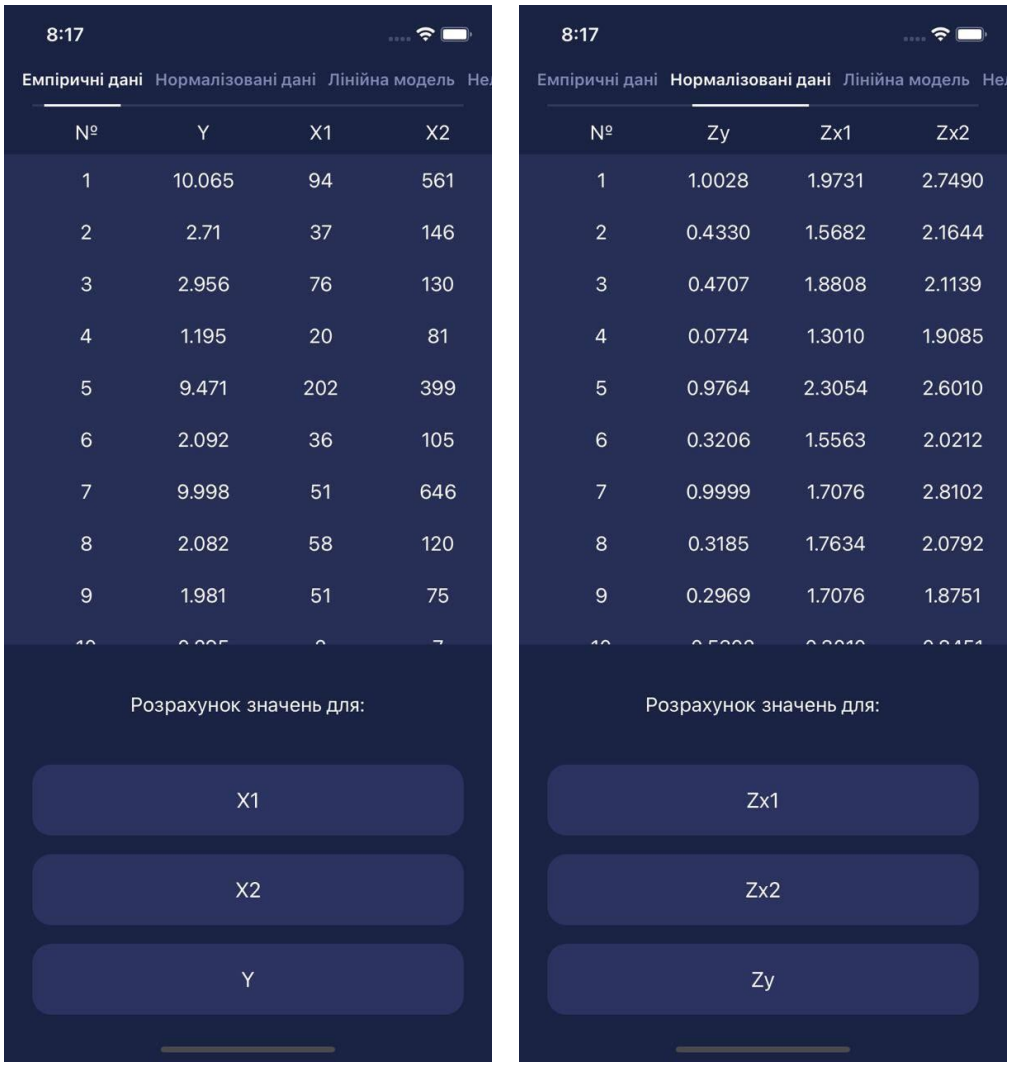

Рисунок Г.2 – Екрани з емпіричними та нормалізованими даними

На екранах з емпіричними та нормалізованими даними користувач має можливість переглянути розраховані програмою значення вибіркового середнього, дисперсії, середньоквадратичного відхилення, мінімального та максимального значеннь з вибірки даних за обраною змінною. На рисунку Г.3 продемонстровано виведення розрахованих значень за вибіркою емпіричних та нормалізованих даних за обраною зміною Y та Z_{X_1} відповідно.

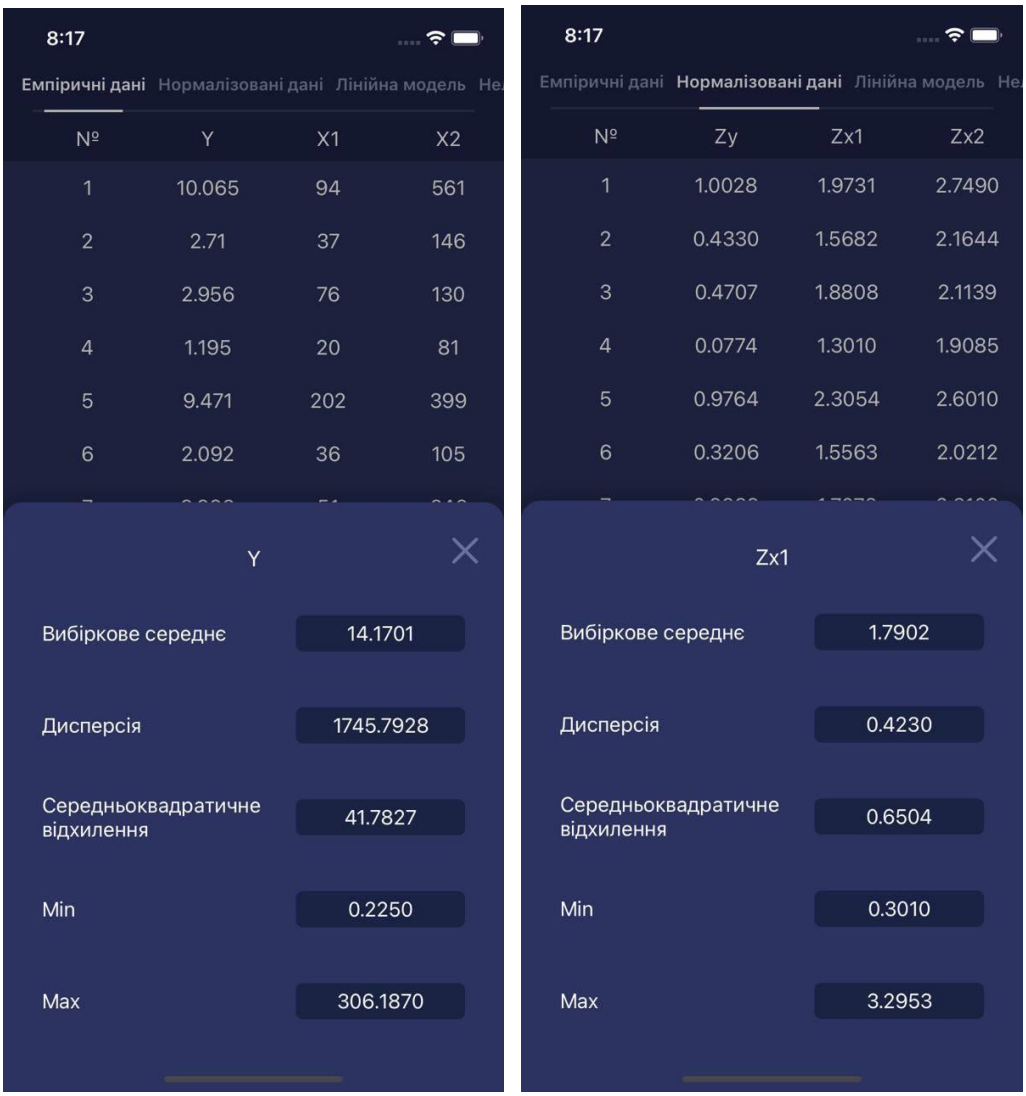

Рисунок Г.3 – Випадаюче меню з розрахованими значеннями вибірки за обраною змінною

Перейшовши до екрану лінійної моделі користувач може виконати перевірку даних на викиди, побудувати лінійну регресійну модель та розрахувати метрики якості прогнозування побудованої лінійної регресійної моделі, що зображено на рисунку Г.4.

Рисунок Г.4 – Екран побудови лінійної регресійної моделі

Відповідним чином перейшовши до екрану нелінійної моделі користувач може виконати побудову нелінійну регресійної модель та розрахувати метрики якості прогнозування побудованої нелінійної регресійної моделі, що зображено на рисунку Г.5. Також користувач може оцінити розмір будь-якого веб-застосунку, розробленого мовою Python з використанням Django Rest Framework, при натисканні на кнопку оцінювання розміру веб-застосунку. В результаті цієї дії користувач перейде до нового екрану, що зображено на рисунку Г.6, з безпосереднім розрахунком розміру програмного проекту за введеними кількістю класів та методів проекту, що цікавить користувача.

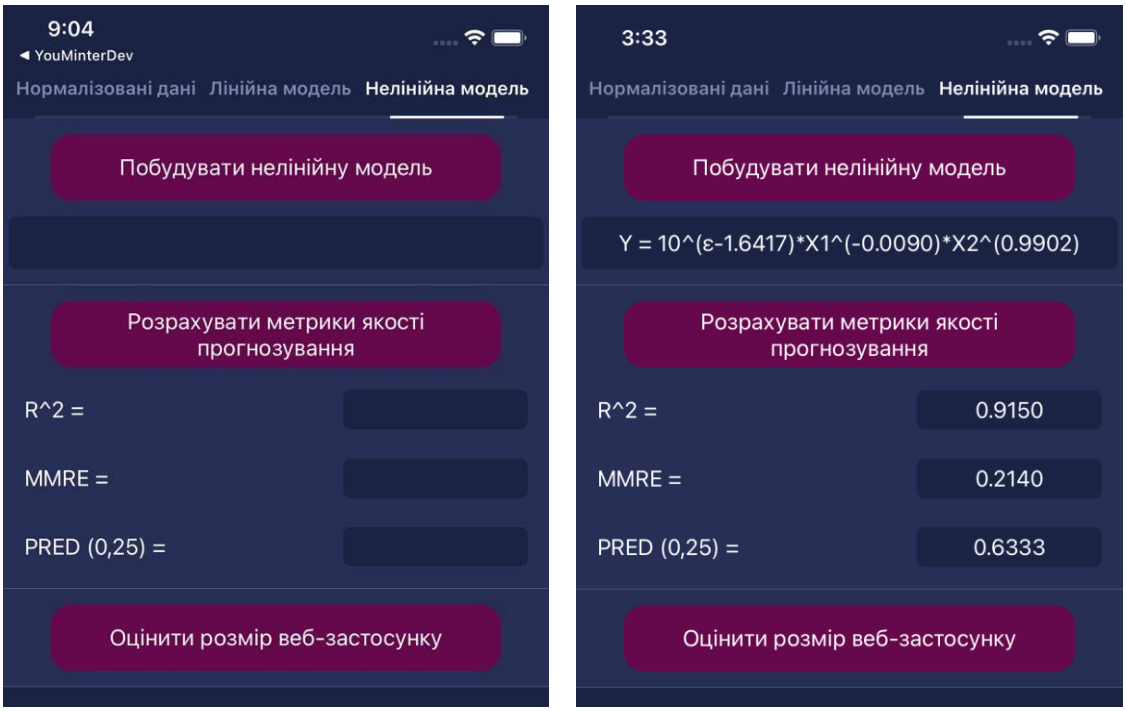

Рисунок Г.5 – Екран побудови нелінійної регресійної моделі

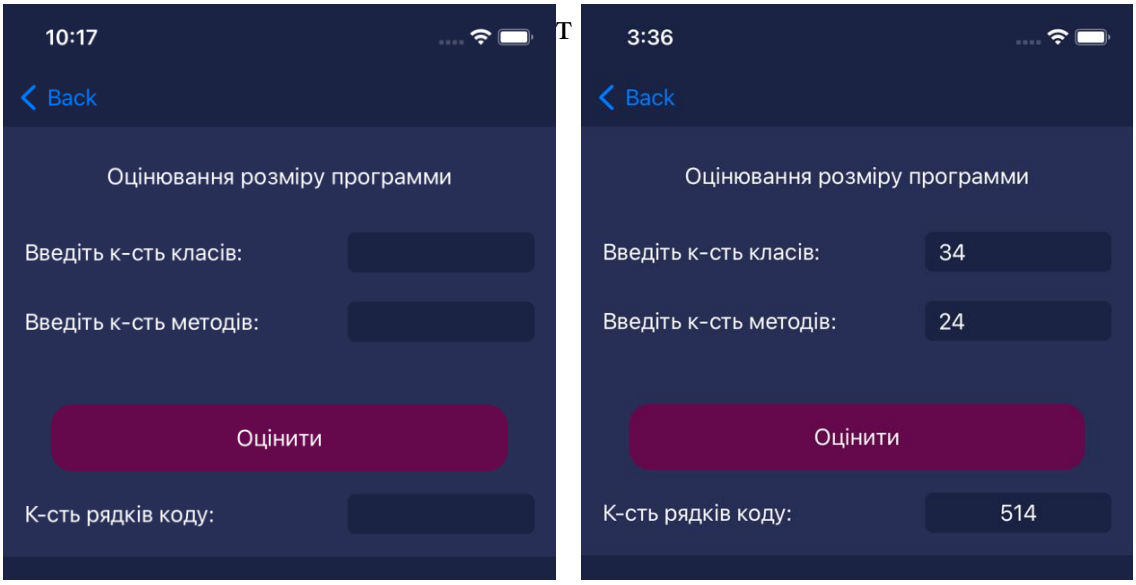

Рисунок Г.6 – Екран оцінювання розміру програмного проекту

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБОВУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програма для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework, розроблена мовою Swift в межах даної кваліфікаційної роботи.

2 Мета випробувань

Метою проведення випробувань є перевірка працездатності та функціональних можливостей програми на відповідність вимогам технічного завдання.

3 Вимоги до програми

При проведенні випробувань необхідно виконати перевірку на відповідність функціональних характеристик розробленої програми вимогам, що викладені у пункті «Вимоги до складу виконуваних функцій» Технічного завдання (Додаток А).

4 Вимоги до програмної документації

У комплект програмної документації повинні входити наступні документи:

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

5 Засоби та порядок випробувань

Вимоги до складу та параметрів технічних і програмних засобів вказані у Технічному завданні (Додаток А). Випробування проводилось на пристрою з операційною системою iOS 16.0.

Порядок проведення випробувань передбачає виконання функцій та проведення аналізу отриманих результатів. Необхідно перевірити відповідність функціональних характеристик розроблюваної програми вимогам, що викладені у програмній документації.

6 Методи випробувань

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» технічного завдання.

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

В процесі тестування була перевірена функціональність програми для оцінювання розміру веб-застосунків, що створюються мовою Python з використанням Django Rest Framework.