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: Нелінійна регресійна модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, та розробка програми для її реалізації

Виконав: студент 6151м групи
Бринзей О.А.
(підпис, ПІБ)

Керівник роботи:
доцент, к.ф.-м.н., доцент
(посада, науковий ступень вчене звання)

Латанська Л.О.
(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 Бринзею Олександру Андрійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Нелінійна регресійна модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, та розробка програми для її реалізації».

Керівник роботи Латанська Л. О.

Затверджені наказом ректора № 798 уч від «28» _____09_____ 2022 р.

2. Термін подання роботи: 05.12.2022 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 1.Тема 2. Актуальність теми, 3. Мета і завдання дослідження, 4. Об'єкт та предмет дослідження, 5. Методи дослідження, 6. Наукова новизна одержаних результатів, 7. Практичне значення одержаних результатів, 8. Апробація результатів досліджень та публікації, 9. Пошук даних з проектів серверних застосунків, що розробляються з використанням фреймворку NestJS, 10. Перевірка даних на нормальність, 11. Вилучення викидів, 12. Побудова нелінійної регресійної моделі, 13. Побудова лінійної регресійної моделі, 14. Визначення довірчих інтервалів та інтервалів передбачення регресійних моделей, 15. Порівняння якості отриманих двохфакторних регресійних моделей, 16. Порівняння нелінійної регресійної моделі з існуючими моделями, 17. Діаграма варіантів використання, 18. Діаграма діяльності, 19.

Діаграма класів, 20. Діаграма послідовності, 21. Демонстрація роботи застосунку, 22-24. Висновки.

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 _____ Бринзей. О. А.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л. О.
(підпис) (ПІБ)

РЕФЕРАТ

Бринзей Олександр Андрійович

«Нелінійна регресійна модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, 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: 89 стор., 16 табл., 7 рис., 23 використаних джерел, 5 додатків.

Актуальність теми: обумовлена потребою в більш точному оцінюванні розміру серверних застосунків, які створюються з використання фреймворку NestJS через відсутність побудованих моделей для цього фреймворку.

Мета дослідження. Метою роботи підвищення достовірності оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, за допомогою нелінійної регресійної моделі.

Об’єкт дослідження: процес оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Предмет дослідження: нелінійні регресійні моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Методи дослідження: методи теорії ймовірностей та математичної статистики, лінійного та нелінійного регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, з використанням нормалізуючого перетворення на основі десяткового логарифму, що дозволяє підвищити достовірність оцінювання розміру програмного забезпечення в порівнянні з двохфакторною лінійною регресійною моделлю та однофакторною нелінійною регресійною моделлю для оцінювання розміру веб-додатків, реалізованих мовою Java. Ця двофакторна нелінійна регресійна модель у порівнянні з побудованою

однофакторною нелінійною регресійною моделлю та існуючою однофакторною нелінійною регресійною моделлю для застосунків, розроблених мовою Java, має кращі показники якості моделей R^2, $MMRE$ та $PRED(0{,}25)$.

Практичне значення одержаних результатів. Розроблено програму для оцінювання розміру серверних, що створюються з використанням фреймворку NestJS. Програму створено на мові програмування Python. Програму, що розроблено, можна використовувати для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, на ранніх стадіях їх проектування.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи опубліковані на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 року).

Публікації: Основні положення й результати кваліфікаційної роботи опубліковано у 1 науковій праці – тезах конференції.

Ключові слова: оцінювання кількості строк коду, серверні застосунки; нелінійна регресійна модель, фреймворк NestJS.

ABSTRACT

Brynzei Oleksandr Anriyovych

«A nonlinear regression model for estimating the size of server applications created using NestJS framework and the developing the software for its implementation»

Qualification work for obtaining a master's degree in specialty 121 - "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2022.

Volume: 89 p., 16 tables, 7 figures, 23 references, 5 appendices.

Topic Relevance: conditioned by the need for a more accurate assessment of the size of server applications that are developed using the NestJS framework due to the lack of built models for this framework.

Research goal. The purpose of the work is to increase the confidence of estimation the size of server applications created using the NestJS framework, using a nonlinear regression model.

Object of research: the process of estimating the size of server applications developed using the NestJS framework.

Subject of research: nonlinear regression models for estimating the size of server applications developed using the NestJS framework.

Methods of research: methods of probability theory and mathematical statistics, linear and non-linear regression analysis.

Scientific contribution: improved the nonlinear regression model for estimating the size of server applications developed using the NestJS framework, using a normalization transformation based on the decimal logarithm, which allows to increase the confidence of estimating the size of the software compared to the two-factor linear regression model and the one-variable nonlinear regression model for estimating the size of web applications developed in Java. This two-factor nonlinear regression model has better R^2 , $MMRE$, and $PRED(0.25)$ model quality scores compared to the constructed one-factor nonlinear regression model and the existing one-factor nonlinear regression model for applications developed in Java.

Practical value of obtained results. The program for estimating the size of servers created using the NestJS framework has been developed. The program was created in the Python programming language. The developed program can be used to estimate the size of server applications created using the NestJS framework in the early stages of their design.

Approbation of the thesis results. The 3nd All-Ukrainian scientific-practical Internet conference "Information technologies: models, algorithms, systems (ITMAS 2022)" (Mykolaiv, October 26-28, 2022).

Publications. The main provisions and results of the qualification work are published in 1 scientific work - theses of the conference.

Keywords: LOC estimation, server applications, non-linear regression model, NestJS framework.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – Персональний комп’ютер

COCOMO – COnstructive COst Model

KLOC – Thousand lines of code

LB – Lower Bound

UP – Upper Bound

LOC – Lines Of Code

MMRE – Mean Magnitude of Relative Error

MRE – Magnitude of Relative Error

PRED – Percentage of prediction

SMD – the Square Mahalanobis Distance

VIF – Variance Influence Factor

ВСТУП

Актуальність теми. Перед початком розробки та на її ранніх стадіях незалежно від типу програмного забезпечення стоїть задача оцінювання трудомісткості розробки програмного забезпечення. Для прогнозування трудомісткості розробки програмного забезпечення може використовуватись така математична модель, як COCOMO II [1, 2]. Одним із головних параметрів для цієї моделі є розмір програмного забезпечення. Тому для визначення трудомісткості розробки програмного забезпечення треба спочатку вирішити задачу прогнозування розміру вихідного коду програмного забезпечення.

В останні роки є великий попит на різного роду сервіси та сервери. В цій області досить добре себе зарекомендувало JavaScript оточення, побудоване на двигуні Chrome V8 – NodeJS. Тому все частіше розробники обирають його для написання серверної частини застосунків будь-якого розміру.

При написані таких застосунків, головним критерієм є їхнє швидке створення та запуск. Для пришвидшення розробки та підтримки чіткої структури та визначеної архітектури застосунку використовуються фреймворки.

Серед серверних фреймворків найпопулярнішими та найкращими є ExpressJS, KoaJS, MeteorJS, NestJS, SailsJS. Перші два більше спрямовані на полегшення створення контролерів, роутерів, тощо. А ось останні три надають розробнику готову структуру застосунку, що пришвидшує розробку.

Для дослідження серед вище вказаних фреймворків було обрано NestJS. Він має наступні переваги перед іншими:

– вбудований TypeScript;

– CLI – для автоматичної генерації модулів, контролерів і таке інше;

– підлягає тестуванню;

– контейнер для введення залежностей;

– прості у використанні зовнішні бібліотеки завдяки впорядкованому розташуванню коду.

В ході аналізу останніх публікацій були знайдені трьохфакторні моделі множинної лінійної регресії [3, 4] на основі концептуальної моделі даних для прогнозування кількості рядків коду програмного забезпечення, що розробляється з використання мови PHP. Також були знайдені трьохфакторні нелінійні регресійні моделі [5-7] для оцінювання розміру web-застосунків, що створюються за допомогою PHP-фреймворків, а також трьохфакторні нелінійні регресійні моделі [8, 9], які використовуються для оцінювання розміру PHP-застосунків з відкритим вихідним кодом.

Точність оцінки розміру програмного забезпечення залежить, головним чином, від мови програмування, фреймворку та інших факторів. Вказані моделі розроблені для мови програмування PHP та PHP-фреймворків.

Відсутність рівнянь або моделей, які дозволяють прогнозувати розмір вихідного коду серверних застосунків, що створюються з використанням фреймворку NestJS, робить актуальним розробку нелінійної регресійної моделі для оцінювання розміру вихідного коду серверних застосунків, що створюються з використанням фреймворку NestJS.

У якості нормалізуючого перетворення використовують десятковий логарифм, нелінійне перетворення Бокса-Кокса (Box-Cox), а також нормалізуюче перетворення Джонсона. Такі моделі як COCOMO та COCOMO II були створені із використанням десяткового логарифму в якості нормалізуючого перетворення. Тому для побудови нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, буде використовуватися десятковий логарифм в якості нормалізуючого перетворення.

Мета та завдання дослідження. Метою роботи є підвищення достовірності оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, за допомогою нелінійної регресійної моделі.

Для досягнення поставленої мети потрібно вирішити такі завдання:

– Провести пошук та виконати аналіз регресійних моделей для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

– Знайти якомога більше проектів з відкритим вихідним кодом та провести збір метрик, що можуть бути використані для побудови нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

– Перевірити зібрані дані на наявність мультиколінеарності і багатовимірних викидів.

– Побудувати нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, на основі нормалізуючого перетворення десяткового логарифму.

– На підставі розробленої нелінійної регресійної моделі розробити програму для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Об’єкт дослідження: процес оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Предмет дослідження: нелінійні регресійні моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Методи дослідження: методи теорії ймовірностей та математичної статистики, лінійного та нелінійного регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, з використанням нормалізуючого перетворення на основі десяткового логарифму, що дозволяє підвищити достовірність оцінювання розміру програмного забезпечення в порівнянні з двохфакторною лінійною регресійною моделлю та однофакторною нелінійною регресійною моделлю для оцінювання розміру веб-додатків, реалізованих мовою Java. Ця двофакторна нелінійна регресійна модель у порівнянні з побудованою однофакторною нелінійною регресійною моделлю та існуючою однофакторною нелінійною регресійною моделлю для застосунків, розроблених мовою Java, має кращі показники якості моделей R^2, $MMRE$ та $PRED(0{,}25)$.

Практичне значення одержаних результатів. Розроблено програму для оцінювання розміру серверних, що створюються з використанням фреймворку NestJS. Програму створено на мові програмування Python. Програму, що розроблено, можна використовувати для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, на ранніх стадіях їх проектування.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною науковою працею. Усі наукові результати отримано автором особисто. У праці [10], яка була опублікована у співавторстві, автору належить нелінійна регресійна модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, на основі нормалізуючого перетворення десяткового логарифму.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи опубліковані на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 року).

Публікації: Основні положення й результати кваліфікаційної роботи опубліковано у 1 науковій праці – тезах конференції.

Ключові слова: оцінювання кількості строк коду, серверні застосунки; нелінійна регресійна модель, фреймворк NestJS.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МАТЕМАТИЧНИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ СЕРВЕРНИХ ЗАСТОСУНКІВ ТА ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ ПРОВЕДЕННЯ ДОСЛІДЖЕНЬ ЗА ОБРАНОЮ ТЕМОЮ

1.1 Аналіз існуючих методів та математичних моделей для оцінювання розміру серверних застосунків

Для оцінювання трудомісткості розробки програмного забезпечення будь-якого типу використовують нелінійні регресійні рівняння та нелінійні регресійні моделі. З найвідоміших та таких, що найчастіше використовуються, можна виділити COCOMO II [1, 2]. До основних факторів в таких моделях та методах частіше всього відноситься розмір програмного забезпечення. Як правило розмір програмного забезпечення перед початком його розробки та на ранніх стадіях розробки невідомий. Тому не менш складним та важливим завданням виступає прогнозування розміру програмного забезпечення. Для вирішення цього завдання будуються лінійні регресійні моделі та рівняння, а також нелінійні регресійні моделі та рівняння.

Під час ознайомлення з останніми публікаціями по даній тематиці, було виявлено трьохфакторні моделі множинної лінійної регресії [3, 4], які призначені для оцінювання кількості рядків коду програмного забезпечення з відкритим вихідним кодом, розробленим на мові PHP та Java. В цих роботах в якості предикторів виступають: середня кількість атрибутів на клас, загальна кількість зв’язків та загальна кількість класів у концептуальній моделі даних.

Окрім цього були знайдені трьохфакторні нелінійні регресійні моделі [5-7], які призначені для оцінювання розміру web-застосунків, що створюються за допомогою PHP-фреймворків. В даних роботах в якості метрик було обрано розмір web-застосунків у тисячах рядків коду, кількість класів, середню кількість методів на клас, глибину дерева наслідування та суму середньої кількості класів, на які

впливає даний клас і середньої кількості класів, з яких даний клас отримує ефекти. Для удосконалення нелінійної регресійної моделі в залежності від трьох факторів було використано чотиривимірне нормалізуюче перетворення Джонсона сім'ї S_B. Дане перетворення дозволяє збільшити достовірність оцінювання та має кращі результати на відміну від лінійних регресійних моделей та нелінійних регресійних моделей, які будуються з використанням одновимірних нормалізуючих перетворень.

Також, окрім вище вказаних моделей, було знайдено трьохфакторні нелінійні регресійні моделі [8, 9], які використовуються для оцінювання розміру PHP-застосунків з відкритим вихідним кодом. В даних моделях, як і в попередніх, в ролі предикторів виступає загальна кількості класів, сума середньої кількості класів, на які впливає даний клас, і середня кількість класів, з яких даний клас отримує ефекти, а також середньої кількості методів на клас. В якості нормалізуючого перетворення в даних моделях було використано, як і в попередніх роботах, чотиривимірне нормалізуюче перетворення Джонсона сім'ї $S_{B.}$ Використання даного нормалізуючого перетворення дозволяє підвищити достовірність оцінювання залежної змінної нелінійної регресії в порівнянні з використанням одновимірних нормалізуючих перетворень.

Окрім трьохфакторних моделей було знайдено однофакторні нелінійні регресійні моделі [11], які призначені для оцінювання розміру веб-додатків, що розробляються з використанням мови Java. В даній роботі представлені моделі, які побудовані на основі двох метрик: кількості рядків коду та загальної кількості класів. В даній роботі вказані нелінійні моделі були побудовані на основі десяткового логарифму та одновимірного нормалізуючого перетворення Джонсона.

Для побудови лінійних регресійних моделей існують певні обмеження. До таких обмежень відноситься обмеження на розподіл залежної змінної та залишків: вони повинні підпорядковуватись нормальному закону розподілу для можливості проведення наступних операцій. Дане обмеження напряму залежить від якості початкових даних. На практиці початкові дані досить рідко підпорядковуються

нормальному закону розподілу. У зв’язку з цим виникає необхідність розробки нелінійних регресійних моделей з використанням відповідних методів множинного нелінійного регресійного аналізу для підвищення якості прогнозування розміру програмного забезпечення.

До таких методів відноситься нормалізація. Для нормалізації випадкових величин, що не підпорядковуються нормальному закону розподілу, можуть бути використані нормалізуючі перетворення. Перетворення Бокса-Кокса та Джонсона мають додаткові параметри, тому краще нормалізують дані, але додатково потребують знаходження цих самих параметрів, що супроводжується додатковими обчисленнями. На відміну від зазначених вище перетворень, нормалізуюче перетворення в формі десяткового логарифму не вимагає додаткових обчислень, хоча і показує гірші результати в порівнянні з ними. Такі моделі як ISBSG, COCOMO та COCOMO II використовують саме десятковой логарифм в якості нормалізуючого перетворення. Тому для побудови нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, в якості нормалізуючого перетворення було обрано десятковий логарифм.

У випадку нормальності початкових даних в якості регресійної моделі можна взяти лінійну регресійну модель з наступним загальним виглядом:

$$Y = \hat{Y} + \varepsilon = \bar{Y} + (X^{+})\hat{b} + \varepsilon, \tag{1.1}$$

де $\hat{Y}$ – результат передбачення рівняння лінійної регресії для значень компонент вектору $X = \{X_1, X_2, \dots, X_k\}$, X^{+} – матриця центрованих змінних, яка містить наступні значення: $X_{1_i} - \bar{X}_1, X_{2_i} - X_2, \dots, X_{k_i} - \bar{X}_k$; $\hat{b}$ – оцінка вектору параметрів рівняння лінійної регресії, $b = \{b_1, b_2, \dots, b_k\}^T$, ε – гаусівська випадкова величина, k – кількість факторів у моделі.

За умови нормальності розподілу випадкових величин, можна побудувати довірчий інтервал лінійної регресії, який будується з використанням t-розподілу Стьюдента

$$\hat{Y}_i \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{ \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+) \right\}^{1\backslash 2}, \tag{1.2}$$

а інтервал передбачення лінійної регресії як

$$\hat{Y}_i \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{ 1 + \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+) \right\}^{1\backslash 2}, \tag{1.3}$$

де $S_Y^2 = \frac{1}{\mathrm{v}} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2$; $\mathrm{v} = N - k - 1$; $(\mathbf{X}^+)^T \mathbf{X}^+ - k \times k$ матриця

$$(\mathbf{X}^+)^T \mathbf{X}^+ = \begin{pmatrix} S_{X_1 X_1} & S_{X_1 X_2} & \dots & S_{X_1 X_k} \\ S_{X_1 X_2} & S_{X_2 X_2} & \dots & S_{X_2 X_k} \\ \dots & \dots & \dots & \dots \\ S_{X_1 X_k} & S_{X_2 X_k} & \dots & S_{X_k X_k} \end{pmatrix}, \tag{1.4}$$

де $S_{X_q X_r} = \sum_{i=1}^{N} \left[X_{q_i} - \bar{X}_q \right] \left[X_{r_i} - \bar{X}_r \right]$, $q, r = \overline{1, k}$.

Використання нормалізуючих перетворень є розповсюдженою практикою при побудові регресійних моделей для негаусівських даних. У [12] запропоновано метод побудови нелінійних регресійних моделей, згідно якого спочатку виконують нормалізацію даних та вилучень викидів, будують лінійну регресійну модель для нормалізованих даних без викидів та в якості останнього кроку з використанням зворотнього нормалізуючого перетворення отримують нелінійну регресійну модель.

Нормалізуюче перетворення негаусівського випадкового вектору $\mathrm{P} = \{Y, X_1, X_2, \dots, X_k\}^T$ у гаусівський випадковий вектор $\mathrm{T} = \{Z_Y, Z_1, Z_2, \dots, Z_k\}^T$, задається як

$$\mathrm{T} = \psi(\mathrm{P}), \tag{1.5}$$

а зворотне перетворення до (1.5) як

$$\mathrm{P} = \psi^{-1}(\mathrm{T}), \tag{1.6}$$

тут ψ – вектор, $\psi = \{\psi_Y, \psi_1, \psi_2, \dots, \psi_k\}^T$.

Таким чином нелінійна регресійна модель, що побудована з використанням нормалізуючого перетворення [13], має вигляд

$$Y = \psi_Y^{-1}[\hat{Z}_Y + \varepsilon] = \psi_Y^{-1}[\bar{Z}_Y + (\boldsymbol{Z}_X^+)\hat{b} + \varepsilon], \tag{1.7}$$

де ψ_Y – перша компонента вектору ψ перетворення (1.5).

В свою чергу довірчий інтервал нелінійної регресії для *k* незалежних змінних можна знайти таким чином

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{\frac{1}{N} + (\boldsymbol{z}_X^+)^T[(\boldsymbol{Z}_X^+)^T\boldsymbol{Z}_X^+]^{-1}(\boldsymbol{z}_X^+)\right\}^{1\backslash 2}\right), \tag{1.8}$$

а інтервал передбачення нелінійної регресії як

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{1 + \frac{1}{N} + (\boldsymbol{z}_X^+)^T[(\boldsymbol{Z}_X^+)^T\boldsymbol{Z}_X^+]^{-1}(\boldsymbol{z}_X^+)\right\}^{1\backslash 2}\right). \tag{1.9}$$

1.2 Обґрунтування необхідності проведення досліджень

З появою оточення NodeJS його популярність росте з кожним роком. Його починають використовувати або переходять на його використання як невеличкі компанії так і гіганти ринку, до яких можна віднести наступні: Netflix, NASA, eBay, LinkedIn, Uber. [14].

Але розробка з використанням одного лиш JS відбувається повільно. Тому для спрощення роботи програмістів були розроблені фреймворки, які дозволяли в

одноманітній манері швидко та зручно розробляти серверні застосунки. Такими фреймворками були і є ExpressJS, KoaJS та інші. Але вони надавали лише API для полегшення написання та роботи з контролерами, роутерами, обробниками запитів. Тому для швидкого розгортання та початку роботи були розроблені фреймворки, які надавали готову структуру проекту та мали можливість легко розширюватись. Одним із таких фреймворків є NestJS.

NestJS з'явився в 2017 році, а вже в 2018 мав 300% збільшення кількості зірок на GitHub. У 2019 році він ще не з'явився в попередньо визначеному списку бекенд-фреймворків на stateofjs [15], але в подальшому потрапив до нього та обійшов конкурентів. Тому в статистиці за 2020 рік ми вже маємо більше даних, так наприклад 87% користувачів задоволені NestJS.

Процес зростання популярності NestJS можна переглянути в матеріалах, які підготували statisticsanddata.org [16]. Крім того NestJS вже має велике комьюніті програмістів і компаній, які ним користуються, таких як: Adidas, Decathlon та багато інших. За списком компаній, які використовують NestJS у розробці, можна стежити на офіційному веб-сайті фреймворку.

Тож можна сказати, що цей фреймворк є вибором не лише програмістів, а й великих бізнес компаній, які за допомогою цього фреймворку економлять час та фінанси.

В ході аналізу останніх публікацій не було знайдено моделей для прогнозування розміру серверних застосунків, що розробляються з використанням NestJS. Знайдені моделі були розроблені для оцінювання розміру програмного забезпечення, що розробляється з використанням мов програмування Java, а також мови PHP та її фреймворків. Таким чином відсутність моделей, що призначені для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, робить актуальною розробку нелінійної регресійної моделі для оцінювання розміру вихідного коду серверних застосунків, що створюються з використанням фреймворку NestJS.

Для побудови нелінійної регресійної моделі було вирішено використовувати нормалізуюче перетворення у вигляді десяткового логарифму.

1.3 Висновки до розділу 1

В цьому розділі наведено огляд існуючих рівнянь, моделей та методів, що використовуються для оцінювання розміру програмного забезпечення. Виявлено відсутність моделей для прогнозування розміру серверних застосунків, що розробляються з використанням фреймворку NestJS. Здійснено обґрунтування актуальності та необхідності проведення досліджень за обраною темою. В розділі також визначено, що:

– На момент написання роботи для оцінювання розміру серверних застосунків використовуються методи лінійного та нелінійного регресійного аналізу.

– Регресійні моделі, частіше всього, є нелінійними та мають кращі показники якості в порівнянні з лінійними. Це призводить до необхідності розробки саме нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що розробляються з використанням фреймворку NestJS.

– Нормалізуючі перетворення часто дають приріст в показниках якості моделей. Причому зазвичай у якості нормалізуючого перетворення використовується десятковий логарифм.

– В знайдених нелінійних регресійних моделях для оцінювання розміру програмного забезпечення частіше всього в якості метрик використовують метрики концептуальної моделі даних у вигляді діаграми класів, такі як: загальна кількість класів, загальна кількість зв'язків та середня кількість атрибутів на клас.

– На момент написання роботи регресійних моделей для оцінювання розміру серверних застосунків, що розробляються з використанням фреймворку NestJS, не було знайдено, що робить актуальним дане дослідження.

2 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ СЕРВЕРНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NESTJS

2.1 Перевірка даних для побудови нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, на наявність мультиколінеарності та викидів

Щоб побудувати нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, було знайдено 62 серверних застосунки з відкритим вихідним кодом, які розміщені на веб-сервісі GitHub (https://github.com/). За допомогою платформи з відкритим вихідним кодом SonarQube (https://www.sonarqube.org/), були зібрані наступні метрики: загальна кількість рядків в тисячах (lines), кількість рядків коду в тисячах (KLOC), кількість файлів (files), кількість класів (classes) та кількість функцій (functions). Для побудови регресійної моделі були обрані наступні метрики: кількість рядків коду в тисячах (KLOC), кількість класів (classes) та кількість функцій (functions). Отримані метрики наведено у таблиці 2.1.

Таблиця 2.1 – Початковий набір даних разом з розрахованою відстанню Махаланобіса (SMD)

№	KLOC	Classes	Functions	SMD	№	KLOC	Classes	Functions	SMD
1	2	3	4	5	6	7	8	9	10
1	1.118	10	95	5.535	32	2.754	73	213	1.464
2	2.813	59	222	0.860	33	1.028	30	80	0.590
3	0.342	13	39	2.180	34	1.473	25	173	1.801
4	5.816	263	800	**9.718**	35	0.945	37	62	3.125
5	0.974	18	92	0.619	36	0.522	17	41	1.327
6	0.284	14	31	3.229	37	2.397	53	222	0.551
7	6.512	140	436	4.113	38	2.193	32	117	3.430
8	6.178	196	590	5.164	39	0.779	18	64	0.392
9	0.194	7	29	5.707	40	2.416	46	212	0.509

Продовження табл. 2.1.

1	2	3	4	5	6	7	8	9	10
10	0.671	16	62	0.458	41	1.511	38	147	0.209
11	1.052	36	107	0.734	42	3.804	76	284	1.618
12	0.404	18	36	2.311	43	0.092	2	13	**11.554**
13	1.043	29	96	0.123	44	5.475	87	719	5.102
14	0.611	19	46	1.232	45	0.494	5	36	7.045
15	0.861	25	66	0.691	46	2.033	12	84	13.207
16	1.186	29	120	0.176	47	1.213	15	103	2.340
17	2.247	34	236	1.606	48	2.238	46	248	1.128
18	0.993	23	91	0.072	49	2.321	56	190	0.666
19	0.946	27	91	0.190	50	0.165	7	21	4.944
20	0.993	42	101	1.690	51	0.767	31	60	1.919
21	0.581	37	50	4.924	52	12.251	225	2617	**17.170**
22	2.397	49	232	0.625	53	0.613	11	65	1.982
23	0.647	28	69	1.712	54	1.154	35	98	0.432
24	1.322	50	116	1.454	55	3.021	45	171	2.982
25	8.099	74	719	5.603	56	1.342	25	96	0.706
26	0.315	14	38	2.947	57	0.752	12	60	1.670
27	0.371	15	31	2.271	58	4.554	80	163	**10.890**
28	0.342	9	39	2.362	59	3.191	33	387	5.481
29	0.604	20	65	0.892	60	1.002	21	67	1.156
30	0.931	27	91	0.237	61	1.273	32	57	6.226
31	2.441	26	227	3.126	62	3.989	61	370	1.792

Окрім значень обраних метрик в таблиці 2.1 наведені значення квадрату відстані Махаланобіса (SMD). Для кожної багатовимірної точки даних, значення SMD можна знайти за формулою:

$$d_i^2 = (X_i - \overline{X})^T S_N^{-1} (X_i - \overline{X}) \,, \tag{2.1}$$

де X_i – i-та точка багатовимірних даних негаусівського вектору $X; \overline{X}$ – вектор вибіркових середніх; S_N – вибіркова коваріаційна матриця. В нашому випадку розглядається 3 компонентний вектор, N – кількість елементів у вибірці.

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (X_i - \overline{X}) \, (X_i - \overline{X})^T. \tag{2.2}$$

Для даних в таблиці 2.1 вектор X є негаусівським випадковим вектором. На це вказує наявність рядків в таблиці 2.1 зі значенням SMD більше за величину квантіля розподілу χ^2 для рівня значимості 0,05. Дані проектів 4, 43, 52, 58 слід вважати багатовимірними викидами, адже для них значення SMD більше критичного значення квантіля розподілу χ^2 для рівня значимості 0,05, яке складає 7,814.

При побудові рівняння множинної регресії може виникнути проблема мультиколінеарності факторів. Тому перед початком побудови нелінійної регресійної моделі для прогнозування розміру серверних застосунків, що створюються з використанням фреймворку NestJS, необхідно провести аналіз предикторів на наявність мультиколінеарності.

Мультиколінеарність – це негативне явище множинного регресійного аналізу, яке не дозволяє здійснити оцінювання окремого впливу кожного фактору на залежну змінну. Наявність мультиколінеарності свідчить про те, що в множинній регресійній моделі два або більше факторів (незалежних змінні) пов‘язані між собою або мають високий ступінь кореляції.

Наявність мультиколінеарності будемо визначати за коефіцієнтами впливу дисперсії (Variance Inflation Factors, VIFs) серед майбутніх предикторів (факторів) в моделі множинної лінійної регресії. Для лінійної моделі множинної регресії з k-предикторами X_i $\left(i = \overline{1,k}\right)$, VIFs – це діагональні елементи оберненої коваріаційної матриці $k \times k$ k-предикторів [17].

Для обраних предикторів Classes та Functions, наведених у таблиці 2.1, отримано значення VIFs 2,456 для обох метрик. Обидва значення менші за 5, що показує відсутність мультиколінеарності в даних.

Перевірка вихідних даних продемонструвала, що вони є негаусівськими. Це зумовлює необхідність їх нормалізації для пошуку та вилучення викидів. Пошук та вилучення викидів проводитимемо з використанням нормалізуючих перетворень, зокрема нормалізуючого перетворення десяткового логарифму, та розрахунку квадрату відстані Махаланобіса SMD як це робиться у [18, 19]. Для окремо взятої точки нормалізованих даних i $(i = \overline{1,N})$ застосовується наступна формула

$$d_i^2 = (Z_i - \overline{Z})^T S_N^{-1} (Z_i - \overline{Z}) , \quad (2.4)$$

де Z_i – i-та точка багатовимірних даних нормалізованого вектору Z; $\overline{Z}$ – вектор вибіркових середніх; S_N – вибіркова коваріаційна матриця для нормалізованих даних. В роботі розглядаємо 3 компонентний вектор.

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \overline{Z}) (Z_i - \overline{Z})^T. \quad (2.5)$$

Щоб отримати вектор багатовимірних нормализованих даних слід використати (1.5) та нормалізуюче перетворенням десяткового логарифму. В цьому випадку за (1.5) отримаємо перетворення, що має вигляд $\psi = \{\lg Y, \lg X_1, \lg X_2\}^T$.

Одним з відомим підходів до перевірки даних на викиди є порівняння значення статистики T_{s_i} з квантілем F-розподілу з рівнем значимості α ($F_{m,N-m,\alpha}$)

$$T_{S_i} = \frac{N(N-m)d_i^2}{m(N^2-1)} . \quad (2.6)$$

Якщо для якогось рядка значення статистики T_{S_i} більше критичного значення Фішера $F_{m,N-m,\alpha}$, або значення SMD більше значення квантілю розподілу χ^2, то такий рядок слід розглядати як викид [20]. Ітераційний процес вилучення викидів проводиться доти, доки усі вики не будуть вилучені.

Ітераційний процес пошуку викидів зайняв 6 ітерацій, усього було вилучено 17 проектів програмного забезпечення. Під час розрахунку значень квадрату відстані Махаланобіса SMD та значення статистики T_{S_i} було визначено, що обидва критерія однаково виявляють набори даних, які слід вважати викидами. Важливо мати на увазі, що така поведінка спостерігається нечасто. Вилучення викидів з вказанням проектів та розрахованих значень SMD на кожному кроці ітераційного процесу наведено у таблиці 2.2.

Таблиця 2.2 – Процес вилучення викидів

№	KLOC (Y)	Classes (X1)	Functions (X2)	SMD
Викиди після 1-ї ітерації				
4	5.82	263.00	800.00	9.718006856
43	0.09	2.00	13.00	11.55471356
46	2.03	12.00	84.00	13.20772904
52	12.25	225.00	2,617.00	17.17067855
58	4.55	80.00	163.00	10.89008614
Викиди після 2-ї ітерації				
8	6.18	196.00	590.00	8.682753076
9	0.19	7.00	29.00	7.827439984
44	5.48	87.00	719.00	8.554798353
45	0.49	5.00	36.00	11.51201271
61	1.27	32.00	57.00	10.608049
Викиди після 3-ї ітерації				
1	1.12	10.00	95.00	9.623294614
38	2.19	32.00	117.00	8.901678909
59	3.19	33.00	387.00	8.150484149
Викиди після 4-ї ітерації				
25	8.10	74.00	719.00	8.047212219
55	3.02	45.00	171.00	8.193187179
Викиди після 5-ї ітерації				
7	6.51	140.00	436.00	8.08982989
Викиди після 6-ї ітерації				
50	0.17	7.00	21.00	8.273983364

Для подальшої побудови нелінійної регресійної моделі оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, буде використано нормалізований набір даних з 45 проектів без викидів.

Як було зазначено вище, для того, аби мати теоретичне підґрунтя для застосування лінійних регресійних моделей залишки регресії або залежна змінна мають бути нормально розподіленими. Для великих вибірок (більше 30 елементів) слушно застосувати критерій Пірсона [21].

Значення критерія Пірсона χ^2 визначають як

$$\chi^2 = \sum_{j=1}^{m} \frac{\left(n_j - np_j\right)^2}{np_j}, \tag{2.7}$$

де m – кількість підінтервалів (часток, кліток), на які розбивається інтервал $[x_{min}, x_{max}]$; x_{min}, x_{max}– відповідно мінімальне і максимальне значення випадкової величини X; n_j – абсолютна частота в j-му підінтервалі (кількість значень випадкової величини, які попадають у j-ий підінтервал); p_j – ймовірність того, що значення випадкової величини X попадають у j-ий підінтервал.

Значення кількості підінтервалів m можна розрахувати за формулою Старджеса або формулою Брукса-Карузена. В даній роботі використовується формула Старджеса - $m = \log_2 n + 1 = 3{,}3 \lg n + 1$. Для наступного розрахунку критерія Пірсона візьмемо $m = 6$.

Розрахунок ймовірностей потрапляння значень випадкової величини у j-ий підінтервал можна провести за допомогою наступної формули:

$$\boldsymbol{p_j = \int_{x_{j-1}}^{x_j} f(x)dx}, \tag{2.8}$$

де x_{j-1} та x_j – ліва і права границі j-го підінтервалу. $f(x)$ – функція щільності ймовірності.

Функція щільності ймовірності нормального розподілу має вигляд

$$\boldsymbol{f(x) = \frac{1}{\sigma_x \sqrt{2\pi}} * e^{-\frac{(x-m_x)^2}{2\sigma_x^2}}}, \tag{2.9}$$

де m_x – математичне сподівання випадкової величини, σ_x – середьоквадратичне відхилення випадкової величини.

Після цього в залежності від рівня значимості α та кількості ступенів вільності ν за таблицею верхніх 100α %-вих точок розподілу χ^2 знаходять значення $\chi^2_{\text{кр}}$. Якщо $\chi^2 \leq \chi^2_{\text{кр}}$, то з ймовірністю $1-\alpha$ можна прийняти гіпотезу про те, що закон розподілу результатів спостережень є нормальним, якщо $\chi^2 > \chi^2_{\text{кр}}$ – цю гіпотезу потрібно відкинути.

Значення ν – кількість ступенів вільності можна визначити як $\nu = m - k - 1$, де k – кількість параметрів, від яких залежить закон розподілу. Для нормального розподілу $k = 2$.

2.2 Побудова нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Для побудови нелінійної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, використаємо дані без викидів. Для отриманих нормалізованих даних без викидів побудуємо лінійну двохфакторну регресійну модель, яка має наступний вигляд

$$Z_Y = \hat{Z}_Y + \varepsilon = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \varepsilon, \tag{2.10}$$

де $Z_y = \lg Y; Z_i = \lg X_i, i = \overline{1,2}$.

Для визначення параметрів моделі (2.10) використовується метод найменших квадратів. Після використання методу найменших квадратів були отримані наступні значення параметрів: $\hat{b}_0 = -1.997$, $\hat{b}_1 = 0.187$, $\hat{b}_2 = 0.884$.

Розрахуємо значення критерія Пірсона для залишків ε лінійної регресії для нормалізованих даних. Це необхідно для того, аби отримати теоретичне підґрунтя для застосування лінійних регресійних моделей.

Використовуючи формулу (2.7) маємо значення критерія Пірсона χ^2 для лінійної регресії для нормалізованих даних 2,873. Критичне значення критерія Пірсона χ^2 двох ступенів вільності та на рівні значимості 0,05 дорівнює 7,814.

Бачимо, що $\chi^2 < \chi^2_{\text{кр}}$, тому нульова гіпотеза щодо нормальності розподілу величини ε приймається на рівні значимості 0,05.

Отримана оцінка математичного сподівання залишків ε лінійної регресії для нормалізованих даних – 5,03e^{-15}, оцінка середньоквадратичного відхилення – 0,060.

За лінійним регресійним рівнянням (2.10) та взаємо-зворотнім нормалізуючим перетворенням отримуємо нелінійну регресійну модель для оцінювання кількості рядків коду серверних застосунків, що створюються з використанням фреймворку NestJS:

$$Y = 10^{\varepsilon + \hat{b}_0} X_1^{\hat{b}_1} X_2^{\hat{b}_2}. \quad (2.11)$$

Отриману модель необхідно перевірити на якість. Для визначення якості моделі можуть бути використані такі показники як множинний коефіцієнт детермінації R^2, середня величина відносної помилки MMRE та відсоток прогнозованих результатів, для яких величина відносної помилки MMRE менша за 0,25 PRED (0,25). Допустимі значення для MMRE мають складати не більше 0,25, а для PRED (0,25) не менше 0,75. Значення R^2 має бути якомога ближчим до одиниці [22, 23].

Після розрахунку значень вказаних показників якості, отримали наступні результати: $R^2 = 0{,}966$, $MMRE = 0{,}114$, $PRED(0{,}25) = 0{,}955$.

Для порівняння було побудовано двохфакторну лінійну регресійну модель у припущені про нормальність розподілу.

$$Y = \hat{Y} + \varepsilon = \hat{b}_0 + \hat{b}_1 X_1 + \hat{b}_2 X_2 + \hat{b}_3 X_3 + \varepsilon. \quad (2.12)$$

Для визначення параметрів моделі (2.12), як і для моделі (2.10) було використано метод найменших квадратів та отримані наступні значення параметрів: $\hat{b}_0 = -0.126$, $\hat{b}_1 = 0.011$, $\hat{b}_2 = 0.009$.

Для отриманої лінійної моделі було розраховано показники якості, які мають наступні значення: $R^2 = 0{,}966$, $MMRE = 0{,}115$, $PRED(0{,}25) = 0{,}888$.

Розрахуємо значення критерія Пірсона для залишків ε лінійної регресії для ненормалізованих даних. Це необхідно для того, аби отримати теоретичне підґрунтя для застосування лінійних регресійних моделей.

Використовуючи формулу (2.7) маємо значення критерія Пірсона χ^2 для лінійної регресії для ненормалізованих даних 0,450. Критичне значення критерія Пірсона χ^2 двох ступенів вільності та на рівні значимості 0,05 дорівнює 7,814. Бачимо, що $\chi^2 < \chi^2_{\text{кр}}$, тому нульова гіпотеза щодо нормальності розподілу величини ε приймається на рівні значимості 0,05.

Отримана оцінка математичного сподівання залишків ε лінійної регресії для ненормалізованих даних – $3{,}91e^{-15}$, оцінка середньоквадратичного відхилення – 0,169.

Визначимо довірчі інтервали лінійної та нелінійної регресійних моделей. Для визначення довірчих інтервалів скористаємось формулами (1.2) та (1.8) відповідно. У таблиці 2.3 показано довірчі інтервали лінійної та нелінійної двохфакторних регресійних моделей та їх ширини.

Таблиця 2.3 – Порівняння ширин довірчих інтервалів лінійної та нелінійної регресії

№	KLOC (Y)	Межі довірчих інтервалів				Ширина інтервалу	
		Лінійна регресія		Нелінійна регресія		Лінійна регресія	Нелінійна регресія
		LB	UB	LB	UB		
1	2	3	4	5	6	7	8
1	2 813	0,6145	1,3874	2,374	2,768	0,2044	0,3944
2	342	0,2593	0,7622	2,377	2,765	0,1560	**0,0627**
3	974	0,2593	0,7622	2,394	2,745	0,1444	**0,1289**
4	284	0,5314	1,0532	2,357	2,789	0,1562	**0,0578**
5	670	-0,2597	0,4078	2,410	2,728	0,1418	**0,0806**
6	1 052	-0,3141	0,3469	2,437	2,698	0,1241	0,1248
7	404	1,2614	1,8487	2,367	2,777	0,1480	**0,0658**
8	1 043	-0,2598	0,5685	2,455	2,677	0,1083	**0,0927**
9	611	1,0950	1,6069	2,404	2,734	0,1392	**0,0664**

Продовження табл. 2.3.

1	2	3	4	5	6	7	8
10	861	-0,0845	0,6518	2,434	2,700	0,1252	**0,0776**
11	1 186	0,4952	0,9510	2,445	2,688	0,1080	0,1233
12	2 247	1,2798	1,7295	2,347	2,800	0,2546	0,4307
13	993	1,0253	1,4115	2,446	2,688	0,1169	**0,0922**
14	946	1,0564	1,4615	2,456	2,676	0,1094	**0,0863**
15	993	2,0282	2,7177	2,397	2,742	0,1745	**0,1612**
16	581	0,7998	1,2518	2,294	2,865	0,2198	**0,1400**
17	2 397	0,7710	1,5507	2,383	2,758	0,1832	0,3758
18	647	0,7260	1,2211	2,424	2,711	0,1303	**0,0886**
19	1 322	2,4491	3,3404	2,373	2,770	0,2219	**0,2171**
20	315	0,6341	1,2244	2,381	2,760	0,1529	**0,0608**
21	370	0,5340	1,1932	2,354	2,792	0,1543	**0,0593**
22	342	1,9716	2,5751	2,314	2,840	0,1754	**0,0794**
23	604	1,1486	1,6669	2,439	2,695	0,1272	**0,0708**
24	930	1,3023	1,8368	2,456	2,676	0,1094	**0,0863**
25	2 441	2,1967	2,6454	2,302	2,856	0,2987	0,4849
26	2 754	1,6070	2,3863	2,337	2,812	0,3156	0,4764
27	1 028	2,6105	3,1389	2,435	2,699	0,1249	**0,0941**
28	1 473	2,2838	2,6403	2,353	2,794	0,2040	0,3011
29	945	2,0791	2,6699	2,340	2,808	0,1980	**0,1388**
30	522	2,0848	2,5719	2,392	2,748	0,1446	**0,0632**
31	2 397	2,4521	3,1216	2,385	2,756	0,1789	0,3642
32	779	2,7006	3,1224	2,428	2,707	0,1334	**0,0745**
33	2 416	3,7664	4,7388	2,395	2,745	0,1623	0,3208
34	1 500	2,7401	3,1082	2,433	2,701	0,1131	0,1714
35	3 804	2,2178	2,7287	2,333	2,818	0,3021	0,6322
36	1 213	3,0323	3,5538	2,326	2,826	0,1790	0,1959
37	2 238	2,6389	3,0444	2,373	2,770	0,2118	0,4183
38	2 321	2,0791	2,6932	2,384	2,757	0,1941	0,3218
39	767	3,2258	3,7350	2,377	2,765	0,1585	**0,1081**
40	613	2,9741	3,6038	2,315	2,839	0,1737	**0,1292**
41	1 154	3,5904	4,1712	2,433	2,701	0,1284	**0,1179**
42	1 342	3,9027	4,8690	2,451	2,681	0,1114	**0,0932**
43	752	4,1865	5,0151	2,348	2,800	0,1639	**0,1055**
44	1 002	4,6572	5,5134	2,442	2,691	0,1245	**0,0711**
45	3 989	5,6767	6,6322	2,321	2,832	0,3663	0,8080

Визначені ширині довірчих інтервалів демонструють зменшення ширини цих інтервалів у випадку застосування нелінійної регресії. Це можна спостерігати для рядків таблиці: 2-5, 7-10, 13-16, 18-24, 27, 29-30, 32, 39-44.

Для визначення інтервалів передбачення скористаємось формулами (1.3) та (1.9) відповідно. У таблиці 2.4 показано інтервали передбачення лінійної та нелінійної двохфакторних регресійних моделей та їх ширини.

Таблиця 2.4 – Порівняння ширин інтервалів передбачення лінійної та нелінійної регресії

№	KLOC (Y)	Межі інтервалів передбачення				Ширина інтервалу	
		Лінійна регресія		Нелінійна регресія		Лінійна регресія	Нелінійна регресія
		LB	UB	LB	UB		
1	2	3	4	5	6	7	8
1	2 813	-0,1192	2,1212	1,905	3,451	0,7279	1,5463
2	342	-0,5703	1,5918	1,905	3,450	0,7157	**0,2499**
3	974	-0,5703	1,5918	1,909	3,444	0,7133	**0,5640**
4	284	-0,2910	1,8756	1,901	3,458	0,7158	**0,2085**
5	670	-1,0291	1,1772	1,911	3,439	0,7128	**0,3872**
6	1 052	-1,0857	1,1185	1,915	3,432	0,7095	0,7249
7	404	0,4634	2,6467	1,903	3,454	0,7140	**0,2485**
8	1 043	-0,9757	1,2844	1,918	3,427	0,7069	**0,6297**
9	611	0,2688	2,4331	1,910	3,441	0,7123	**0,3077**
10	861	-0,8304	1,3977	1,915	3,432	0,7097	**0,4419**
11	1 186	-0,3527	1,7989	1,917	3,429	0,7068	0,7688
12	2 247	0,4295	2,5798	1,899	3,462	0,7435	1,4882
13	993	0,1494	2,2874	1,917	3,429	0,7083	**0,5763**
14	946	0,1882	2,3297	1,918	3,427	0,7071	**0,5925**
15	993	1,2665	3,4795	1,909	3,443	0,7200	**0,7171**
16	581	-0,0497	2,1012	1,884	3,488	0,7323	**0,3931**
17	2 397	0,0395	2,2822	1,906	3,448	0,7222	1,5476
18	647	-0,1066	2,0537	1,914	3,435	0,7106	**0,4706**
19	1 322	1,7527	4,0367	1,904	3,451	0,7330	0,8448
20	315	-0,1628	2,0213	1,906	3,448	0,7151	**0,2473**
21	370	-0,2382	1,9655	1,900	3,459	0,7154	**0,2114**
22	342	1,1795	3,3672	1,890	3,477	0,7202	**0,2397**
23	604	0,3248	2,4906	1,916	3,431	0,7100	**0,4176**
24	930	0,4847	2,6544	1,918	3,427	0,7071	**0,5925**
25	2 441	1,3460	3,4962	1,887	3,484	0,7597	1,3975
26	2 754	0,8754	3,1180	1,896	3,466	0,7665	1,5751
27	1 028	1,7906	3,9588	1,915	3,432	0,7096	**0,5418**
28	1 473	1,3956	3,5285	1,900	3,459	0,7277	1,0652
29	945	1,2824	3,4666	1,897	3,465	0,7261	**0,4650**
30	522	1,2491	3,4076	1,908	3,445	0,7134	**0,2732**

Продовження табл. 2.4.

1	2	3	4	5	6	7	8
31	2 397	1,6835	3,8903	1,907	3,447	0,7211	1,5098
32	779	1,8391	3,9839	1,914	3,434	0,7112	**0,4050**
33	2 416	3,0942	5,4110	1,909	3,444	0,7172	1,4067
34	1 500	1,8568	3,9916	1,915	3,432	0,7076	0,9706
35	3 804	1,3913	3,5553	1,895	3,468	0,7611	2,0511
36	1 213	2,2098	4,3763	1,893	3,471	0,7211	**0,6192**
37	2 238	1,7708	3,9124	1,904	3,451	0,7299	1,6287
38	2 321	1,2909	3,4815	1,907	3,447	0,7250	1,3295
39	767	2,3986	4,5622	1,905	3,450	0,7163	**0,4305**
40	613	2,1914	4,3865	1,890	3,477	0,7198	**0,3910**
41	1 154	2,7900	4,9716	1,915	3,432	0,7103	**0,6677**
42	1 342	3,2287	5,5429	1,917	3,428	0,7074	**0,6129**
43	752	3,4707	5,7309	1,899	3,462	0,7175	**0,3647**
44	1 002	3,9501	6,2206	1,916	3,430	0,7095	**0,4325**
45	3 989	4,9996	7,3093	1,892	3,474	0,7887	2,5012

Визначені ширині інтервалів прогнозування демонструють зменшення ширини цих інтервалів у випадку застосування нелінійної регресії. Це можна спостерігати для рядків таблиці: 2-5, 7-10, 13-16, 18, 20-24, 27, 29-30, 32, 36, 39-44.

Також у випадку застосування лінійної регресії серед нижніх границь інтервалів трапляються від'ємні значення, що є недопустимим.

При порівнянні лінійної двохфакторної регресійної моделі з нелінійною можна спостерігати покращення показника відсотку прогнозованих результатів $PRED(0{,}25)$ на 0,107 на користь нелінійної моделі. Отримане покращення разом з меншими ширинами довірчих інтервалів та інтервалів прогнозування для 29 рядків (що складає 64,4% від кількості проектів) є ваговим аргументом на користь застосування саме нелінійної регресійної моделі.

У відкритих джерелах не було знайдено двохфакторних нелінійних регресійних моделей для мов програмування та фреймворків, зокрема для JavaScript та NestJS. В основному присутні моделі з кількістю факторів від трьох та більше, але наявні й однофакторні моделі. Тому зважаючи на обмеження в кількості зібраних метрик було вирішено побудувати однофакторну нелінійну регресійну модель та порівняти її з існуючою моделлю для мови програмування

Java [11]. Для побудови такої однофакторної моделі використовуватимемо кількість рядків коду в тисячах та кількість класів.

Отримана нелінійна однофакторна регресійна модель має наступні коефіцієти: $\hat{b}_0 = -1.672,\ \hat{b}_1 = 0.194$. Порівняння регресійних моделей для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, наведено у таблиці 2.5.

Таблиця 2.5 – Порівняння регресійних моделей

Оцінки якості моделі	Модель		
	Нелінійна однофакторна	Лінійна двохфакторна	Нелінійна двохфакторна
R^2	0,781	0,996	0,996
$MMRE$	0,304	0,115	0,114
$PRED(0{,}25)$	0,481	0,888	0,955

З таблиці 2.5 бачимо, що застосування багатофакторного регресійного аналізу значно підвищує показники якості отриманих моделей у порівнянні з використанням однофакторної моделі.

Також для проведення порівняння до емпіричних даних було застосовано модель [11] для оцінювання розміру веб-застосунків, реалізованих мовою Java. Отримані оцінки якості при застосуванні цієї моделі значно поступаються відповідним оцінкам нелінійної однофакторної регресійної моделі для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS. Це підтверджує залежність моделей, що розробляються для прогнозування розміру програмного забезпечення, від мови програмування, фреймворків та типу програмного забезпечення, а також дозволяє стверджувати, що для отримання якісних моделей необхідно будувати модель під обрану мову програмування, фреймворк, тощо.

2.3 Висновки до розділу 2

В даному розділі було побудовано нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS на основі нормалізуючого перетворення десяткового логарифму.

В розділі також:

– Виконано пошук проектів серверних застосунків та виконано збір метрик по ним за допомогою сайту GitHub та програми SonarQube відповідно.

– Перевірено вихідні дані на нормальність та визначено, що вони не підпорядковуються нормальному розподілу.

– Перевірено вихідні дані на мультиколінеарность серед предикторів. Мультиколінеарність відсутня.

– Виконано пошук та вилучення викидів за допомогою методу, що базується на нормалізуючих перетвореннях та розрахунку квадрату відстані Махаланобіса. Процес пошуку викидів зайняв 6 ітерацій, 17 проектів серверних застосунків визначено як викиди.

– Для 45 проектів, які залишились після пошуку викидів, побудовано двохфакторну нелінійну регресійну модель з використанням нормалізуючого перетворення десяткового логарифма.

– Побудовано двохфакторну лінійну регресійну модель для порівняння з побудованою нелінійною регресійною моделлю.

– Проведено порівняння побудованих моделей. Для нелінійної регресійної моделі оцінки якості наступні: $R^2 = 0{,}966$, $MMRE = 0{,}114$, $PRED(0{,}25) = 0{,}955$. Для лінійної моделі маємо такі оцінки: $R^2 = 0{,}966$, $MMRE = 0{,}115$, $PRED(0{,}25) = 0{,}888$. Бачимо покращення показника $PRED(0{,}25)$ для нелінійної регресійної моделі.

– Проведено обчислення довірчих інтервалів та інтервалі прогнозування лінійної та нелінійної регресій. Визначено, що для 64% даних характерне

зменшення ширини довірчих інтервалів та інтервалів прогнозування у випадку використання нелінійної регресійної моделі.

– Побудовано нелінійну однофакторну модель з кількістю класів у якості предиктора та проведено порівняння отриманої моделі з моделлю для мови програмування Java, застосованої до емпіричних даних по фреймворку NestJS. Порівняння показало необхідність побудови моделей під конкретну мову програмування чи фреймворк.

3 РОЗРОБКА ПРОЕКТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ СЕРВЕРНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NESTJS

3.1 Постановка задачі на розробку проекту програмного забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

Розробка програмного забезпечення передбачає досить великий перелік заходів щодо планування, проектування, тощо. Одним з важливих питань – є вибір програмних засобів для реалізації розроблюваної програми, зокрема мови програмування. Обираючи мову програмування та інші програмні засоби слід враховувати, чи потрібно буде користувачу виконувати додаткові дії щодо налаштування середовища виконання, тощо.

Оптимальний вибором може слугувати мова Python, адже вона має повну сумісність з розповсюдженими операційними системами: Windows, Linux, MacOS. Python – мова програмування високого рівня з динамічною строгою типізацією з відкритим кодом. В якості інших значущих причин вибору саме цієї мови можна вказати:

– Наявність готових рішень, що обумовлено високою популярністю цієї мови.

– Велику кількість бібліотек, зокрема бібліотек NymPy та SciPy для математичних та статистичних обчислень.

– Можливість використання сумісної з Python бібліотеки Qt для розробки графічного інтерфейсу користувача.

– Великий вибір текстових редакторів.

Зважаючи на описані вище переваги від використання мови Python, для розробки програмного забезпечення в межах роботи використовуватимемо саме Python.

Розробка програми для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, буде проводитись згідно технічного завдання, наведеного у додатку А. Розробка програми передбачає розробку ескізного, технічного та робочого проектів.

3.2 Розробка ескізного проекту програмного забезпечення оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

За допомогою діаграми варіантів використання та діаграми діяльності проведемо моделювання концептуального рівня розроблюваного програмного забезпечення.

3.2.1 Побудова моделі варіантів використання програмного застосунку «NestJS KLOC estimation»

Діаграма варіантів використання (або прецедентів) використовується для відображення операцій (варіантів використання, прецедентів), які можуть виконувати користувачі (актори). Для цієї діаграми важливим є акцент на виокремленні операцій без деталей їх наступної реалізації.

Користувачем програми буде один користувач, отже діаграма варіантів використання буде мати лише одного актора.

Виявлені варіанти використання наведено у таблиці 3.1.

Таблиця 3.1 – Виявлені варіанти використання

Основні актори	Найменування	Формулювання
Користувач	Ввести кількість класів	Користувач вводить кількість класів серверного застосунку
Користувач	Ввести кількість функцій	Користувач вводить кількість функцій серверного застосунку
Користувач	Ввести довірчу ймовірність	Користувач вводить довірчу ймовірність щоб оцінити інтервал передбачення та довірчий інтервалу серверного застосунку
Користувач	Оцінити розмір	Користувач отримує оцінку розміру серверного застосунку
Користувач	Оцінити довірчий інтервал	Користувач отримує оцінку довірчого інтервалу серверного застосунку
Користувач	Оцінити інтервал передбачення	Користувач отримує оцінку інтервалу передбачення серверного застосунку

Діаграму варіантів використання наведено на рисунку 3.1.

Рисунок 3.1 – Діаграма варіантів використання

Отриману діаграму варіантів використання використовуватимемо в подальшому в проектуванні в ескізному, технічному та робочому проектах.

3.2.2 Специфікації варіантів використання програмного застосунку «NestJS KLOC estimation»

Наведемо детальну специфікацію варіантів використання

1. *Найменування:* Ввести кількість класів

Актори: користувач

Зв'язок з іншими варіантами використання: відсутній

Короткий опис: варіант використання «Ввести кількість класів» призначення для вказання кількості класів серверного застосунку, для якого буде оцінювання, у відповідне поле графічного інтерфейсу.

Основний потік подій: варіант використання розпочинає виконання при взаємодії користувача та поля графічного інтерфейсу.

Базовий потік: Ввести кількість класів серверного застосунку:

1) Користувач вказує кількість класів серверного застосунку у поле графічного інтерфейсу.

Спеціальні вимоги:

1) Значення кількості класів – натуральне число.

Подібно до зазначеного вище варіанту використання виконуються варіанти використання «Ввести кількість функцій» та «Ввести довірчу ймовірність». Для варіанту використання «Ввести кількість функцій» в якості вхідних даних виступає натуральне число, для варіанту використання «Ввести довірчу ймовірність» – дійсне додатне число.

2. *Найменування:* Оцінити розмір

Актори: користувач

Зв'язок з іншими варіантами використання: відсутній

Короткий опис: варіант використання «Оцінити розмір» призначений для виконання розрахунків для отримання оцінки розміру серверного застосунку. Результат роботи варіанту використання – оцінка розміру серверного застосунку на екрані інтерфейсу програми.

Основний потік подій: варіант використання починає роботу автоматично після введення кількості класів, кількості функцій, довірчої ймовірності.

Базовий потік: Оцінити розмір серверного застосунку:

1) Користувач вказує кількість класів, кількість функцій та довірчу ймовірність для виконання оцінювання розміру серверного застосунку.

2) Система відображає отриману оцінку розміру серверного застосунку.

Спеціальні вимоги: відсутні.

Альтернативні потоки: відсутні.

3. *Найменування:* Оцінити довірчий інтервал

Актори: користувач

Зв'язок з іншими варіантами використання: відсутній

Короткий опис: варіант використання «Оцінити довірчий інтервал» призначений для виконання розрахунків для отримання оцінки довірчого інтервалу серверного застосунку. Результат роботи варіанту використання – оцінка довірчого інтервалу серверного застосунку на екрані інтерфейсу програми.

Основний потік подій: варіант використання починає роботу автоматично після введення кількості класів, кількості функцій, довірчої ймовірності.

Базовий потік: Оцінити довірчий інтервал серверного застосунку:

1) Користувач вказує кількість класів, кількість функцій та довірчу ймовірність для виконання оцінювання розміру серверного застосунку.

2) Система відображає отримані межі довірчого інтервалу розміру серверного застосунку.

Спеціальні вимоги: відсутні.

Альтернативні потоки: відсутні.

Аналогічно можна описати поведінку для варіанту використання «Виконати оцінювання інтервалу передбачення».

3.2.3 Моделювання сценарію виконання програмного застосунку «NestJS KLOC estimation» за допомогою діаграми діяльності

Діаграма діяльності має на меті передати процес виконання операції з акцентом на семантиці станів. Діаграма діяльності допомагає відобразити алгоритм виконання операції, адже кожен елемент діаграми, тобто, кожна діяльність – це певна сукупність дій або обчислень, які виконує автомат.

Елементами діаграми є:

– Стан – певна ситуація, в якій опинився об’єкт за результатами виконання вхідної дії. Стан має хоча б один результуючий перехід.

– Перехід між станами – передача управління в результаті виконання вихідного стану до наступного.

Процес виконання операцій під час виконання програмного застосунку для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, наведено на рисунку 3.2.

Рисунок 3.2 – Діаграма діяльності

Отримана діаграма в подальшому буде використана при реалізації алгоритмів виконання програмного забезпечення.

3.3 Розробка технічного проекту програмного забезпечення оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

3.3.1 Статична модель програмного застосунку «NestJS KLOC estimation» у вигляді діаграми класів

При переході від концептуального рівня проектування розроблюваної системи до деталей її реалізації будують діаграму класів. Ця діаграма покликана відобразити статичну структуру програмного забезпечення, показує класи, відношення між ними, їх внутрішню структуру, пакети, інтерфейси.

Діаграму класів для програмного застосунку для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS наведено на рисунку 3.3.

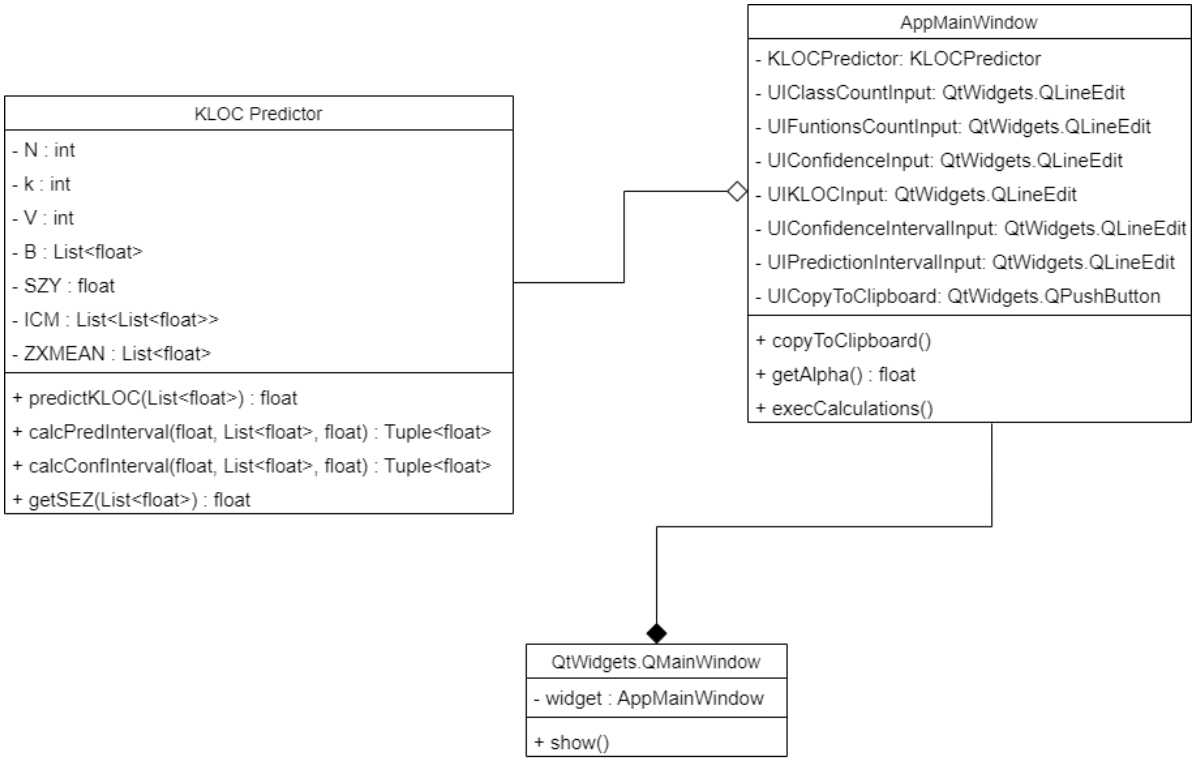

Рисунок 3.3 – Діаграма класів

Побудована діаграма слугуватиме для кодогенерації та безпосередньому написанні програмного коду на етапі реалізації програмного забезпечення.

3.3.2 Специфікації класів програмного застосунку «NestJS KLOC estimation»

Коротку специфікацію класів можна представити наступним чином:

– Клас AppMainWindow – клас-контролер, що оброблює дії користувача та введення даних у поля графічного інтерфейсу.

– Клас QtWidget – клас, що є базовим відносно компонентів інтерфейсу користувача бібліотеки Qt.

– Клас KLOCPredictor – клас, який виконує оцінювання розміру, довірчого інтервалу та інтервалу передбачення обраного серверного застосунку.

3.3.3 Динамічна модель програмного застосунку «NestJS KLOC estimation» у вигляді діаграм послідовностей

Діаграма послідовності покликана відобразити деталі реалізації алгоритму для виконуваної операції. Часто в якості операції обирають варіант використання. Ця діаграма містить:

– об’єкти, що взаємодіють між собою;

– фокус управління;

– повідомлення, якими обмінюються об’єкти.

Діаграму послідовності для варіанту використання «Ввести кількість класів» наведено на рисунку 3.4:

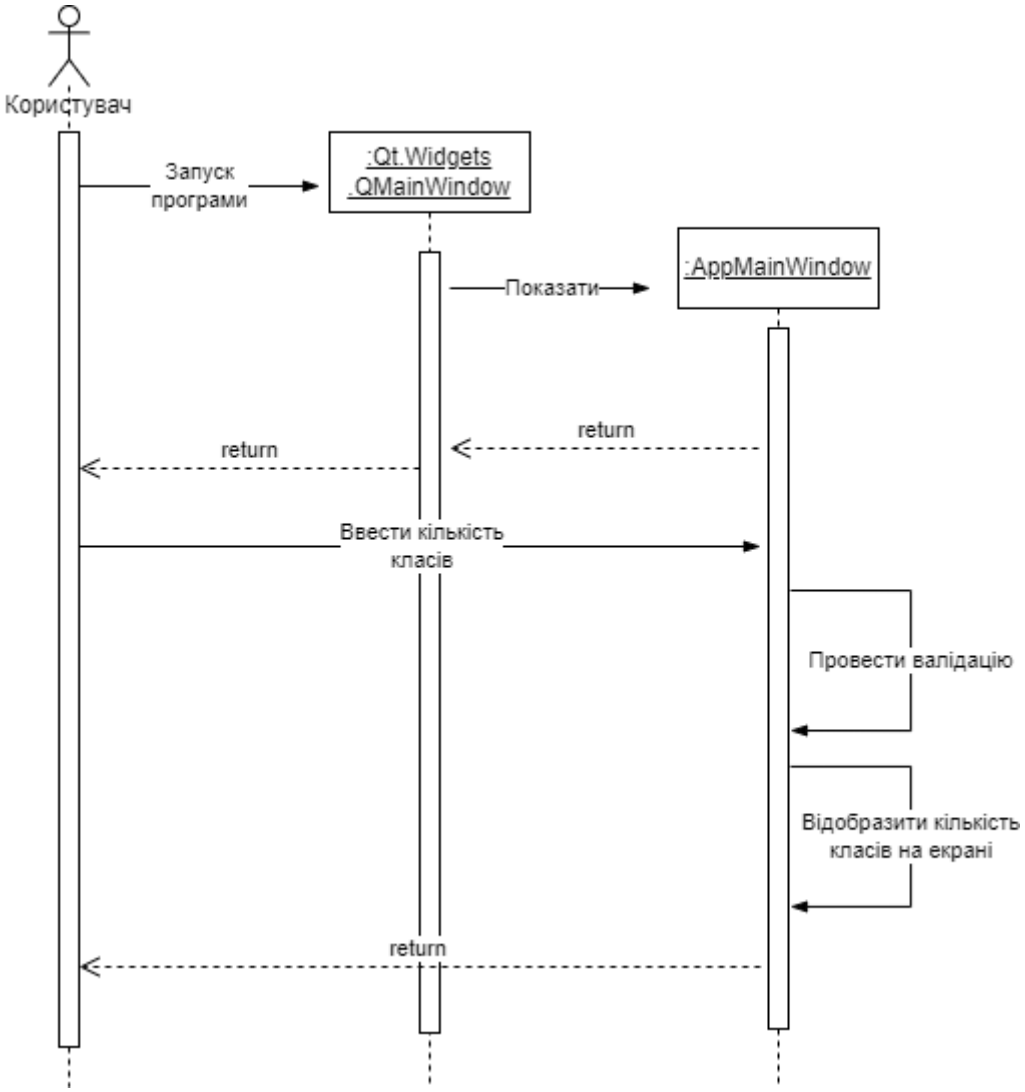

Рисунок 3.4 – Діаграма послідовності для варіанту використання «Ввести кількість класів»

Діаграму послідовності для варіанту використання «Оцінити розмір» наведено на рисунку 3.5:

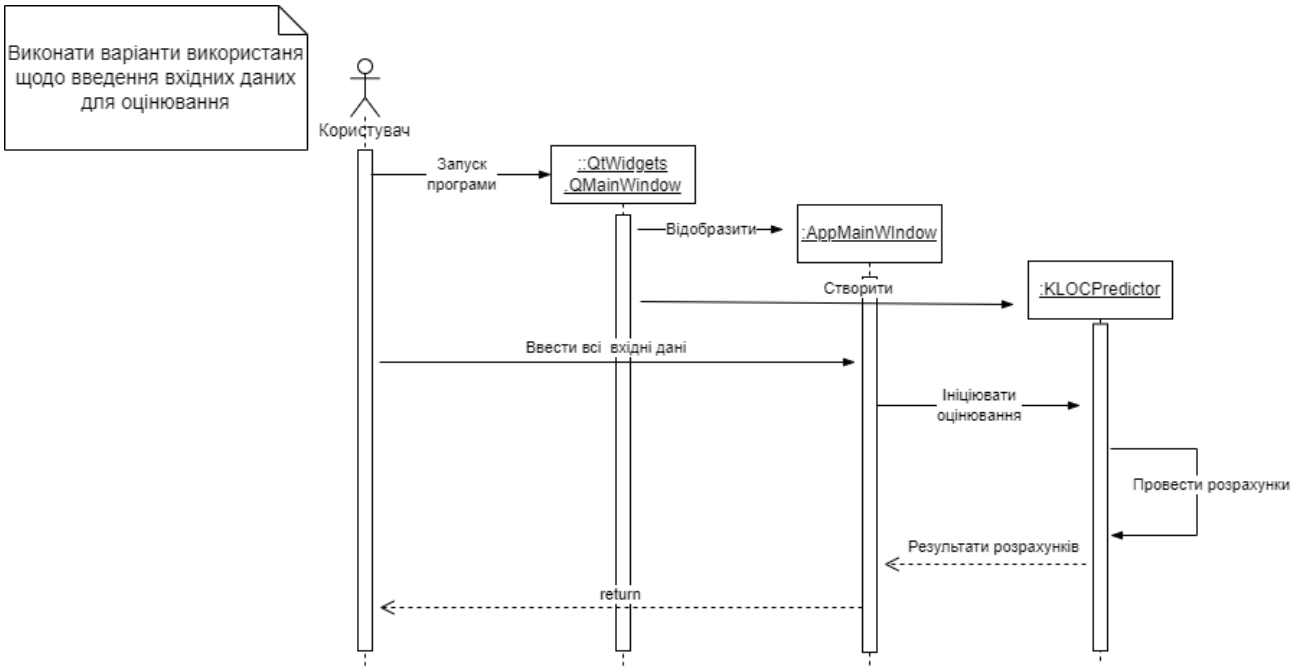

Рисунок 3.5 – Діаграма послідовності для варіанту використання «Оцінити розмір»

Проведене моделювання поведінки для основних варіантів використання полегшить роботу під час написання коду для їх реалазації.

3.4 Розробка робочого проекту програмного забезпечення оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

3.4.1 Кодування програмного застосунку «NestJS KLOC estimation»

Для написання коду в межах роботи було обрано мову Python. Нижче наведено код основного класу програмного застосунку, який відповідає за виконання розрахунків, необхідних для виконання оцінювання.

```
class KLOCPredictor:
    N = 45
    k = 2
    V = N - k - 1
```

```
    B = [
        -1.99787698508896,
        0.187113121751769,
        0.884520277756632
    ]

    SZY = 0.061780775235153

    # Inverse covariance matrix
    ICM = np.array([
        [1.13885728885, -0.72182851705254],
        [-0.72182851705254, 0.731137536286726]
    ])

    ZXMeans = [1.43484, 1.97255]

    def predictKLOC(self, X):
        return (10 ** self.B[0]) * (X[0] ** self.B[1]) * (X[1] ** self.B[2])

    def getSEZ(self, X):
        zx = np.log10(X) - self.ZXMeans
        return np.matmul(np.matmul(zx.T, self.ICM), zx)

    def calcPredInterval(self, value, X, alpha):
        Y = math.log10(value)
        sez = self.getSEZ(X)
        predHalfIntervalValue = (stats.t.ppf(1 - alpha / 2, self.V) * self.SZY *
math.sqrt((1 + 1.0 / self.N + sez)))
        return 10 ** (Y - predHalfIntervalValue), 10 ** (Y +
predHalfIntervalValue)

    def calcConfInterval(self, value, X, alpha):
        Y = math.log10(value)
        sez = self.getSEZ(X)
        confHalfIntervalValue = (stats.t.ppf(1 - alpha / 2, self.V) * self.SZY *
math.sqrt((1.0 / self.N + sez)))
        return 10 ** (Y - confHalfIntervalValue), 10 ** (Y +
confHalfIntervalValue)
```

Повний текст програми наведено в додатку Г.

3.4.2 Тестування програмного застосунку «NestJS KLOC estimation»

За результатами завершення етапу написання коду програми було проведено тестування, розроблено програмну документацію згідно вимог до проекту, а також проведено приймально-здавальні випробування.

Для розробки тестів в межах даної роботи використаємо метод еквівалентного розбиття (класів еквівалентності).

Проведемо тестування варіанту використання «Ввести кількість класів» за допомогою еквівалентного розбиття. Класи еквівалентності наведено у таблиці 3.2.

Таблиця 3.2 – Перелік класів еквівалентності для тестування варіанту використання «Ввести кількість класів»

Вхідні данні	*Правильні класи*	*Неправильні класи*
Кількість класів	Введено натуральне число (1)	*Введено дійсне число (2).* *Введено від'ємне число (3).* *Введено не число (4).* Пусте поле (5).

Результати виконання тестування з використанням правильних класів еквівалентності наведено у таблиці 3.3.

Таблиця 3.3 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	48	Можливість виконати оцінювання	Пройдено

Результати виконання тестування з використанням неправильних класів еквівалентності наведено у таблиці 3.4.

Таблиця 3.4 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	2	10,56	Повідомлення про недопустиме значення кількості класів	Пройдено
3	3	-10,55	Повідомлення про недопустиме значення кількості класів	Пройдено
4	4	Фіа293!;	Повідомлення про недопустиме значення кількості класів	Пройдено
5	5	Відсутні	Повідомлення про необхідність заповнити поле	Пройдено

Класи еквівалентності для варіанту використання «Ввести кількість функцій» наведено у таблиці 3.5.

Таблиця 3.5 – Перелік класів еквівалентності для тестування варіанту використання «Ввести кількість функцій»

Вхідні данні	*Правильні класи*	*Неправильні класи*
Кількість функцій	Введено дійсне додатне число (1)	*Введено дійсне число (2).* *Введено від'ємне число (3).* *Введено не число (4).* Пусте поле (5).

Результати виконання тестування з використанням правильних класів еквівалентності наведено у таблиці 3.6.

Таблиця 3.6 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	19	Можливість виконати оцінювання	Пройдено

Результати виконання тестування з використанням неправильних класів еквівалентності наведено у таблиці 3.7.

Таблиця 3.7 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	2	10,56	Повідомлення про недопустиме значення кількості функцій	Пройдено
3	3	-10,55	Повідомлення про недопустиме значення кількості функцій	Пройдено
4	4	Фіа293!;	Повідомлення про недопустиме значення кількості функцій	Пройдено
5	5	Відсутні	Повідомлення про необхідність заповнити поле	Пройдено

Виконаємо тестування варіанту використання «Ввести довірчу ймовірність» за допомогою еквівалентного розбиття. Класи еквівалентності наведено у таблиці 3.8.

Таблиця 3.8 – Перелік класів еквівалентності для тестування варіанту використання «Ввести довірчу ймовірність»

Вхідні данні	*Правильні класи*	*Неправильні класи*
Довірча ймовірність	Введено дійсне додатне число в межах від нуля до одиниці включно (1)	*Введено від'ємне число (2).* *Введено не число (3).* Пусте поле (4).

Результати виконання тестування з використанням правильних класів еквівалентності наведено у таблиці 3.9.

Таблиця 3.9 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	0,95	Можливість виконати оцінювання	Пройдено

Результати виконання тестування з використанням неправильних класів еквівалентності наведено у таблиці 3.10.

Таблиця 3.10 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	2	-10,55	Повідомлення про недопустиме значення довірчої ймовірності	Пройдено
3	3	Фіа293!;	Повідомлення про недопустиме значення довірчої ймовірності	Пройдено
4	4	Відсутні	Повідомлення про необхідність заповнити поле	Пройдено

Результати тестування дозволяють зробити висновок про належну роботу програми.

3.4.3 Випробування програмного застосунку «NestJS KLOC estimation»

Згідно документу «Програма та методика випробувань», який наведено у Додатку Б, було проведено випробування розробленого програмного застосунку.

Перелік протестованих функцій:

– Введення кількості класів серверного застосунку.

– Введення кількості функцій серверного застосунку.

– Введення довірчої ймовірності для оцінювання серверного застосунку.

– Проведення оцінки розміру серверного застосунку, оцінок довірчого інтервалу та інтервалу передбачення серверного застосунку.

Програмне забезпечення виконує всі функції належним чином.

Результат роботи програми для оцінювання розміру серверного застосунку, що створюється з використанням фреймворку NestJS, наведено на рисунку 3.6 та 3.7.

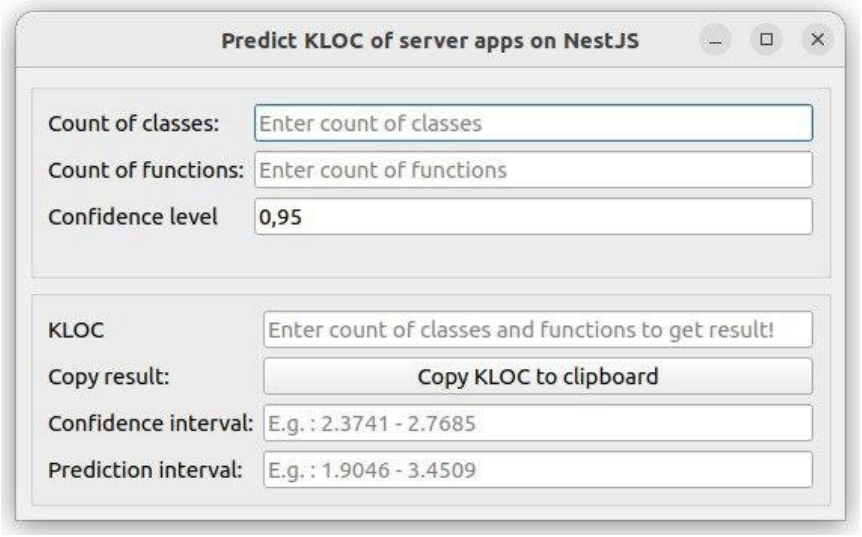

Рисунок 3.6 – Вікно програми до виконання оцінювання

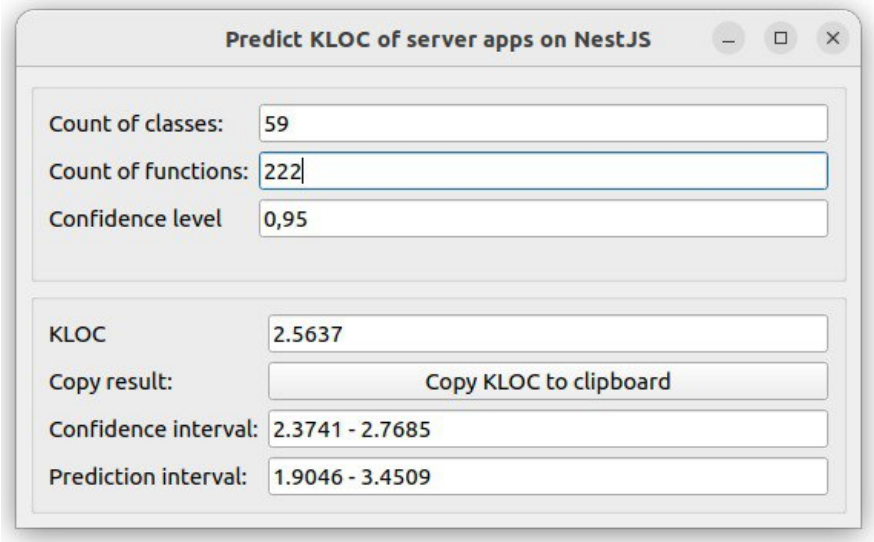

Рисунок 3.7 – Результат оцінювання розміру серверного застосунку, що створюється з використанням фреймворку NestJS

Отримані результати показують, що розроблена програма працює належним чином та відповідає висунутим вимогам.

3.5 Висновки до розділу 3

В даному розділі було проведено розробку проекту програмного забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

Також в розділі було:

– Виконано вибір мови програмування та інших програмних засобів для наступного проведення розробки.

– Виконано розробку ескізного проекту програмного забезпечення. Ескізний проект включає в себе окреслення меж розроблюваної системи за допомогою визначення варіантів використання з побудовою відповідної діаграми та специфікацій до неї.

– Виконано розробку технічного проекту програмного забезпечення. Технічний проект включає в себе діаграму класів як спосіб зображення статичної

структури розроблюваної системи. До отриманої діаграми класів було створено специфікації класів. Також за діаграмою класів було побудовано діаграми послідовностей для обраних варіантів використання.

– Виконано розробку робочого проекту програмного забезпечення. Робочий проект включає в себе заходи щодо безпосередньої реалізації розроблюваної системи: кодування, тестування, випробовування програми. Тестування виконувалось за допомогою методики еквівалентного розбиття.

–

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ СЕРВЕРНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NESTJS

Вступ

Розробка програмного забезпечення передбачає витрати на купівлю техніки, наймання та підготовку кадрів, витрати на підготовчі роботи та розробку безпосередньо. Визначити економію від використання програмного забезпечення можна зважаючи на витрати при його розробці. Відношення економії до витрат характеризує економічну ефективність. Нашою метою є визначити ключові економічні показники, які слід урахувати, для розроблюваного програмного забезпечення та на їх основі проаналізувати доцільність такої розробки.

4.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Можна виділити наступний перелік витрат, що складають витрати на розробку:

– Витрати на заробітну плату.

– Витрати на амортизацію ОЕМ для виконання розробки.

– Витрати на експлуатацію цієї ЕОМ.

– Витрати на засоби розробки.

– Витрати на матеріали та комплектуючі.

Розробку буде виконувати програміст з заробітною платою 10000грн. на місяць.

Вартість сучасного та підходящого для виконання роботи ПК (чи ноутбука) становить приблизно 15000 грн.

Вартість кіловат-години електроенергії становить 1,68 грн.

Також слід врахувати інші можливі витрати, які наведено у таблиці 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Папір для принтера	250
2	Картридж та тонер для принтера	400
3	Використання флеш-накопичувача	150
4	Непередбачувані витрати	500
	Разом	1300

Тоді вартість розробки програмного забезпечення можна розрахувати як

$$С_{пр} = (З_{зп} + З_{зс} + З_{зг} + З_{е}) * Т + З_{м},$$

де Т – тривалість розробки у місяцях

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

$З_{зс}$ – відрахування на соціальні заходи (15% від основної та додаткової заробітної плати), грн.

$З_{е}$ – витрати на електроенергію, грн.

$З_{м}$ – витрати на допоміжні матеріали, грн.

$З_{е}$ при витратах в 0,5 кВт з тривалістю роботи на місяць 22*8 = 176 годин та вартості електроенергії 1,68грн становить

$$З_{е} = 176 * 1,68 * 0,5 = 147,84 \text{ грн.} \quad (4.1)$$

Деталізовані витрати на розробку програмного забезпечення наведено у таблиці 4.2

Таблиця 4.2 – Витрати на розробку програмного забезпечення

№	Пункти витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	2
2	Основна та додаткова заробітна плата (Ззп)	Грн.	10000
3	Відрахування на соціальні витрати (Ззс)	Грн.	1500
4	Загальногосподарські витрати (Ззг)	Грн.	1000
5	Витрати на допоміжні матеріали (Зм)	Грн.	1300
6	Витрати на електроенергію *(Зе)	Грн.	147,84

За формулою (4.1) виконаємо розрахунок вартості програми:

$$С_{пр} = (10000 + 1500 + 1000 + 147.84) * 2 + 1300 = 26595,68 \text{ грн.}$$

Витрати на амортизацію становлять 25% від балансової вартості на рік:

$$А_{об} = 15000 * 0,25 = 3750 \text{ грн.}$$

Для підприємств річні витрати на основні та допоміжні матеріали складають 5% від вартості основного устаткування:

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

Річний обсяг робіт ПК у годинах можна визначити як

$$Ф_{м} = 264,5 * Т_{з}, \quad (4.2)$$

де Тз – середнє навантаження на устаткування в місяць (6 годин), 264,5 – середня кількість робочих днів на рік.

Отже, річний обсяг роботи ПК складає:

$$Ф_{м} = 264,5 * 6 = 1578 \text{ годин}$$

Витрати на електроенергію Зе при 1578 годинах роботи устаткування на рік становлять:

$$З_{е} = 1587 * 1,68 = 2666,16$$

Експлуатаційні витрати на ПК на рік становлять:

$$З_{зр} = 3750 + 750 + 2666,16 = 7166,16$$

Таким чином витрати на розробку та експлуатацію програмного забезпечення в перший рік становлять:

$$З_{р} = 26595,68 + 7166,16 = 33761,84$$

4.2 Економічна ефективність розробки та впровадження програмного забезпечення

Показник наявності економічної ефективності від використання програмного забезпечення щодо оцінювання розміру програмного забезпечення – покращення часових витрат та трудомісткості проведення такого оцінювання.

За експертною оцінкою фахівця підприємства, впровадження програмного забезпечення дозволить вивільнити 1 працівника.

Заробітна плата одного працівника на рік складає:

$$З = 10500 * 12 * 1 = 126000 \text{ грн.}$$

Отримати річний економічний ефект можна за допомогою формули:

$$E_{рік} = \Delta C_n - E_n * k, \quad (4.3)$$

де ΔE_n – кошти, що буде вивільнено при впровадженні програмного забезпечення (126000) без експлуатаційних витрат (33761,84), тобто, 92238,16 грн.;

E_n – коефіцієнт ефективності (такий же як й коефіцієнт амортизації – 0,25)

k – одноразові витрати на впровадження програмного забезпечення (26595,68 грн.).

$$E_{рік} = 126000 - 0{,}25 * 26595{,}68 = 119351{,}08$$

Термін, за який програмне забезпечення окупиться наступний:

$$T = \frac{k}{\Delta C_n} = \frac{26595{,}68}{92238{,}16} = 0{,}28 \text{ року}$$

Отже термін, за який програмне забезпечення окупиться – приблизно 4 місяці.

4.3 Висновки до розділу 4

В розділі було виконано розрахунок витрат, економічної ефективності та терміну, за який програмне забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, окупиться. За результатами розрахунків було визначено, що це займе 4 місяці. Отже, розробка програмного забезпечення є доцільною з економічної точки зору.

5 ОХОРОНА ПРАЦІ

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

Роботодавець – власник підприємства, установи, організації або уповноважений ним орган, незалежно від форм власності, виду діяльності, господарювання, і фізична особа, яка використовує найману працю.

Працівник – особа, яка працює на підприємстві, в організації, установі та виконує обов'язки або функції згідно з трудовим договором (контрактом).

Метою охорони праці є забезпечення безпечних умов праці, тобто, таких умов, за яких є відсутнім виникнення травматизму на робочому місці, захворювань, аварійних ситуацій. Охорона праці здійснюється через реалізацію практичних та наукових задач. Практичні задачі – розробка заходів щодо досягнення вищеописаних цілей. Наукові задачі – отримані нових знань, що засновані на аналізі технологічних процесів.

Основне завдання охорони праці – покращення умов праці. До завдань охорони праці також відносять:

1) Знаходження оптимальних співвідношень між різними факторами виробничого середовища.

2) Впровадження норм гранично допустимих рівнів виробничих факторів, визначення ступеня шкідливості і небезпеки праці.

3) Розробка та планування заходів щодо поліпшення умов праці.

4) Організація безпечного виконання робіт працівниками.

5) Впровадження технічних засобів і заходів щодо боротьби з травматизмом і профзахворюваннями.

6) Розробка методів оцінки соціальної та економічної ефективності, заходів з удосконалення умов і охорони праці.

Правовою основою законодавства щодо охорони праці є Конституція Україн, Закони України: «Про охорону праці», «Про охорону здоров'я», «Про пожежну безпеку», «Про використання ядерної енергії та радіаційний захист», «Про забезпечення санітарного та епідеміологічного благополуччя населення», а також Кодекс законів про працю України.

Основоположним законодавчим документом в галузі охорони праці є Закон України «Про охорону праці», дія якого поширюється на всі підприємства, установи і організації незалежно від форм власності та видів їх діяльності, на усіх громадян, які працюють, а також залучені до праці на цих підприємствах.

5.1 Аналіз шкідливих та небезпечних факторів в офісі фірми

Шкідливі виробничі фактори – фактори середовища і трудового процесу, які можуть викликати професійну патологію, тимчасове або стійке зниження працездатності, підвищити частоту соматичних та інфекційних захворювань, призвести до порушення здоров'я потомства.

При роботі за комп’ютерною технікою працівник може зазнавати багатьох видів шкідливих та небезпечних виробничих факторів. Умовно їх можна поділити на фактори виробничого середовища та психофізіологічні фактори.

До факторів виробничого середовища можна віднести:

– незадовільний мікроклімат (температура, вологість, вентиляція повітря, інфрачервоне або ультрафіолетове випромінювання) в приміщенні;

– барометричний тиск;

– постійні електричні поля і випромінювання;

– небезпечне іонізуюче випромінювання;

– високий рівень промислових шумів та вібрацій (місцевий або загальний);

– недостатнє природнє або технічне освітлення в робочих приміщеннях.

До психофізіологічних факторів відноситься:

– емоційна напруга (обумовлена, наприклад, надмірним навантаженням на центральну нервову систему, органи чуттів);

– динамічні й статичні перевантаження;

– вимушене положення тіла при виконанні різноманітних виробничих операцій; надмірний і тривалий тиск різних предметів на кінцівки та інші частини тіла,

– перевантаження окремих систем організму; недостатня рухова активність; надмірно швидкий темп роботи.

Робота на персональному комп'ютері супроводжується постійною і значною напругою функцій зорового аналізатора. Розлад органів зору різко збільшується при роботі понад чотири годин на день .

Нервово-емоційне напруження при роботі на ПК виникає внаслідок дефіциту часу, великого об'єму і щільності інформації, особливостей діалогового режиму спілкування людини і ПК, відповідальності за безпомилковість інформації. Тривала робота з дисплеєм, особливо в діалоговому режимі, може призвести до нервово-емоційного перенапруження, порушення сну, погіршення стану, зниження концентрації уваги та працездатності, хронічного головного болю, підвищеної збудливості нервової системи, депресії.

Підвищені статичні і динамічні навантаження у користувачів ПК призводять до скарг на болі в спині, шийному відділі хребта і руках. З усіх нездужань, обумовлених роботою на комп'ютерах, частіше зустрічаються ті, які пов'язані з використанням клавіатури. У період виконання операцій введення даних кількість дрібних стереотипних рухів кистей і пальців рук за зміну може перевищити 60 тис., що відповідно до гігієнічної класифікації праці відноситься до категорії шкідливих і небезпечних. Більшість працюючих рано чи пізно починають пред'являти скарги на болі в шиї і спині. Ці нездужання накопичуються поступово і отримали назву «синдром тривалих статичних навантажень» (СТСН).

Іншою причиною виникнення СТСН може бути тривале перебування в положенні «сидячи», яке призводить до сильного перенапруження м'язів спини і ніг. Основною причиною перенапруги м'язів спини і ніг є нераціональна висота робочої поверхні столу і сидіння, відсутність опорної спинки і підлокітників, незручне розміщення монітора, клавіатури та документів, відсутність підставки для ніг.

Ще одним небезпечним фактором може стати недостатність освітлення, що приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. неправильний напрям світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть привести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Тривала і інтенсивна робота на комп'ютері може стати джерелом важких професійних та звичайних захворювань таких як:

– **Неправильна постава** – це призводить до подальшого розвитку викривлення хребта сколіозу, лордозу, кіфозу, а як результат головні болі, болі в області шиї і всього хребта, болі в області таза. Неправильно положення ніг може привести до артриту (запалення суглобів), артрозу (деформації).

– **Тунельний синдром** – найвідоміше захворювання людей, що працюють за комп'ютером, або ж синдром зап'ястного каналу. Тунельний синдром проявляється після кількох годин напруженої роботи за комп'ютером.

– **Геморой** – хвороба аналогічна варикозному розширення вен. Причиною хвороби може бути застій крові у венах.

– **Розвиток короткозорості** – через те, що екран монітора за контрастом вище, ніж навколишні об'єкти, розвивається короткозорість.

– **Порушення фокусування** – наслідком напруженої роботи за монітором є порушення фокусування, яке може бути викликане перенапруженням очних м'язів.

– **Сухість очей** – при погляді на джерело світла око починає менше моргати через рефлекс. Виникає «обсушування» рогівки ока, яке призводить до болі в очах.

5.2 Заходи щодо зменшення впливу шкідливих факторів роботи за персональним комп’ютером

Працюючи за комп’ютером, рекомендується дотримуватися правил тривалості роботи, правильної постави, розміру шрифтів та зображень, вимог до приміщення тощо.

Принципи правильної роботи за комп’ютером:

– у робочому приміщенні (кімнаті), де встановлені комп’ютери, щодня потрібно виконувати вологе прибирання;

– приміщення, у якому знаходяться комп’ютери, потрібно провітрювати щогодини;

– після кожного часу роботи рекомендується робити десяти хвилинну перерву, яку зручно суміщати з провітрюванням. За будь-яких умов безперервна робота за комп’ютером для дорослої людини не повинна перевищувати двох годин. Під час перерви не варто читати або дивитися телевізор;

– необхідно постійно слідкувати за станом екрану монітора: він має бути чистим, без плям та пилу;

– потрібно слідкувати за поставою: ноги повинні твердо стояти на підлозі чи на спеціальній підставці; стегна повинні бути розташовані під прямим кутом до тулуба, а гомілки – під прямим кутом до стегон; сидіти потрібно прямо або злегка нахилившись вперед; відстань від очей до екрану монітора – не менше 55-60 см; центр екрану має знаходитися на рівні очей чи трохи нижче; рекомендується хоча б раз на день виконувати гімнастику для очей;

– щоб попередити „синдром сухого ока”, потрібно моргати кожні 3-5 секунд;

– не використовувати звичайний телевізор замість монітору;

– у процесі роботи рекомендується періодично (приблизно раз на 20-30 хвилин) переводити погляд з екрану на найбільш віддалений предмет у кімнаті, а ще краще – на віддалений об’єкт за вікном;

– якщо з’явилося відчуття втоми, напруження, сонливості, тяжкості в очах, потрібно припинити роботу та хоча б трохи відпочити.

Рекомендовані умови для роботи за комп'ютером:

– твердий стілець; край стільця не повинен тиснути на судини під колінами;

– відстань до монітора повинна бути 50-70см;

– перерву в сидячій роботі, 15-20 хвилин кожні 1-2 години;

– правильне освітлення робочого місця. при слабкому світлі очі напружуються і болять;

– потрібно закривати очі для відпочинку. час від часу відводите очі вбік, щоб дати відпочити своєму зору.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Навколишнє середовище – система пов’язаних між собою природних та антропогенних явищ та об’єктів, вплив та використання яких дозволяють працювати, вести побут, відпочивати, тощо. Навколишнє середовище включає в себе соціальні, природні, штучно створені (фізичні, хімічні, біологічні) фактори. Важливим викликом сьогодення є необхідність зберегти природу та запобігти негативного впливу господарської діяльності людини на природнє середовище, адже економічна діяльність в будь-якому прояві викликає забруднення навколишнього середовища.

Екологія – наука, що вивчає взаємовідносини між живими організмами та сферою їх перебування. Збільшення та розвиток виробництва стрімко посилює негативний вплив на навколишнє середовище, адже все більша кількість різних речовин, які часто є токсичними та не притаманними середовищу, потрапляють в нього, зумовлюючи негативні наслідки.

Джерела забруднення навколишнього середовища можуть досить легко переміщуватись на великі відстані. У атмосфері небезпечні речовини розповсюджуються повітряними течіями. Інтенсивність такого розповсюдження залежить від напряму переміщення повітряних мас, погодних умов, тощо. Потрапивши у воду, небезпечні речовини переміщуються від одної водойми до іншої, поширюючи свій вплив на флору та фауну.

Охорону навколишнього середовища можна організувати різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними, тощо.

Загалом, охорона навколишнього середовища має вирішити дві основні задачі:

– Запобіжну – забезпечити чистоту екосистеми.

– Організаційну – забезпечити раціональне використання природних ресурсів.

Головною проблемою використання природних ресурсів є їх вичерпність та обмеженість, як це дуже гарно видно на прикладі корисних копалин, що накладає певні обмеження на економічний зріст. Вичерпність та обмеженість природних ресурсів викликає конкуренцію за їх видобуток, користування, тощо, так як нині природні ресурси слугують для задоволення різних потреб.

Раціональний підхід до природокористування передбачає розробку та реалізацію заходів щодо раціонального використання та відтворення природних ресурсів, відповідальну взаємодію людини та природи. Для цього необхідно:

– Виконувати оптимальний розподіл ресурсів для різних сфер господарської діяльності.

– Використовувати технології, що спрямовані на економне використання ресурсів.

– Проводити заходи для поповнення природних ресурсів.

Забезпечення людства чистим повітрям, водними ресурсами, чистими ґрунтовими покровами для можливості забезпечення продуктами харчування є важливим питанням, що пов’язане з охороною природи, та дозволить підтримувати високий рівень життя населення довгі роки.

6.1 Забруднення навколишнього середовища

В офісі можна виділити наступні джерела забруднення:

– Твердопаливні котли.

– Дизельні електростанції.

– Побутове сміття.

– Електромагнітне випромінювання.

Зазвичай у офісах знаходиться велика кількість одиниць комп’ютерної техніки, стаціонарних телефонів, серверів, оргтехніки, тощо. Вплив планового чи аварійного відключення електроенергії призводить до зупинки робочого процесу, тому існує практика щодо розміщення електростанцій на території підприємств,

зокрема дизельних електростанцій. Така станція може працювати як в штатному режимі, так і в аварійному.

Як результат роботи дизельної електростанції отримуємо продукти згоряння дизельного палива та шумове забруднення.

В холодну пору року для обігріву та опалення приміщень використовують твердопаливні котли. Для опалення використовують вугілля, мазут, дерев'яні пелети. Пелети хоч мають менше шкідливих викидів в атмосферу, ніж вугілля чи мазут, але все одно не можуть вважатись екологічно чистими.

В таблиці 6.1 наведено порівняльну характеристику згоряння різного палива.

Таблиця 6.1 – Рівні викидів забруднюючих речовин в атмосферу при спалюванні різних видів палива

Вид палива	Викиди забруднюючих речовин в атмосферне повітря без систем очищення, тонн на 1 тис. тонн нат. палива				
	CO2	NO2	SO2	Тверді частинки(пил неорг.)	РАЗОМ
Природний газ	1,18	3,52	0,00	0,00	4,70
Древні брикети, пелети	**4,6**8	**9,3**1	**0,2**8	**4,11**	**17,69**
Деревина дров'яна	4,9	9,4	0,3	4,3	18,9
Тирса деревна	5,0	9,6	0,5	5,0	20,0
Деревні відходи, обрізки	5,2	9,9	0,4	5,2	20,7
Швидкозростаюча деревина	4,8	9,5	0,0	8,4	22,7
Тріска, сучки, кора	5,6	11,4	0,8	13,4	31,3
Мазут	5,20	5,20	35,30	0,30	45,90
Брикет торф'яний	8,04	26,81	3,00	13,02	50,87
Кам'яне вугілля	9,58	63,56	9,20	65,32	147,66

Побутові відходи – це залишки речовин і предметів, які утворюються в результаті побутової та господарської діяльності людини і які не можуть бути використані на місці утворення, а їх накопичення і зберігання порушують санітарний стан навколишнього середовища.

Побутові відходи можуть бути:

– Рідкі – нечистоти, помиї, стічні води

– Тверді – сміття (побутові відходи), кухонні відходи, зокрема папір, картон, скло, пластмаса, продукти харчування.

Серед основних генераторів електромагнітного випромінювання можна виділити комп'ютерну техніку. Більша частина техніки працює протягом робочого дня, тобто, порядка 8 годин, частина не вимикається взагалі (сервери, тощо). В більшості своїй, люди стикаються з випромінювання від комп'ютерних процесорів та моніторів. Також джерелом електромагнітного випромінювання є мікрохвильові печі, яка можуть бути на кухнях в офісі. Серед негативних наслідків впливу електромагнітного випромінювання на організм людини можна виділити головний біль, порушення сну, перевтому.

6.2 Розробка заходів щодо зменшення забруднення

Повністю виключити забруднення навколишнього середовища вкрай складно, тому необхідно виробити заходи щодо зменшення такого впливу. Можна виокремити наступний перелік заходів:

– Використовувати природний газ для опалення та обігріву приміщень. Такими чином вдасться зменшити вплив шкідливих викидів, адже порівняння, проведене у таблиці 6.1 показало, що для природного газу показники є найнижчими. Але слід зважати на те, що використання природного газу може не бути оптимальним з точки зору економічної ефективності.

– Раціоналізувати процес позбавлення від побутових відходів. Відправляти вторинну сировину на переробку, а деякі побутові відходи, що передбачають безпечну утилізацію шляхом спалювання, – на спалення для отримання електроенергії. Зменшити кількість пластмас, що використовуються для упаковки на користь такої тари та упаковки, що передбачає повторне використання.

– Зменшити рівень електромагнітного впливу та убезпечити себе від такого впливу. Цього можна досягти за рахунок своєчасного вимикання не потрібних для

подальшої роботи електроприладів та використання перерв під час роботи з комп’ютером та іншими електроприладами.

ВИСНОВКИ

За результатами кваліфікаційної роботи було підвищено достовірність оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, за рахунок побудови нелінійної регресійної моделі.

В процесі роботи було досягнуто наступних результатів:

– Досліджено існуючі регресійні моделі для оцінювання розміру програмного забезпечення для мов програмування Java, PHP, фреймворків мови PHP. У знайдених роботах використовуються різні набори метрик для побудови моделей. Моделей саме для мови JavaScript та її фреймворків для серверної частини не було знайдено.

– Проведено пошук проектів, які було створені з використанням фреймворку NestJS. Пошук проводився серед проектів з відкритим кодом на сайті GitHub. Для збору метрик використано платформу з відкритим кодом SonarQube.

– Проведено перевірку вихідних даних на викиди. Перевірка показала негаусівський розподіл даних, тому для усунення викидів було застосовано метод, що заснований на нормалізуючих перетвореннях та розрахунку квадрату відстані Махаланобіса.

– Побудовано двохфакторну нелінійну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS. Отримана нелінійна модель має високі оцінки якості R^2, $MMRE$ та $PRED(0{,}25)$.

– Побудовано лінійну двохфакторну регресійну модель для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, з метою її порівняння з нелінійної моделлю. Отримана модель теж має високі оцінки якості з дещо меншим показником $PRED(0{,}25)$ у порівнянні з нелінійною регресійною моделлю.

– Проведено розрахунок довірчих інтервалів та інтервалів передбачення лінійної та нелінійної регресійних моделей. Отримані результати показують, що

для 64% даних отримано менші ширини як довірчих інтервалів, так й інтервалів передбачення.

– Побудовано нелінійну однофакторну регресійну модель, в якій предиктором виступає кількість класів. Цю модель порівняно з нелінійною однофакторною моделлю для мови програмування Java. Порівняння показало погані характеристики якості при застосуванні моделі мови Java, що підтверджує необхідність будувати моделі під конкретну мову програмування чи фреймворк.

– Виконано розробку проекту програмного забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS. Розроблений застосунок пройшов тестування успішно та повністю відповідає висунутим вимогам.

– Виконано розрахунок економічної ефективності від розробки та впровадження програмного забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS. Розрахунки засвідчили доцільність розробки та використання програми.

– Розроблено розділи з охорони праці та охорони навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Trendowicz A., Jeffery R. Software project effort estimation. foundations and best practice guidelines for success. Springer International Publishing, 2014. 491 p. DOI: https://doi.org/10.1007/978-3-319-03629-8.

2. Boehm B. W. Software engineering economics. Englewood Cliffs, NJ: Prentice Hall. 1981. 768 p.

3. H. B. K. Tan, Y. Zhao, H. Zhang. Estimating LOC for information systems from their conceptual data models. *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. May 20-28, 2006. Shanghai, China, 2016. P. 321–330.

4. H. B. K. Tan, Y. Zhao, H. Zhang. Conceptual data model-based software size estimation for information systems. *Transactions on Software Engineering and Methodology.* 2009. Vol. 19, Issue 2, Article No. 4.

5. Приходько С. Б., Приходько А. С., Шутко І. С. Нелінійні регресійні моделі для оцінювання кількості строк коду web застосунків, що створюються за допомогою PHP-фреймворків. *Інформаційні технології: моделі, алгоритми, системи* : зб. тез II всеукраїнської науково-практичної інтернет конференції, 26-28 жовтня. 2021 р. С 11–12.

6. Ворона М. В., Приходько А. С., Шутко І. С. Трьохфакторні нелінійні регресійні моделі для оцінювання розміру web-застосунків, що створюються з використанням php фреймворків. *Інформатика, інформаційні системи та технології* : зб. тез доповідей вісімнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 23 квітня 2021 р. Одеса, 2021. С. 111–113.

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. *Інформаційні технології та комп'ютерна інженерія.* 2021. Т. 50, № 1. С. 115–121.

8. Приходько С. Б., Приходько Н. В., Фаріонова Т. А., Ворона М. В. Трьохфакторна нелінійна регресійна модель для оцінювання розміру Php застосунків з відкритим кодом. *Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки*. 2020. № 4 (56). С. 124–131.

9. Vorona M. V. Comparison o f the evaluation results of the size of open source php-based apps by nonlinear regression models. *German International Journal of Modern Science*. 2021. №6. P 43–47.

10. Латанська Л. О., Бринзей О. А. НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ СЕРВЕРНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NESTJS. *Інформаційні технології: моделі, алгоритми, системи* : зб. тез III всеукраїнської науково-практичної інтернет конференції, 26-28 жовтня. 2022 р. С 8–10.

11. 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.* 2019. №2 (69). С. 145–153.

12. Montgomery D.C., Peck E. A., Vining G. G. Introduction to Linear Regression Analysis : Wiley, 2021. 702 p.

13. Prykhodko S. B., Prykhodko N. V. Constructing the non-linear regression models on the basis of multivariate normalizing transformations. *Electronic modeling*. 2018. Т.40. № 6. С. 99–108.

14. 15 компаній, які успішно використовують Node.js у 2021 році [Електронний ресурс] // Режим доступу : URL: https://www.trio.dev/blog/companies-use-node-js (дата звернення: 10.09.2022).

15. Показники використання серверних фреймворків та зацікавленості серед зацікавлених осіб [Електронний ресурс] // Режим доступу : URL: https://2020.stateofjs.com/en-US/technologies/back-end-frameworks/ (дата звернення: 10.09.2022).

16. Найпопулярніші бекенд-фреймворки – 2012/2022_[Електронний ресурс]. // Режим доступу : URL: https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2022/ (дата звернення: 10.09.2022).

17. Frost J. Regression Analysis: An intuitive guide for using and interpreting linear models.by Example. Statistics by Jim Publishing, 2020. 355 p.

18. Prykhodko S. B., Prykhodko N. V., Makarova L. M., Pugachenko K. M. Detecting Outliers in Multivariate Non-Gaussian Data on the basis of Normalizing Transformations. *Proceedings of the 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* : «Celebrating 25 Years of IEEE Ukraine Section». Kyiv, Ukraine, May 29-June 2. Kyiv, 2017. P. 846–849.

19. Prykhodko S. B., Prykhodko N. V., Makarova L. M. Pukhalevych A. V. 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, Ukraine, February 20–24. Lviv-Slavske, 2018. P. 962–965.

20. Arkes J. Regression Analysis: A practical introduction. Taylor & Francis Group. 2019. 342 p.

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

22. Prykhodko S. B., Prykhodko N. V. Mathematical Modeling of Non-Gaussian Dependent Random Variables by Nonlinear Regression Models Based on the Multivariate Normalizing Transformations. Mathematical Modeling and Simulation of Systems (MODS'2020). MODS 2020. Advances in Intelligent Systems and Computing. 2020. Vol. 1265. P. 166–174.

23. Foss T. A, Stensrud E., Kitchenham B., Myrtveit I. Simulation study of the model evaluation criterion MMRE. *IEEE Transactions on software engineering*. 2003. Vol. 11 (29). P. 985–995.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Найменування програми: «NestJS KLOC estimation».

1 Підстави для розробки

Підставою для розробки є наказ на затвердження тем кваліфікаційних робіт магістрів.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програми є надання користувачу можливості оцінити розмір серверних застосунків, що створюються з використанням фреймворку NestJS.

2.2 Експлуатаційне призначення

Програма призначена для експлуатації на персональних комп’ютерах користувачів для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

3 Вимоги до програми або програмного продукту

3.1 Вимоги для функціональних характеристик

3.1.1 Вимоги до переліку виконуваних функцій

Розроблена програма має надавати можливість виконувати наступні функції:

– Оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

– Оцінювання інтервалу передбачення значення розміру серверних

застосунків, що створюються з використанням фреймворку NestJS.

3.1.2 Вимоги для організації вхідних та вихідних даних

Введення вхідних даних відбувається за рахунок заповнення полів графічного інтерфейсу користувача.

Результати виводяться у відповідні поля графічного інтерфейсу користувача.

Вхідними даними для програми є наступні: кількість класів, кількість функцій, довірча ймовірність для проведення оцінювання

Кількість класів та кількість функцій серверного застосунку – це натуральні числа, довірча ймовірність – дійсне число від 0 до 1.

Вихідними даними є отримана оцінка розміру серверного застосунку у тисячах рядків коду, оцінки довірчого інтервалу та інтервалу передбачення, які буде відображено у відповідних полях графічного інтерфейсу програми.

3.2 Вимоги до надійності

Надійне (стійке) функціонування програмного забезпечення повинно бути забезпечено виконанням сукупності організаційно-технічних заходів, виконанням валідації вхідних даних згідно вимог та відображення повідомлень про виявлення некоректних даних з закликом ввести коректні.

3.3 Умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.4 Вимоги до технічної та програмної сумісності

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітний процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

3.5 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм не висуваються.

3.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки не висуваються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4 Вимоги до програмної документації

Склад програмної документації має включати в себе:

1) Технічне завдання;
2) Програму та методику випробувань;
3) Інструкцію користувача;
4) Опис програмного забезпечення;
5) Текст програми.

5 Техніко-економічні показники

Економічна ефективність від впровадження програмного забезпечення була розрахована у розділі 4. Виходячи з одержаних результатів, впровадження даного програмного забезпечення є доцільним, приблизний термін окупності – 4 місяці.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи виконання розробки програмного забезпечення можна представити наступним чином:

Таблиця А.1 – Стадії та етапи розробки

Стадії розробки	Назва етапу	Термін виконання
Технічне завдання	Розробка технічного завдання	30.09.2022
	Затвердження технічного завдання	10.10.2022
Ескізний проект	Розробка ескізного проекту	25.10.2022
	Затвердження ескізного проекту	01.11.2022
Технічний проект	Розробка технічного проекту	15.11.2022
	Затвердження технічного проекту	30.11.2022
Робочий проект	Розробка програми	01.12.2022
	Розробка програмної документації	14.12.2022
	Випробування програми	20.12.2022

7 Порядок контролю та прийомки

Приймально-здавальні роботи мають проводитись під наглядом Замовника або уповноваженої ним особи.

Приймально-здавальні випробування повинні проводитись згідно документу «Програма та методика випробувань», який узгоджується між стороною розробника та стороною замовника.

Хід проведення приймально-здавальних робіт Замовник та Виконавець документують в «Протокол проведення випробувань».

На основі Протоколу проведення випробувань Виконавець та Замовник підписують «Акт прийому-здачі програми у експлуатацію».

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом дослідження є програмне забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

2 Мета випробувань

Мета випробувань – перевірка працездатності програмного застосунку, перевірка відповідності фактичних характеристик програмного застосунку тим вимогам, які було викладено у технічному завданні.

3 Вимоги до застосунку

Під час виконання випробувань перевіряються функціональні характеристики (вимоги), що були викладені в технічному завданні.

Розроблена програма має надавати можливість виконувати наступні функції:

– Оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

– Оцінювання інтервалу передбачення значення розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

4 Вимоги до програмної документації

Програмна документація включає в себе наступні документи:

1) технічне завдання;

2) програму та методику випробувань;

3) інструкцію користувача;

4) опис програмного забезпечення.

5) текст програми

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувались під час випробувань:

– Lenovo IdeaPad 3 15ALC6.

– AMD Ryzen 5 5500U (2.1 - 4.0 ГГц).

– Оперативна пам’ять: 16 Гбайт.

– Накопичувач даних: SSD 512 Гбайт.

5.2 Програмні засоби, що використовувались під час випробувань

Операційна система Windows 10 Pro 21H2 19042.2006

5.3 Порядок проведення випробувань

Порядок проведення випробувань – перевірка відповідності функціональних характеристик програмного застосунку, перевірка вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

За відсутності конкретних вимог до оформлення та подання програмної документації, розроблені програмні документи перевіряються представником замовника візуально. Уповноважена особа перевіряє відповідність розроблених програмних документів з тими, що були вказані у пункті «Склад програмної документації, що пропонується на випробування».

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

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

Перевірку можна вважати успішно завершеною у випадку відповідності виконуваних функції технічному завданню.

Порядок випробувань:

– Перевірка роботи застосунку на моніторах з різною розподільною здатністю.

– Перевірка відображення шрифтів.

– Перевірка кожної форми графічного інтерфейсу на можливість введення даних.

– Перевірка кожної форми графічного інтерфейсу на введення невалідних даних.

– Перевірка можливості виконання оцінювання розміру, визначення довірчого інтервалу та інтервалу передбачення для серверних застосунків, що створюються з використанням фреймворку NestJS.

7 Результати випробувань

За результатами випробувань було підтверджено відповідність функціональних можливостей висунутим вимогам до програмного забезпечення.

8 Відомості про відмови, збої та аварійні ситуації

Під час проведення випробувань відмов, збоїв та аварійних ситуацій помічено не було.

9 Відомості про коригування параметрів об’єкта випробувань та технічної документації

Коригування параметрів об’єкта випробувань та технічної документації в процесі виконання випробувань не проводилось.

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА

Вступ

Інструкція користувача призначена для ознайомлення користувача з правилами та порядком виконання операцій, які забезпечуються роботою програмного забезпечення.

Ця програма призначена для оцінювання розміру та визначення інтервальних оцінок розміру у тисячах строк коду серверних застосунків, що створюються з використанням фреймворку NestJS. Програма є вільним програмним забезпеченням з графічним інтерфейсом користувача з можливістю вводити дані та переглядати результати.

Загальні відомості про програму

Найменування програми: «NestJS LOC estimation».

Використані мови програмування

Програма написана на мові Python, для розробки графічного інтерфейсу користувача було використано сумісну з Python бібліотеку Qt.

Призначення програми

Програма призначена для оцінювання розміру (в тисячах рядків коду) серверних застосунків, що створюються з використанням фреймворку NestJS на основі таких метрик як кількість класів та кількість функцій.

Функціональні можливості програми

– Оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру серверного застосунку, що створюється з використанням фреймворку NestJS.

– Оцінювання інтервалу передбачення значення розміру серверного застосунку, що створюється з використанням фреймворку NestJS.

Обмеження області застосування програми

Область застосування програми обмежена галуззю розробки програмного

забезпечення.

Умови, необхідні для використання програми

Спеціальні умови для використання програми відсутні

Вимоги до технічної сумісності та програмного середовища

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітній процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

Загальний вигляд стартового вікна програми

Загальний вигляд стартового вікна програми «NestJS LOC estimation» наведено на рисунку В.1.

Стартове вікно містить наступний перелік полів для введення даних:

– Поле «Count of classes» дозволяє вказати кількість класів оцінюваного серверного застосунку. В якості кількості класів можна ввести натуральне число.

– Поле «Count of functions» дозволяє вказати кількість методів в оцінюваному серверному застосунку. В кількості функцій можна ввести натуральне число.

– Поле «Confidence level» дозволяє вказати довірчу ймовірність для виконання розрахунків для оцінювання розміру, довірчого інтервалу та інтервалу передбачення. В якості довірчої ймовірності можна ввести дійсне число від 0 до 1.

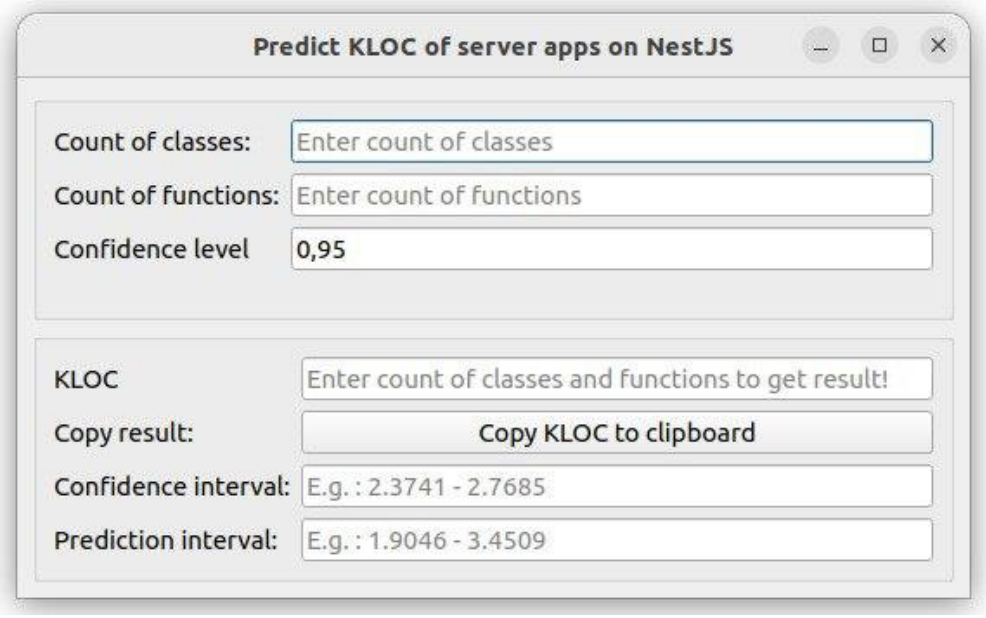

Рисунок В.1 – Вигляд вікна програми до виконання оцінювання

Результати оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS

Після введення вхідних даних програмне забезпечення автоматично виконає розрахунок відповідних оцінок. Результати оцінювання розміру серверного застосунку (його розмір у тисячах строк коду, довірчі інтервали та інтервали прогнозування нелінійної регресії), будуть показані в полях в нижній частині екрану.

Результат виконання оцінювання можна побачити на рисунку В.2.

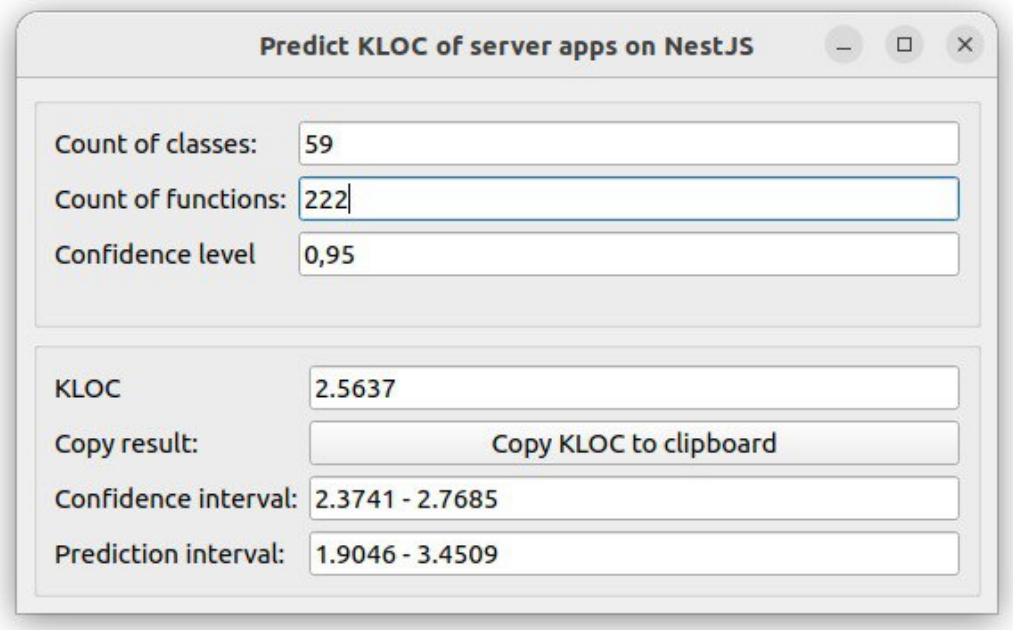

Рисунок В.2 – Результат оцінювання серверного застосунку, що створюється з використанням фреймворку NestJS

ДОДАТОК Г – ТЕКСТ ПРОГРАМИ

```
# General libraries
import sys

# Libraries for calculations
import math
import numpy as np
from scipy import stats

# User interface libraries
from PySide2 import QtWidgets, QtGui


class KLOCPredictor:
    N = 45
    k = 2
    V = N - k - 1

    B = [
        -1.99787698508896,
        0.187113121751769,
        0.884520277756632
    ]

    SZY = 0.061780775235153

    # Inverse covariance matrix
    ICM = np.array([
        [1.13885728885, -0.72182851705254],
        [-0.72182851705254, 0.731137536286726]
    ])

    ZXMeans = [1.43484, 1.97255]

    def predictKLOC(self, X):
        return (10 ** self.B[0]) * (X[0] ** self.B[1]) * (X[1] ** self.B[2])

    def getSEZ(self, X):
        zx = np.log10(X) - self.ZXMeans
        return np.matmul(np.matmul(zx.T, self.ICM), zx)

    def calcPredInterval(self, value, X, alpha):
        Y = math.log10(value)
        sez = self.getSEZ(X)
        predHalfIntervalValue = (stats.t.ppf(1 - alpha / 2, self.V) * self.SZY *
math.sqrt((1 + 1.0 / self.N + sez)))
        return 10 ** (Y - predHalfIntervalValue), 10 ** (Y +
predHalfIntervalValue)

    def calcConfInterval(self, value, X, alpha):
        Y = math.log10(value)
        sez = self.getSEZ(X)
        confHalfIntervalValue = (stats.t.ppf(1 - alpha / 2, self.V) * self.SZY *
math.sqrt((1.0 / self.N + sez)))
        return 10 ** (Y - confHalfIntervalValue), 10 ** (Y +
confHalfIntervalValue)
```

```
class AppMainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()

        self.KLOCPredictor = KLOCPredictor()

        # Input for count of classes
        self.UIClassesCountInput = QtWidgets.QLineEdit("", self)
        self.UIClassesCountInput.setValidator(QtGui.QIntValidator(self))
        self.UIClassesCountInput.setPlaceholderText("Enter count of classes")
        self.UIClassesCountInput.textChanged.connect(self.execCalculations)

        # Input for count of functions
        self.UIFunctionsCountInput = QtWidgets.QLineEdit("", self)
        self.UIFunctionsCountInput.setValidator(QtGui.QIntValidator(self))
        self.UIFunctionsCountInput.setPlaceholderText("Enter count of functions")
        self.UIFunctionsCountInput.textChanged.connect(self.execCalculations)

        # Input for value of confidence
        self.UIConfidenceInput = QtWidgets.QLineEdit("0,95", self)
        self.UIConfidenceInput.setValidator(QtGui.QDoubleValidator(0.0, 1.0, 2,
parent=self))
        self.UIConfidenceInput.setPlaceholderText("E.g. : 0,95")
        self.UIConfidenceInput.textChanged.connect(self.execCalculations)

        # Output for result predicted KLOC
        self.UIKLOCInput = QtWidgets.QLineEdit(self)
        self.UIKLOCInput.setReadOnly(True)
        self.UIKLOCInput.setMinimumWidth(150)
        self.UIKLOCInput.setPlaceholderText("Enter count of classes and functions
to get result!")

        # Button for copy the calculated value of KLOK to the clipboard
        self.UICopyToClipboardBtn = QtWidgets.QPushButton(self)
        self.UICopyToClipboardBtn.setText("Copy KLOC to clipboard")
        self.UICopyToClipboardBtn.clicked.connect(self.copyToClipboard)

        # Output for result predicted confidence interval
        self.UIConfidenceIntervalInput = QtWidgets.QLineEdit(self)
        self.UIConfidenceIntervalInput.setReadOnly(True)
        self.UIConfidenceIntervalInput.setMinimumWidth(150)
        self.UIConfidenceIntervalInput.setPlaceholderText("E.g. : 2.3741 -
2.7685")

        # Output for result predicted prediction interval
        self.UIPredictionIntervalInput = QtWidgets.QLineEdit(self)
        self.UIPredictionIntervalInput.setReadOnly(True)
        self.UIPredictionIntervalInput.setMinimumWidth(150)
        self.UIPredictionIntervalInput.setPlaceholderText("E.g. : 1.9046 -
3.4509")

        UIDataInputGroup = QtWidgets.QGroupBox(self)
        UIDataOutputGroup = QtWidgets.QGroupBox(self)

        UIMainWidget = QtWidgets.QWidget()
        self.setCentralWidget(UIMainWidget)

        UIMainLayout = QtWidgets.QVBoxLayout(UIMainWidget)
        UIMainLayout.addWidget(UIDataInputGroup)
        UIMainLayout.addWidget(UIDataOutputGroup)

        # Add UI items to data input group layout
        UIDataInputGroupLayout = QtWidgets.QFormLayout(UIDataInputGroup)
```

```
        UIDataInputGroupLayout.addRow("Count of classes:",
self.UIClassesCountInput)
        UIDataInputGroupLayout.addRow("Count of functions:",
self.UIFunctionsCountInput)
        UIDataInputGroupLayout.addRow("Confidence level", self.UIConfidenceInput)

        # Add UI items to data output group layout
        UIDataOutputGroupLayout = QtWidgets.QFormLayout(UIDataOutputGroup)
        UIDataOutputGroupLayout.addRow("KLOC", self.UIKLOCInput)
        UIDataOutputGroupLayout.addRow("Copy result:", self.UICopyToClipboardBtn)
        UIDataOutputGroupLayout.addRow("Confidence interval:",
self.UIConfidenceIntervalInput)
        UIDataOutputGroupLayout.addRow("Prediction interval:",
self.UIPredictionIntervalInput)

    def copyToClipboard(self):
        QtGui.QClipboard().setText(self.UIKLOCInput.text())

    def getAlpha(self):
        return 1 - float(self.UIConfidenceInput.text().replace(",", "."))

    def execCalculations(self):
        X = []
        for edit in [self.UIClassesCountInput, self.UIFunctionsCountInput]:
            if edit.text() and edit.hasAcceptableInput():
                potentialNumber = edit.text().replace(",", ".")
                X.append(float(potentialNumber))
            else:
                return

        alpha = self.getAlpha()
        value = self.KLOCPredictor.predictKLOC(X)
        predIntervalValues = self.KLOCPredictor.calcPredInterval(value, X, alpha)
        confIntervalValues = self.KLOCPredictor.calcConfInterval(value, X, alpha)

        self.UIKLOCInput.setText('%.4f' % value)
        self.UIPredictionIntervalInput.setText('%.4f - %.4f' % predIntervalValues)
        self.UIConfidenceIntervalInput.setText('%.4f - %.4f' % confIntervalValues)


if __name__ == "__main__":
    app = QtWidgets.QApplication([])
    widget = AppMainWindow()
    widget.setWindowTitle("Predict KLOC of server apps on NestJS")
    widget.resize(550, 300)
    widget.show()
    sys.exit(app.exec_())
```

ДОДАТОК Д – ОПИС ПРОГРАМИ

1 Загальні відомості

Найменування: «NestJS LOC estimation».

1.2 Програмне забезпечення, необхідне для функціонування програми

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

1.3 Мови програмування

Програмне забезпечення для оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS, розроблене з використанням мови програмування Python та використанням бібліотек NumPy та SciPy для виконання математичних обчислень. Для створення графічного інтерфейсу користувача було використано сумісну з Python бібліотеку Qt.

2 Функціональне призначення

Функціональним призначенням програми є надання користувачу можливості оцінити розмір серверних застосунків, що створюються з використанням фреймворку NestJS.

Функції програмного застосунку, які доступні користувачу:

– Оцінювання розміру серверних застосунків, що створюються з використанням фреймворку NestJS у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

– Оцінювання інтервалу передбачення значення розміру серверних застосунків, що створюються з використанням фреймворку NestJS.

2.1 Функціональні обмеження

Програмне забезпечення не вимагає використання зовнішніх ресурсів, тому функціональні обмеження відсутні.

3 Технічні засоби, що використовуються

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітній процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

4 Виклик та завантаження

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

Запуск програмного застосунку відбувається за рахунок натискання на ярлик програмного застосунку.

5 Вхідні дані

Вхідними даними для програми є кількість класів, кількість функцій.

Кількість класів та кількість функцій серверного застосунку – це натуральні числа, довірча ймовірність.

Довірча ймовірність вказує на відсотки та є дійсним числом від нуля до одиниці включно.

6 Вихідні дані

Вихідними даними є отримані оцінки розміру у тисячах рядків коду, оцінки довірчого інтервалу та інтервалу передбачення. Отримані значення буде відображено у відповідних полях графічного інтерфейсу програми.