МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут
комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи за темою

УДОСКОНАЛЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ КОДУ ВЕБ-ЗАСТОСУНКІВ, НАПИСАНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ DJANGO ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЇЇ РЕАЛІЗАЦІЇ

Виконав: студент 6 курсу, групи 6151м
спеціальності
121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)
Дікопольцев Іван Олександрович
(підпис, прізвище та ініціали)
Керівник МР к.т.н. Кошкін В. К.
(підпис, прізвище та ініціали)
Рецензент _____
(підпис, прізвище та ініціали)
Завідувач кафедри д.т.н, проф.Приходько С.Б.
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

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

ЗАТВЕРДЖУЮ
Завідувач кафедри

"26" 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Удосконалення математичної моделі для оцінювання розміру коду веб-застосунків, написаних з використанням фреймворку Django та розробка програмного забезпечення для її реалізації

керівник роботи: Кошкін В.К., к.т.н.; доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "26" жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи: результат оцінювання, довірчий інтервал, інтервал передбачення
4.Зміст кваліфікаційної(магістерської) роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації.)
• Огляд літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту програмного забезпечення
• Організаційно-економічний
• Розділи з охорони праці та охорони навколишнього середовища
• Висновки
• Список використаних джерел
• Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)
5. Перелік графічного матеріалу
1 Тема кваліфікаційної роботи 2 Актуальність роботи 3 Мета дослідження 4 Завдання дослідження 5 Об'єкт, предмет та методи дослідження 6 Наукова новизна 7 Практичне значення одержаних результатів 8 Апробація та публікації результатів роботи 9 Визначення метрик для оцінювання ПЗ

10 Результати власних досліджень 11 Нелінійне множинне регресійне рівняння для оцінювання розміру програмного забезпечення 12 Дослідження регресійного рівняння 13 Перевірка нормальності розподілу залишкової компоненти 14 Довірчі та прогнозовані інтервали 15-17 Реалізація ПЗ 18-19 Інтерфейс користувача 20 Висновки_____

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

Назва етапів кваліфікаційної (магістерської) роботи (МР)	Термін виконання	Примітка
1. Підготовка вступної частини МР	18.10.2019	
2. Підготовка розділу (ів) МР з огляду літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямів досліджень	19.10.2019	
3. Підготовка розділу (ів) МР з результатів власних досліджень	22.10.2019	
4. Підготовка розділу МР з проекту програмного забезпечення	16.11.2019	
5. Підготовка організаційно-економічного розділу	18.11.2019	
6. Підготовка розділу з охорони праці	20.11.2019	
7. Підготовка розділу з охорони навколишнього середовища	23.11.2019	
8. Підготовка розділу МР – Висновки	25.11.2019	
9. Оформлення списку використаних джерел	27.11.2019	
10. Оформлення додатків	30.11.2019	
11. Підготовка презентації МР та доповіді	01.12.2019	
12. Подання МР на попередній захист	01.12.2019	
13. Подання МР рецензенту	11.12.2019	
Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	14.12.2020	
Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді	18.12.2020	
Подання на кафедру ПЗАС письмового відгуку та рецензії на кваліфікаційну роботу	18.12.2020	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

Реферат до кваліфікаційної роботи
РЕФЕРАТ

Дікопольцев Іван Олександрович

«Удосконалення математичної моделі для оцінювання розміру коду веб-застосунків, написаних з використанням фреймворку Django та розробка програмного забезпечення для її реалізації»

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

Обсяг роботи. 126 стор., 22 табл., 11 рис., 46 використаних джерел, 6 додатків

Актуальність теми роботи. Підвищення достовірності оцінювання кількості рядків коду програмного забезпечення веб-застосунків в даний час є важливим завданням, яке потребує удосконалення існуючих математичних моделей, оскільки ефективність оцінювання розміру програмного забезпечення може стати точкою для успіху або ж невдачі проекту на ранньому етапі розробки.

Мета дослідження. Підвищення достовірності оцінювання розміру веб-застосунків, реалізованих за допомогою фреймворку Django.

Об’єкт дослідження. Процес оцінювання розміру веб-застосунків з відкритим кодом, реалізованих за допомогою фреймворку Django та розробка програмного забезпечення для її реалізації.

Предметом дослідження є математична модель для оцінювання розміру веб-застосунків з відкритим кодом, реалізованих за допомогою фреймворку Django.

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

Наукова новизна одержаних результатів. Отримана однофакторна нелінійна регресійна модель для оцінювання розміру веб-застосунків з відкритим кодом написаних за допомогою фреймворку Django дозволяє підвищити достовірність оцінювання розміру веб-застосунків з відкритим кодом, реалізованих за допомогою фреймворку Django, в порівнянні з існуючою моделлю.

Практичне значення одержаних результатів. Програма для оцінювання розміру веб-додатків з відкритим кодом, реалізованих за допомогою фреймворку Django, яка розроблена в рамках магістерської роботи, дозволить автоматизувати та скоротити час відповідних розрахунків.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у магістерській роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології» 30 листопада 2020 року за тематикою «Сучасні комп’ютерні системи та мережі в управлінні».

Публікації. За результатами досліджень було опубліковано 1 друковану роботу у збірниках тез конференцій.

Ключові слова. Програмне забезпечення, математичне моделювання, ефективність оцінювання, відкритий код, регресійний аналіз, теорія ймовірностей та математичної статистики, фреймворк Django.

Abstract for master's work

ABSTRACT

Dikopoltsev Ivan

«Improving the mathematical model for estimating the code size of web applications written using the Django-framework and software development for its implementation»

Master's work in obtaining an educational master's degree in specialty 121 – «Software Engineering». The Admiral Makarov National University of Shipbuilding. Mykolaiv, 2020

The scope of work. 126 pages, 22 tables, 11 figures, 46 list of references, annexes.

Relevance of the topic of work. Increasing the reliability of the size of open-source web applications implemented by using Django-framework and to develop a program for its implementation.

The aim of the study. Increasing the reliability of the size of open-source web applications implemented by using Django-framework and to develop a program for its implementation.

Object of study. The process of estimating the size of open source web applications implemented by using Django-framework.

Subject of study. Mathematical model for estimating the size of Django-based open source web applications.

Research methods. Methods of probability theory and mathematical statistics, mathematical modeling, regression analysis, object-oriented programming were applied to solve the problems.

Scientific novelty of the obtained results. Further developed a one-factor nonlinear regression model for estimating the size of Django-based open source web

applications, which made it possible to increase the reliability of Django-based applications of open-source applications, compared to the existing model.

The practical value of the results. A Django-based open source web application size estimation program designed to help you automate and reduce the time of relevant calculations.

Approbation of research results. The main provisions and results of the research presented in the master's work were tested at the III All-Ukrainian scientific-practical Internet conference of young scientists and students "Modern information systems and technologies" on November 30, 2020 on "Modern computer systems and networks in management".

Publications. According to the results of the research, 1 print job was published in the abstracts of conferences.

Keywords. Software, mathematical modeling, estimated efficiency, open source, regression analysis, probability theory and mathematical statistics, Django framework.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ВВ – випадкова величина

ІС – інформаційні системи

ІКТ – інформаційно-комунікаційні системи

ВВ – випадкові величини

ПЗ – програмне забезпечення

ER – Entity-Relationship

ISO – International System Organization

UML – Unified Modelling Language

ВСТУП

Актуальність теми. В нинішній час, галузь інформаційних технологій є однією з найбільш успішних та перспективних на ринку праці, що потребує спеціалістів відповідного рівня. В свою чергу, замовлення розробки програмного забезпечення (ПЗ) на рівні аутсорсу - це вже не просто ідея з оптимізації виконання завдань, а буденність компаній. З кожним днем кількість підприємств, яким необхідно користуватись послугами аутсорсингових компаній або розробників, невпинно зростає. Прагнення оптимізувати процеси управління ресурсами, складання звітності, проведення аудитів різних рівнів роботи компанії є основними для того причинами. І як тільки компанії, починають замислюватись про розширення, відразу вбачають необхідність автоматизації як окремих бізнес-процесів, так і роботи персоналу (співробітників) взагалі.

Розмір ПЗ є одним з найвагоміших факторів в управлінні процесом розробки ПЗ. Доведено, що розмір ПЗ корелює з витратами, зусиллями та ресурсами, необхідними на його розробку. Також інформацію, отриману у результаті оцінювання розміру ПЗ, можна використати для прогнозування зусиль розробки ПЗ за моделями COCOMO, які використовуються для надійного прогнозування різних параметрів, пов’язаних з проектом, та оцінюванням витрат. В якості «розміру» ПЗ розуміється кількість рядків вихідного коду.

В наш час існує досить поширена проблема: не існує єдиного вимірювання, набору метрик та показників для оцінювання розмірів ПЗ. Існує багато методів для оцінювання розміру програми, деякі з них базуються на кількості строк коду, інші методи обчислюють розмір з функціональних, технічних або інших аспектів. Але більшість з них стають непридатними до використання через відсутність даних, ресурсів або експертних навичок в цій

галузі.

Django вважається найкращим веб-фреймворком, написаним на мові програмування Python. Цей інструмент зручно використовувати для розробки сайтів, що працюють з базами даних. Навколо Django швидко сформувалося активна спільнота розробників. Фреймворк став стрімко розвиватися зусиллями волонтерів. Значну роль в успіху Django зіграли кілька відомих сайтів, які використовували цей фреймворк. В їх число входять Pinterest, Dropbox, Spotify, сайт The Washington Post. В даний час співтовариство Django включає більше 11 тис. розробників з 166 країн світу. Це великою мірою зумовлено тим, що на ринку програмного забезпечення найбільш затребуваними є програмне забезпечення рівня великих корпорацій, які автоматизують різні аспекти діяльності підприємств, організацій, спільнот, наприклад документообіг, взаємодія з клієнтами та ін. Вимоги до такого роду і рівня ПЗ постійно зростають, і, як наслідок, зберігається тенденція до ускладнення концепцій розробки[1].

Тому виникає потреба в підвищені достовірності оцінюванні розміру майбутнього ПЗ. Саме розмір ПЗ являє собою один з найбільш вагомих атрибутів, який використовується в різних моделях для прогнозування вартості, зусиль співробітників, ресурсів, необхідних для розробки та впровадження такого ПЗ.

Розробка веб-застосунків на мові програмування Python з використаннями фреймворку Django стає все більш актуальною темою для багатьох компаній через свою зручність та гнучкі функціональні можливості. Однак варто зауважити, що з розвитком інтернет- та цифрових технологій, робота в сфері програмування значно ускладнилася і одному розробнику часто не під силу впоратися з поставленими завданнями. Поступово, у зв’язку зі збільшенням завдань від замовників, дана сфера діяльності переходить до невеликих професійних компаній. Можна констатувати, що успіх програмних проектів на початкових стадіях залежить від рішень, які приймаються

проджект-менеджерами.

Більшість з аспектів оцінювання походять від методу аналізу функціональних точок (Funcional Points, FP). Інший підхід полягає в тому, щоб провести функціональне вимірювання, для вираження функціональності у кількості, що представляє саме поняття «розміру» ПЗ. Також існують певні методи визначення розміру програмного забезпечення, що включають оцінювання на основі варіантів використання. [2]

До прикладу, модель COCOMO (COnstructive COst MOdel) розраховує трудомісткість розробки як функцію від розміру ПЗ і безлічі «чинників вартості», що включають суб'єктивні оцінки характеристик продукту, проекту, персоналу і апаратного забезпечення. Ця модель часто використовується для надійного прогнозування різних параметрів, пов'язаних з проектом і оцінюванням витрат на розробку та впровадження. [3]

Актуальність проблеми отримання ефективної системи оцінювання кількості строк коду в даний час є важливим завданням, що вимагає удосконалення існуючих методів. Адже саме ефективність оцінки розміру програм може стати відправною точкою для успіху або невдачі проекту на ранньому етапі розробки.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання розміру програмного забезпечення, розробленого на базі фреймворка Django для веб-застосунків, та розробка програмного забезпечення для його реалізації.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати існуючі моделі оцінювання розміру ПЗ, порівняти їх;
- обґрунтувати необхідність удосконалення математичної моделі для оцінювання розміру ПЗ;
- дослідити різні джерела з відкритим вихідним кодом та визначити

веб–застосунки, реалізовані на основі фреймворку Django, які можуть бути використані для перевірки розроблюваної моделі;

- побудувати діаграми класів та отримати необхідні метрики з кожного проекту;
- перевірити вихідні емпіричні дані на наявність викидів;
- виконати нормалізацію отриманих емпіричних даних;
- побудувати лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для нормалізованих даних;
- виконати перехід від лінійної регресії до нелінійної та побудувати рівняння регресії, довірчий інтервал та інтервал прогнозування для вихідних даних;
- удосконалити математичну модель для оцінювання розміру веб–застосунків з відкритим кодом на базі фреймворку Django;
- розробити ПЗ для оцінювання розміру веб–застосунків на базі Django-фреймворку.

Об’єктом дослідження є процес оцінювання розміру веб–застосунків з відкритим кодом на базі Django-фреймворку.

Предметом дослідження є математична модель для оцінювання розміру веб –застосунків з відкритим кодом на базі Django-фреймворку.

Методи дослідження. Для вирішення поставлених завдань були використані методи теорії ймовірностей та математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування. Методом дослідження є програмне забезпечення для моделювання множинного регресійного рівняння та його оцінки, побудова довірчих та прогнозованих інтервалів.

Наукова новизна одержаних результатів полягає в удосконаленні існуючої математичної моделі для оцінювання розміру веб–застосунків на базі Django-фреймворку, за рахунок використання параметрів нормалізуючого логарифмування за натуральною основою, та нелінійне рівняння множинної

регресії, що дозволило підвищити достовірность оцінювання розміру програмного забезпечення.

Практичне значення одержаних результатів. ПЗ для оцінювання веб–застосунків з відкритим кодом на базі Django-фреймворку, що розроблено в рамках магістерської роботи, дозволило автоматизувати та скоротити час відповідних розрахунків. Впровадження отриманої математичної моделі дозволило удосконалити процес оцінювання розміру веб–застосунків з відкритим кодом на базі Django-фреймворку, за рахунок чого підвищити точність та достовірність оцінювання.

Особистий внесок здобувача. Магістерська робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у магістерській роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології» 30 листопада 2020 року за тематикою «Сучасні комп’ютерні системи та мережі в управлінні». [4]

Публікації. Основні результати магістерської роботи надіслано до публікації в збірнику для III Всеукраїнської науково-практичної інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології» 30 листопада 2020 року за тематикою «Сучасні комп’ютерні системи та мережі в управлінні».

1 АНАЛІЗ ПРОБЛЕМИ ТА СУЧАСНИХ МОДЕЛЕЙ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Оцінювання розміру проекту програмного забезпечення

Оцінювання розміру проекту на ранньому етапі розробки відіграє одну з ключових ролей у практичних завданнях з його управління, розробки та впровадження. Як правило, кількість рядків коду ПЗ є випадковою величиною, яка залежить від ряду певних факторів (в тому числі й метрик та здібностей програміста, котрий може притримуватись певного стилю кодування), що впливають на кінцевий результат. В свою чергу, метрики ПЗ кількісно визначають різні властивості програмних продуктів та процесів у вигляді чисельного відображення.

Таким чином, мета розробки полягає у виведенні одного або декількох факторів особливостей ПЗ, що дає змогу порівнювати ці значення з подібними проектами, що мають специфічні стандарти, притаманні компанії. Виходячи з отриманих результатів, можна дійти висновку щодо якості ПЗ та всього процесу розробки, а також, у разі необхідності, подальших заходів.

Існує цілий ряд алгоритмічних моделей для прогнозування витрат і собівартості, а також створення графіку робіт для програмних проектів. Принципово вони між собою не відрізняються, хоча використовують значення різних параметрів. Модель COCOMO (Constructive COst MOdel - конструктивна вартісна модель) - заснована на досвіді реалізації багатьох програмних проектів. Вона створена шляхом збору даних про велику кількість проектів і аналізу цієї інформації, в результаті чого отримані формули, що найкращим чином апроксимують наявні дані.

Залежність зростання обсягу робіт від збільшення розміру проекту розраховується за моделлю COCOMO. [3] Враховуючи те, що для різних категорій методів оцінки існують принципово різні способи та беручи до уваги фактор «розміру» програмного проекту, виникає потреба у виборі метрики його

розміру. Серед чинників, що впливають на оцінку, розмір проекту є найбільш важливим показником. Хоча оцінки розміру ПЗ недостатньо для визначення складності проекту, що розроблюється, однак існує явна залежність між розміром проекту і його трудомісткістю. [5]

Завдання оцінювання розміру програмного забезпечення (ПЗ) на ранній стадії його розробки є важливою задачею, оскільки ця інформація використовується для прогнозування трудомісткості розробки ПЗ за допомогою таких моделей як COCOMO II. [1] Що доводить необхідність розробки відповідних моделей для оцінювання розміру ПЗ, включаючи ПЗ інформаційних систем з відкритим кодом для веб–застосунків на основі фреймворку Django.

1.2 Аналіз моделей та метрик для оцінювання розміру програмного забезпечення

Основними причинами для оцінювання розміру програмного забезпечення є наступні чинники:

–визначення якості існуючого ПЗ або його розробки чи експлуатації;

–прогнозування якості ПЗ або його розробки чи експлуатації у майбутньому;

–поліпшення якості ПЗ;

–оцінка вартості майбутніх проектів;

–оцінка продуктивності застосування нових засобів і методів;

–прогноз майбутніх потреб у персоналі;

–передбачення і скорочення майбутніх потреб у технічному обслуговуванні.

Моделі оцінювання розміру ПЗ поділяються на п'ять категорій: аналогові; регресійні; моделі на основі експертних оцінок; моделі, які базуються на функціональних точках; параметричні моделі. [6]

Сьогодні існують різні методи для оцінювання розміру ПЗ. Нормалізуючі перетворення дуже часто є надійним способом побудови

нелінійних регресійних моделей та рівнянь, наведених у Розділі 2. Однак добре відомі методи їх побудови, що засновані на одновимірних нормалізуючих перетвореннях (таких як логарифмічне і Бокса-Кокса, Джонсона), не враховують кореляції між випадковими змінними у разі нормалізації багатовимірних негаусових даних. Це призводить до необхідності використання багатовимірних нормалізуючих перетворень, які враховують цю кореляцію, для побудови нелінійних регресійних моделей та рівнянь для оцінювання розміру ПЗ інформаційних систем, в тому числі з відкритим кодом на Python. [7]

В Розділі 2 побудовано нелінійне регресійне рівняння для оцінювання розміру програмного забезпечення з відкритим кодом на Python на основі натурального логарифму. Виникає потреба порівнянні моделей та у побудові відповідного рівняння для інших видів ПЗ інформаційних систем з відкритим кодом, наприклад для веб-застосунків, що написані з викристанням Django-фреймворку. [1]

Підвищення надійності оцінки часу розробки програмного забезпечення відіграє важливу роль в практичних завданнях. Як правило, час отриманий при реальних розробках є негаусівською випадковою величиною (ВВ), яка залежить від ряду факторів. Для оцінки часу розробки такого ПЗ необхідно побудувати відповідну регресійну модель [8,9] (нелінійну [10]). Підвищити надійність її оцінки можна за рахунок побудови довірчого інтервалу нелінійної регресії [11,12].

У разі використання ненормалізованих випадкових величин, побудова довірчого інтервалу нелінійної регресії без припущення про їх нормальність стає більш складною задачею. Застосування такого припущення може істотно спотворити результати, тому до існуючих даних необхідно застосовувати саме нормалізуючі перетворення.

При нормальному законі розподілу випадкової величини, довірчий інтервал лінійного рівняння регресії можливо побудувати традиційним методом

з використанням t-розподілу Стьюдента [13]. Однак для нелінійної регресії, даний метод не враховує ряд особливостей емпіричного розподілу даних (наприклад, його асиметрію).

Використання нормалізуючих перетворень зводиться до отримання лінійної регресійної моделі з вихідної нелінійної шляхом заміни змінних і коефіцієнтів. Однак така заміна призводить до спрощення регресійної моделі і деякої втрати інформації, пов'язаної з нелінійністю. [14,15]

Застосування нормалізуючих перетворень дозволяє перейти до лінійної регресії для нормалізованих даних. Наступним кроком необхідно побудувати довірчий інтервал з використанням t-розподілу Стьюдента і, застосувавши відповідне перетворення, перейти до нелінійної регресії та її довірчого інтервалу [16, 17]. Даний підхід позбавлений недоліків, зазначених у попередніх методах. В якості нормалізуючого перетворення використовується логарифмічне перетворення за натуральною основою [18, 19, 20].

Наступним етапом є визначення метрик ПЗ. Їх мета у виведенні одного або декількох значень особливостей ПЗ, що дає змогу порівнювати ці значення з подібними проектами зі специфічними стандартами, що притаманні певній компанії. З отриманих результатів можна дійти висновку щодо якості ПЗ та всього програмного процесу, а також, у разі необхідності, вживання подальших заходів.

Моделі оцінювання розміру програмного забезпечення поділяються на наступні п'ять категорій: аналогові; регресійні; моделі на основі експертних оцінок; моделі, які базуються на функціональних точках; параметричні моделі. [6]

Також існують різні методи для оцінювання розміру програмного забезпечення, що використовуються на практиці. Більшість з них походять від методу аналізу функціональних точок (FPA - Function Point Analysis). [2] Інші підходи полягають у проведенні функціонального вимірювання, щоб виразити функціональність у кількості параметрів, що представляють розмір

програмного забезпечення. Також існують методи визначення розміру ПЗ, що включають оцінювання на основі варіантів використання (Use Case) [21]. Але найбільш поширеною та вживаною методологією визначення розміру програмного забезпечення є підрахунок кількості рядків, написаних у вихідному коді програмного забезпечення.

Серед методик підрахунку кількості рядків коду (LOC) є дві основні:

- по числу фізичних рядків - визначається як загальне число рядків вихідного коду, включаючи коментарі та порожні рядки;
- по числу логічних рядків коду - визначається як загальна кількість команд і залежить від використовуваної мови програмування. Якщо мова підтримує розміщення кількох команд в одному рядку, то один фізичний рядок повинен бути врахований як кілька логічних, якщо він містить більше однієї команди мови.

Також існують похідні від основних методик (SLOC), які в залежності від завдання можуть містити додаткову інформацію за такими показниками як:

- число порожніх рядків;
- число рядків, що містять коментарі;
- відсоток коментарів (відношення рядків коду до закоментованих строк);
- середнє число рядків для функцій (класів, файлів);
- середня кількість рядків, що містять вихідний код для функцій (класів, файлів);
- середнє число рядків для модулів і т.д.

Аналіз функціональних точок (Function points) - це метод вимірювання розміру програмного забезпечення з точки зору користувачів системи. Метод був розроблений А.Альбрехтом ще в середині 1970-х років. [2]

Метод Use Case Points (UCP) - оцінка розміру проектів на основі діаграм UML і методології RUP (Rational Unified Process). Як і багато інших сучасні методів оцінки, UCP базується приблизно на тих же принципах, що і метод

функціональних точок. Головна відмінність полягає в заміні одиниць вимірювання з функціональних точок на варіанти використання. [21]

Варто також звернути увагу на вдосконалену модель COCOMO II - це модель регресії, що заснована на кількості рядків коду. Це процедурна модель оцінювання витрат для програмних проектів часто використовується для надійного прогнозування різних параметрів, пов'язаних з програмним проектом, таких, як: розмір проекту, зусилля на розробку, витрати, час та якість проекту, які необхідні для впровадження програмного забезпечення.

При використанні точної та надійної методології для оцінювання розміру можна зробити висновок про якість програмного забезпечення та навіть всього процесу розробки і, за необхідності, вжити подальших заходів для удосконалення розробки. [3]

Для перевірки адекватності лінійного рівняння регресії використаємо коефіцієнт детермінації (1.1):

$$R^2 = \frac{\sum_{i=1}^{n}(Y_{розр} - Y_{сер})^2}{\sum_{i=1}^{n}(Y_{факт} - Y_{сер})^2}, \qquad (1.1)$$

де $Y_{факт}$ - емпіричне значення y; $Y_{розр}$ - розрахункове значення y; $Y_{\text{сер}} = \frac{1}{n}\sum_{i=1}^{n} Y_{\text{ф}}$ середнє значення випадкової величини y.

Значення R^2 характеризує собою частку дисперсії, яка обумовлена регресією, в загальній дисперсії показника y. Коефіцієнт детермінації R^2 приймає значення від 0 до 1. Чим ближче значення коефіцієнта за модулем до 1, тим тісніше зв'язок результативної ознаки з досліджуваними факторами. При значенні $R^2>0,5$ можна вважати, що дана модель є прийнятною.

Достатньо ефективною та результативною можна вважати модель з показником детермінації $R^2>0,8$. Якщо $R^2=1$, тоді лінія регресії точно відповідає усім спостереженням та вимогам, а модель можна вважати адекватною та достовірною. [22]

Величина коефіцієнта детермінації (1.1) виступає значущим критерієм оцінки якості лінійних і нелінійних моделей. Чим вагоміша частка пояснюваної варіації, тим менша роль інших факторів, а отже, модель регресії краще апроксимує вихідні дані і такою регресійною моделлю можна скористатися для прогнозу значень результативного показника.

Для перевірки якості знайденого рівняння регресії, окрім критерію R^2, використовується також сума квадратів відхилень S_v:

$$S_y = \sum_{i=1}^{n} \left[Y_{\text{факт}} - Y_{\text{розр}} \right]^2, \text{ де} \qquad (1.2)$$

– $Y_{факт}$, - фактичне значення випадкової величини *y;*

– $Y_{розр}$ - розрахункове значення (згідно рівняння регресії).

Сума квадратів відхилень S_v використовується для перевірки якості як нелінійного, так і лінійного рівняння регресії. Цей параметр також використовується для порівняння різних моделей. [22]

Величину відносної похибки для лінійного та нелінійного рівнянь регресії можна розрахувати за наступною формулою:

$$\vartheta = \frac{\sqrt{D}}{\bar{y}} * 100\%, \text{ де} \qquad (1.3)$$

– $\bar{y}$ - середнє значення результативної ознаки;

– $\sqrt{D}$ – стандартна похибка.

Таким чином, показник середньої величини відносної похибки ρ можна розрахувати за формулою:

$$\rho = \frac{1}{n} \sum_{i=1}^{n} \vartheta_i, \qquad (1.4)$$

Рівень прогнозування $\tau(l)$ можна розрахувати за формулою:

$$\tau(l) = \frac{k}{n}, \qquad (1.5)$$

де k - кількість значень з $\vartheta \leq l$. [22]

При вивченні складних процесів та явищ часто застосовується саме моделювання. В даному випадку, модель — це об'єкт, на якому відтворюються певні характеристики досліджуваного явища, а моделювання — це конкретне відтворення цих характеристик, що дає змогу вивчати можливу поведінку явища без проведення експериментів над ним.

Моделювання є важливим інструментом наукової абстракції, що допомагає виокремити, уособити та проаналізувати суттєві для даного об'єкта наступні характеристики: властивості, взаємозв'язки, структурні та функціональні параметри.

Вихідні дані для оцінювання розміру програмного забезпечення, як правило, не мають характеристик нормального розподілу, що в результаті призводить до помилок у розрахунках та негативно впливає на достовірність отриманих результатів - на оцінювання розміру ПЗ. Щоб уникнути даної проблеми, варто перед тим, як будувати математичну модель, виконати нормалізацію початкових даних.

Для нормалізації негаусівських даних найдоцільніше використати перетворення на основі десяткового або натурального логарифму, перетворення Бокса-Кокса, перетворення Джонсона та інші. В якості нормалізуючого перетворення планується використовувати перетворення на основі натурального логарифму.

Виходячи з вищевказаного, побудуємо довірчий інтервал лінійного рівняння регресії за наступною формулою:

$$Y_{(x_k)} \pm t_{(\frac{\alpha}{2}, n-2)} * \left[S * \sqrt{\frac{1}{n} + \frac{(x_k - \bar{x})^2}{\sum_{i=1} (x_i - \bar{x})^2}} \right], \text{ де} \qquad (1.6)$$

$$S = \sqrt{\frac{\sum_{i=1}(y_i - y_i)^2}{n - p - 1}}$$

де Y_{x_k} - значення *y* розраховане за рівнянням регресії; $t_{(a/2,n-2)}$ - квантіль *t*-розподілу Стьюдента; α - рівень значущості; *n* - кількість значень випадкових величин у вибірці; *p* –кількість змінних.

Інтервал прогнозування лінійного рівняння регресії побудуємо за наступною формулою:

$$y = Y_{x_k} \pm t_{(\frac{\alpha}{2},n-2)} * S * \sqrt{1 + \frac{1}{n} + \frac{(x_k - \bar{x})^2}{\sum_{i=1}(x_i - \bar{x})^2}}, \quad (1.7)$$

$$S = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_i - Y_{x_k}\right)^2}{n - p - 1}}.$$

де Y_{x_k} - значення *y* розраховане за рівнянням регресії; $t_{(a/2,n-2)}$ - квантіль *t*-розподілу Стьюдента; α - рівень значущості; *n* - кількість значень випадкових величин у вибірці; *p* –кількість змінних. [22]

Виконавши необхідні розрахунки, отримаємо більш точну математичну модель для оцінки розміру веб-застосунків розроблених з використанням фреймворку Django.

Для перевірки адекватності математичної моделі оцінювання розміру програмного забезпечення використаємо коефіцієнт детермінації R^2 або суму квадратів відхилень. Перед тим, як будувати математичну модель, потрібно нормалізувати вхідні дані. У якості нормалізуючого перетворення буде використовуватись частковий випадок одномірного перетворення Бокса-Кокса, при $\lambda=0$. Значення залежної змінної буде нормалізовуватись логарифмуванням за натуральною основою. Перетворення Бокса-Кокса має наступний вигляд:

$$x(\lambda) = \begin{cases} \frac{x^{\lambda}-1}{\lambda}, \lambda \neq 0, \\ \ln(x), \lambda = 0. \end{cases}, \text{ де} \quad (1.8)$$

– x – випадкова величина, яка підлягає нормалізації;

– λ – параметр розподілу.

Перетворення мають наступні часткові випадки [23]:

$$\lambda = -1{,}0, \quad \mathrm{x_i}(\lambda) = \frac{1}{\mathrm{x_i}};$$

$$\lambda = -0{,}5, \quad \mathrm{x_i}(\lambda) = \frac{1}{\sqrt{\mathrm{x_i}}};$$

$$\lambda = 0{,}0, \quad \mathrm{x_i}(\lambda) = \ln(x_i); \quad (1.9)$$

$$\lambda = 0.5, \quad \mathrm{x_i}(\lambda) = \sqrt{x_i};$$

$$\lambda = 2{,}0, \quad \mathrm{x_i}(\lambda) = x_i^2.$$

Значення параметру розподілу визначаються декількома шляхами:

- за максимумом логарифму функції правдоподібності (n –кількість вибірок);
- в результаті вирішення задачі математичного програмування.

Вибір конкретного нормалізуючого перетворення необхідно виконувати залежно від емпіричних даних, які мають піддаватися обробці. Наступним кроком слід перевірити адекватність отриманого закону розподілу.

Використання нормалізуючих перетворень дозволяє перейти до лінійної регресії для нормалізованих даних, потім необхідно для неї побудувати довірчий інтервал та інтервал прогнозування. Останнім кроком буде, шляхом застосування відповідного зворотнього перетворення, перехід до нелінійної регресії. Таким чином отримаємо більш точну математичну модель для оцінки розміру веб-застосунків з кодуванням на базі фремворку Django.

Для побудови нелінійного рівняння регресії необхідно використати наступні обрані метрики:

– Y - вибірку восьмивимірних негаусових даних з фактичним

розміром ПЗ в тисячах рядків коду (LOC);

- x_1 - загальна кількість класів;
- x_2 - середня кількість атрибутів;
- x_3 - середня кількість методів;
- x_4 - середня кількість post методів;
- x_5 - середня кількість get методів;
- x_6 - середня кількість методів-конструкторів;
- x_7 - кількість відносин між класами.

У концептуальній моделі даних взято 54 одиниці програмних продуктів, написаних з використанням Django-фреймворку, з найбільш відомих репозиторіїв: GitHub, SourceForge, AwesomeOpenSource. Надалі планується застосування даних для побудови лінійного та нелінійного регресійного рівняння для оцінювання розміру ПЗ (веб-застосунків написаних з використанням Django-фреймворку).

2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При використанні об'єктно-орієнтованої методології для візуалізації компонентів, зв’язків та процесів розроблюваного програмного забезпечення використовуються засоби UML.

В ескізному проекті наводять обов’язково: діаграму варіантів використання (Use Case); специфікації до варіантів використання; діаграми діяльності або діаграми станів; концептуальну модель даних; проект інтерфейсу користувача (ескізи форм) [21].

Специфікація варіантів використання має відображати наступне: передумови варіантів використання (їх початковий стан); дії, якими запускається варіант використання; сценарії варіантів використання; умови завершення варіанту використання; результат, отриманий після завершення варіанту використання; післяумови варіанта використання (його кінцевий стан). Також необхідно вказати основні, альтернативні і заборонені шляхи виконання варіантів використання, взаємодії акторів з системою та інформація, якою вони обмінюються. Інформаційна модель може бути представлена у вигляді діаграмами класів UML або, наприклад, діаграмою «сутність - зв'язок».

У проекті інтерфейсу користувача визначаються: елементи призначеного для користувача інтерфейсу, необхідні акторам для виконання варіантів використання; зв'язки елементів інтерфейсу; застосування елементів інтерфейсу в різних варіантах використання.

У «технічному проекті» програмне забезпечення подається вже як система, що складається з досить простих для реалізації елементів: модулів, класів. Також визначаються зв'язки між елементами за даними і управлінням, що забезпечує функціонування всієї системи відповідно до вимог, а також логічні моделі структур даних.

2.1 Ескізний проект програмного забезпечення

Для побудови складної інформаційної системи необхідним етапом є проектування. В останнє десятиліття в комп'ютерному світі намітилася тенденція проектування систем візуальними (наочними) моделями. Причому в нових методах проектування складних комп'ютерних систем, наприклад об’єктно-орієнтованому проектуванні (ООП) і об’єктно-орієнтованому аналізі (ООА), наочні моделі дуже часто зв'язуються з такими зоровими образами як "погляди", спрямовані на складну систему з різних точок зору. Набір з декількох наочних моделей (модельних поглядів) створює у свідомості фахівців інтегральний образ складної комп'ютерної системи, яку вони спільно проектують. Разом з тим, наочні моделі є ефективним засобом документування комп'ютерних систем і їх програмних забезпечень, а також мовою спілкування між програмістами, системними аналітиками й замовниками систем.

Головною метою моделювання на етапі ескізного проектування є відображення процесу створення програмного забезпечення, визначення взаємозв’язків між модулями ПЗ у вигляді візуалізації в діаграмах.

На поточний момент існує багато методологій моделювання систем, тому необхідно провести їх аналіз та визначити найзручніші засоби для моделювання розроблюваної системи. Найбільш відомими візуальними моделями, використовуваними для проектування комп'ютерних систем і їхніх програмних забезпечень, є діаграми мови UML (Unified Modeling Language) і стандарту IDEF0, таблиці й діаграми стандарту IDEF1X. Ці візуальні моделі мають математичну основу у вигляді теорій графів, множин і матриць.

Зрозуміло, що об'єднання тексту програми (її вихідного коду) з характеристиками об'єкта автоматизації здійснюється тільки у свідомості програміста, а документальний зв'язок між ними відсутній.

Діаграми й специфікації мови UML зв'язали вихідний текст програми з характеристиками об'єкта автоматизації. При цьому UML діаграми опираються на теоретичний фундамент. Наявність теоретичної основи дозволяє спростити

операції перетворення UML-діаграм, зображених на екранах дисплеїв, і зменшити об'єм пам'яті, необхідної для зберігання діаграм.

UML-діаграми можуть бути перетворені у вихідний код (пряме перетворення) і навпаки вихідний код може бути перетворений в діаграми (зворотне перетворення). У деяких випадках пряме перетворення може здійснюватися автоматично за допомогою програм-конвертерів. У цей час йде активна робота над рішенням проблеми прямого перетворення діаграм UML. Зворотне перетворення може виконати тільки людина.

Документування вихідних кодів програм UML-діаграмами й специфікаціями створює єдину мову спілкування між програмістами, системними аналітиками й замовниками автоматизованої системи. Але саме головне, що мова UML надала можливість широкої стандартизації мов програмування. Відомо, що в різних мовах програмування використаються однакові операції й методи, але вони мають різні назви й символьні позначення. Мова UML дозволяє стандартизувати як самі операції й методи мов програмування так і їхню термінологію.

Отже, для проектування розроблюваного ПЗ доцільно буде використати саме моделювання UML [23]. В якості середовища моделювання буде використано безкоштовну онлайн-систему Draw.io.

2.1.1 Розробка моделі варіантів використання

Для графічного відображення майбутнього ПЗ для оцінювання розміру веб–застосунків на базі фреймворка Django, побудуємо діаграму варіантів використання. Обов’язковими елементами такої діаграми є: прецеденти, актори або дійові особи і безпосередньо відносини між акторами та варіантами використання.

Актором називається певний об'єкт моделі, суб'єкт чи система, яка взаємодіє з модельованою системою з метою вирішення певних завдань. Це може бути людина, технічний пристрій, програма або будь-яка інша система,

що служить джерелом впливу на модельовану систему [22].

Модель варіантів використання представлена на рисунку 2.1 з актором «Користувач».

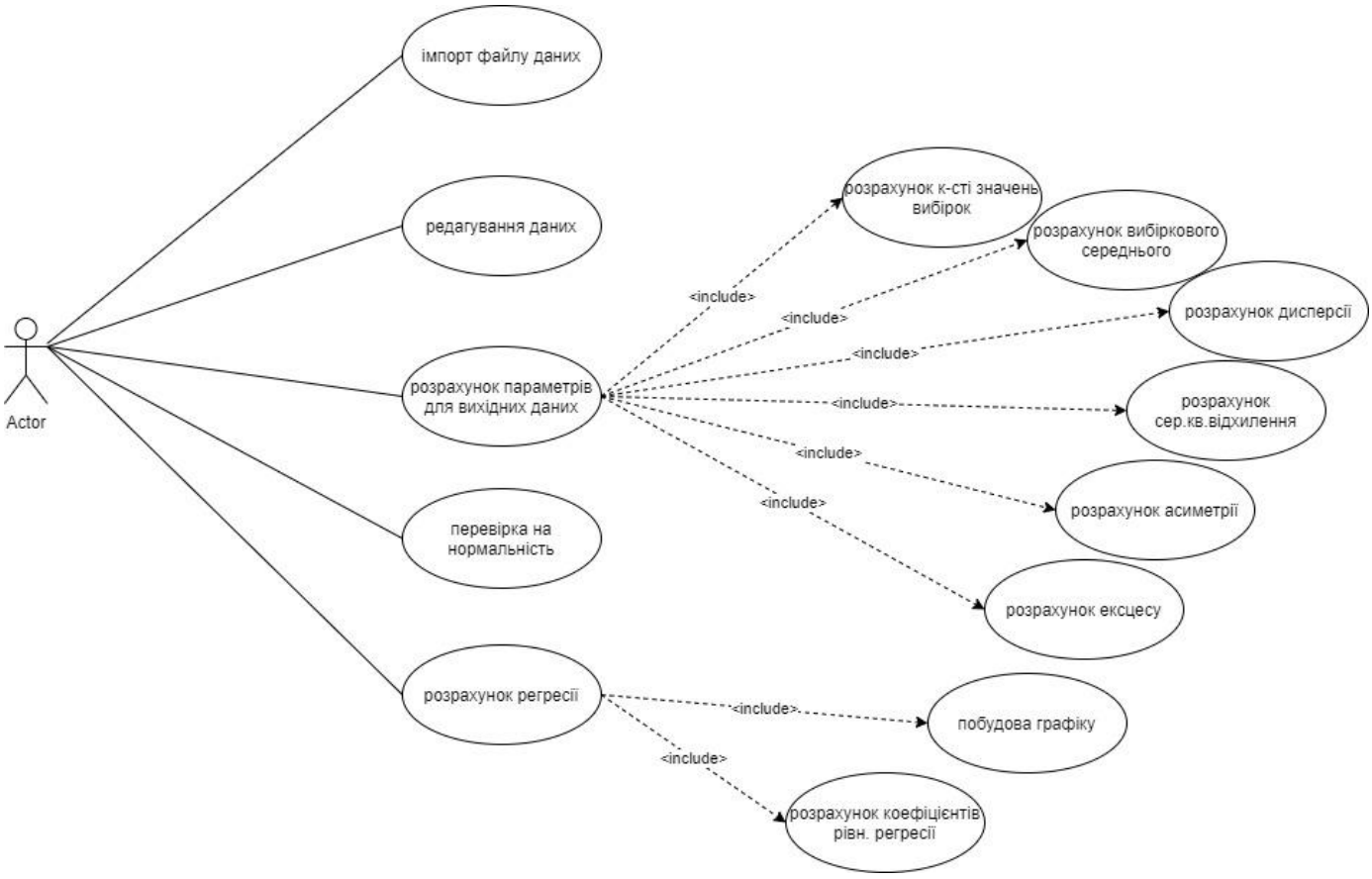

Рисунок 2.1 — Діаграма варіантів використання ПЗ

Функції користувача: завантажує файл зі статистичними даними до програми, заносить додаткові дані, редагує дані, розраховує параметри для вихідних та нормалізованих вибірок, обирає змінні, нормалізує вихідні вибірки, перевіряє вихідні та нормалізовані вибірки на нормальність розподілу, будує лінійне та нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування.

2.1.2 Розробка специфікації варіантів використання

Потік подій варіанта використання має такі складові: короткий опис; передумови; основний потік подій; альтернативний потік подій; постумови. В табл.2.1 наведено опис варіантів використання з діаграми на рис.2.1.

Таблиця 2.1 – Опис варіантів використання системи

Опис	Складова
Імпорт файлу даних	
Короткий опис	Дана функція надає змогу Користувачу імпортувати до програми файл із вихідними статистичними даними.
Передумови	Система запущена та виводить початкову форму програми.
Основний потік подій	1. Система виводить форму для імпорту файлу. 2. Користувач вибирає файл з файлової системи та підтверджує вибір; 3. Система перевіряє формат файлу на відповідність; 4. Якщо формат вірний - зчитуються дані вихідних вибірок. 5. Результат імпорту виводиться на екрані.
Альтернативний потік подій	1. Система виводить форму для імпорту файлу; 2. Користувач вибирає файл з файлової системи та підтверджує вибір; 3. Система перевіряє формат файлу на відповідність; 4. Якщо формат не вірний – система видає повідомлення про помилку зчитування даних; 5. Система пропонує обрати інший файл з файлової системи. 6. Користувач обирає інший файл з файлової системи. 7. Якщо формат вірний - зчитуються дані вихідних вибірок. 8. Результат імпорту виводиться на екрані
Постумови	Зчитуються дані вихідних вибірок із файлу.
Редагування даних	
Короткий опис	Дана функція надає змогу Користувачу змінювати допустимі поля з початковими даними.
Передумови	Система запущена, табличні поля для вводу доступні для редагування
Основний потік подій	1. Користувач вводить дані до поля вводу; 2. Система перевіряє коректність введених даних; 3. Система обробляє нові дані; 4. Система приймає нові дані; 5. Система виводить результат у формі програми.

Продовження таблиці 2.1

Опис	Складова
Альтернативний потік подій	1. Користувач вводить дані до поля вводу; 2. Система перевіряє коректність введених даних; 3. Система повідомляє про неправильний формат введених даних; 4. Система скидає останнє введення користувача; 5. Система виводить результат у формі програми
Постумови	Оновлення даних в полях вводу та в полях результатів розрахунків.
Розрахунок параметрів для вихідних даних	
Короткий опис	Дана функція призначена для розрахунку параметрів для вихідних вибірок: кількість значень вибірки; вибіркове середнє; дисперсія; середньоквадратичне відхилення; асиметрія; квадрат асиметрії; ексцес.
Передумови	Користувач імпортував файл з даними і перевірив внесені дані для розрахунків та обрав функцію розрахунку.
Основний потік подій	1. Система виконує розрахунок параметрів; 2. Система виводить результати розрахунків; 3. Система пропонує зберегти результати в окремий файл.
Альтернативний потік подій	Відсутній
Постумови	Вивід розрахованої інформації до форми
Перевірка нормалізованих вибірок на нормальність	
Короткий опис	Функція виконує перевірку нормалізованих вибірок на нормальність розподілу
Передумови	Користувач імпортував коректний файл з даними для подальшої обробки
Основний потік подій	1. Користувач обрав функцію «Перевірити на нормальність»; 2. Система виконує аналіз даних на нормальність розподілу; 3. Система виводить повідомлення про нормальність розподілу та відображає розраховані параметри.
Альтернативний потік подій	Відсутній
Постумови	Вивід розрахованої інформації до форми

Продовження таблиці 2.1

Опис	Складова
Розрахунок регресії	
Короткий опис	Функція виконує побудову лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування
Передумови	Користувач імпортував коректний файл з даними, провів нормалізацію вихідних вибірок
Основний потік подій	1. Користувач обрав функцію «Розрахунок регресії»; 2. Система відкриває нове вікно для виведення форми розрахунку параметрів регресії; 3. Користувач натискає кнопку «Розрахувати» на вкладці «Лінійне рівняння регресії»; 4. Користувач натискає «Побудувати графік»; 5. Система розраховує коефіцієнти рівняння регресії; 6. Система виводить результати на екран; 7. Система будує графік для рівняння регресії, довірчого інтервалу та інтервалу прогнозування. 8. Система пропонує виконати збереження результатів до окремого файлу.
Альтернативний потік подій	Відсутній
Постумови	Вивід розрахованої інформації та графіка на екран

2.1.3 Побудова концептуальної моделі даних

Концептуальна модель — модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

Концептуальна (змістовна) модель — це абстрактна модель, що визначає структуру модельованої системи, властивості її елементів і причинно-наслідкові зв'язки, властиві системі і суттєві для досягнення мети моделювання.

Основні елементи концептуальної моделі:

- умови функціонування об'єкта, визначені характером взаємодії між об'єктом і його оточенням, а також між елементами об'єкта;

- мета дослідження об'єкта та напрямок покращення його функціонування;

- можливості керування об'єктом, визначення складу керованих змінних об'єкта.

Діаграма «сутності - зв’язку» виступає засобом опису схеми бази даних на концептуальному рівні проектування. На діаграмах концептуального рівня сутності зображуються прямокутниками, атрибути - еліпсами, зв’язки – ромбами [23].

Однак дана програмна розробка не є програмним забезпеченням подібного типу структури та не потребує функціональності бази даних. Вхідні дані ПЗ отримуватиме за допомогою завантаження файлу та зчитування з нього початкових даних для подальших розрахунків.

Після розрахунків усі отримані параметри будуть записуватись у вихідний файл у форматі CSV. Перевагою даного формату є те, що він може використовуватись як вхідний файл до значної кількості інших програмних продуктів та є універсальним в певному розумінні.

2.1.4 Розробка діаграм діяльності

Діаграма діяльності — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія (англ. action) є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати

їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку. [24]

Діаграма активностей може вважатись формою блок-схеми. На рисунку 2.2 зображено сценарій варіантів використання для роботи розроблюваного програмного забезпечення.

Рисунок 2.2 – Діаграма діяльності для розроблюваного ПЗ

Наведена діаграма сценаріїв діяльності для прецедентів типово показує послідовність дій та події, що ініціюють дії або є їх кінцевим результатом.

2.1.5 Розробка прототипу інтерфейсу користувача

Робота з програмним забезпеченням відбувається за допомогою інтерфейсу користувача. Беручи до уваги вимоги, що зазначені у технічному завданні, визначимо зовнішній інтерфейс ПЗ.

На рисунку 2.3 представлено прототип користувацького інтерфейсу програмного забезпечення. Для проектування даного інтерфейсу використано спеціалізоване програмне забезпечення Balsamiq Wireframes [25].

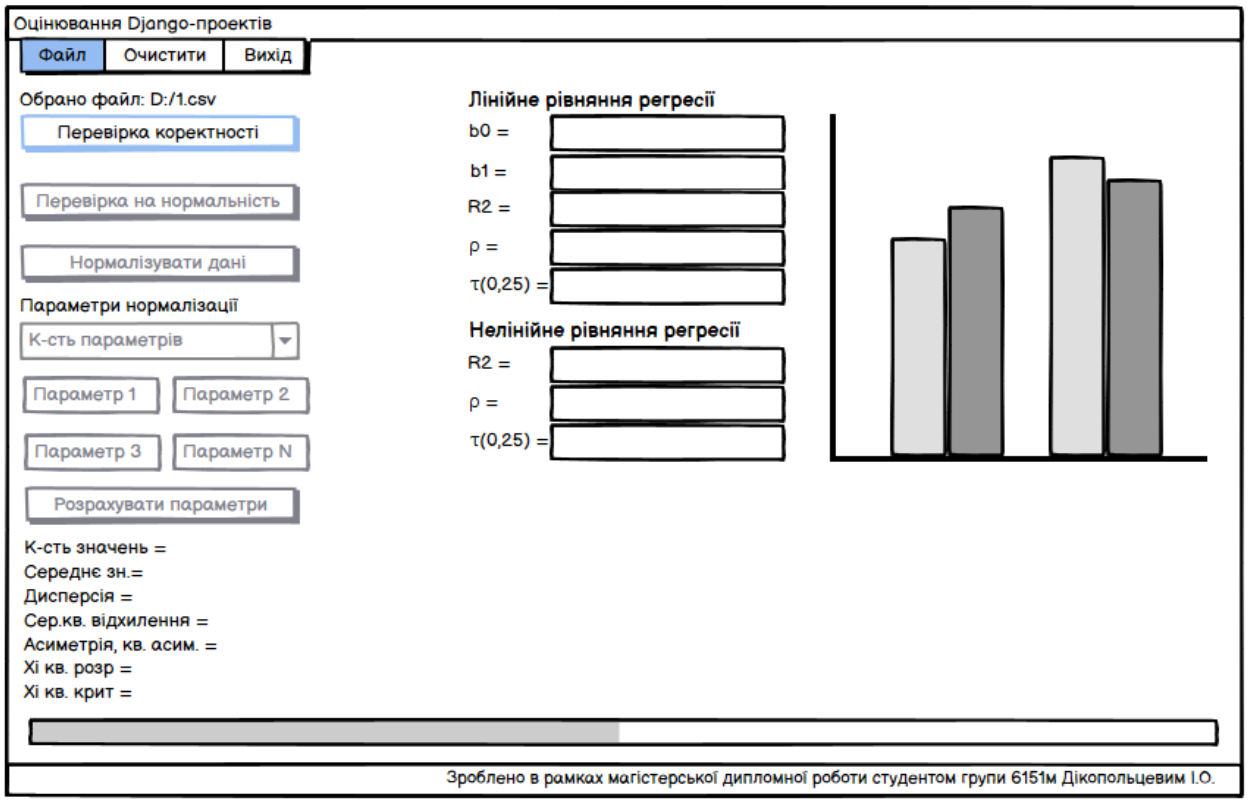

Рисунок 2.3 - Ескіз головної форми програмного забезпечення

Після успішної перевірки коректності вхідних даних стає доступною кнопка «Перевірка на нормальність» та «Нормалізувати дані» (не існує такої можливості, щоб дані були нормалізовані відразу без попередньої

нормалізації). Після розрахунку початкових параметрів буде можливість перевірити розподіл та перейти далі до нормалізації. Після нормалізації переходимо до обчислення регресії лінійної та нелінійної. Збоку праворуч відображається гістограма після нормалізації даних.

2.2 Технічний проект програмного забезпечення

2.2.1 Розробка статичної моделі програмного забезпечення

Діаграма класів — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини. В UML існують такі типи зв'язків:

- Асоціація ;
- Агрегація;
- Композиція;
- Наслідування. [26]

Діаграма класів програмного забезпечення для автоматизації роботи розроблюваної програми представлена на рисунку 2.4. На діаграмі класів визначені наступні елементи: клас «Оцінювання розміру», клас «Нормалізація», клас «Регресія», перелік «Параметри вибірки». Вказано атрибути та їх типи, а також методи.

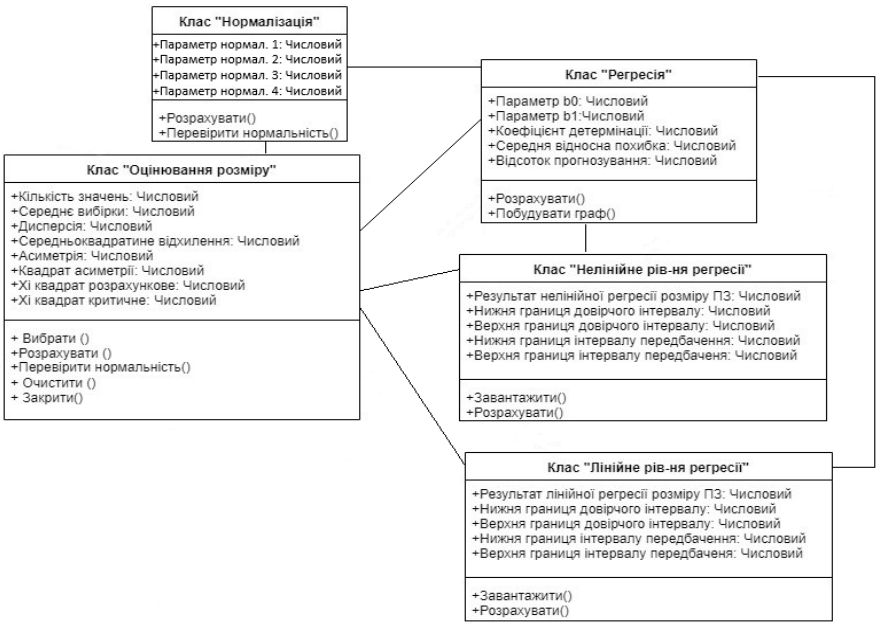

Рисунок 2.4 - Діаграма класів програмного забезпечення

2.2.2 Розробка специфікації класів програмного забезпечення

Специфікація класу для об'єкта - це визначення його властивостей (які характеризують стан об'єкта) і методів (які є способом реалізації його поведінки). Специфікація класів програми представлена в таблиці 2.2.

Таблиця 2.2 - Специфікація класів ПЗ

Опис	Складова
Оцінювання розміру проекту	
Відповідальність	Даний клас відповідає за оцінювання розміру вихідних вибірок.
Атрибути	Кількість значень вибірки, середнє вибірки, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, χ^2розрахункове, χ^2критичне
Методи	Вибрати(): обрати файл в файловій системі Розрахувати(): розрахунок параметрів вибірки Перевірити нормальність(): перевірка на нормальність активного розподілу Очистити(): очистка усіх полів вводу та рез-тів розрахунків, графіка з гістограмою після нормалізації даних Закрити(): вихід з програми

Продовження таблиці 2.2

Опис	Складова
Лінійне рівняння регресії	
Відповідальність	Даний клас відповідає за інформацію про параметри вибірки лінійного рівняння регресії
Атрибути	Результат лінійної регресії розміру ПЗ, нижня границя довірчого інтервалу, верхня границя довірчого інтервалу, нижня границя інтервалу передбачення, верхня границя інтервалу передбаченя
Методи	Завантажити(): імпорт даних з класу «Регресія» Розрахувати(): розрахунок за обраними параметрами
Нормалізація вибірки	
Відповідальність	Даний клас відповідає за нормалізацію вихідної вибірки.
Атрибути	Параметр нормалізації 1; Параметр нормалізації 2; Параметр нормалізації 3; Параметр нормалізації 4
Методи	Перевірити нормальність(): визначення, чи відповідає вибірка вимогам нормального закону розподілення Розрахувати(): розрахунок за обраними параметрами
Нелінійне рівняння регресії	
Відповідальність	Даний клас відповідає за інформацію про параметри вибірки нелінійного рівняння
Атрибути	Результат нелінійної регресії розміру, нижня границя довірчого інтервалу, верхня границя довірчого інтервалу, нижня границя інтервалу передбачення, верхня границя інтервалу передбаченя:
Методи	Завантажити(): імпорт даних з класу «Регресія» Розрахувати(): розрахунок за обраними параметрами
Регресія	
Відповідальність	Даний клас відповідає за інформацію про лінійне та нелінійне рівняня регресії.
Атрибут	Параметри наближення, коефіцієнт детермінації, середня відносна похибка, відсоток прогнозування.
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів. Побудувати графік(): побудова графіку рівняння регресії, довірчого інтервалу та інтервалу прогнозування.

2.2.3 Побудова динамічної моделі програмного забезпечення

Динамічна модель програми може бути представлена:

- діаграмами станів.
- діаграмами послідовності;
- діаграмами взаємодії;
- діаграмами діяльності.

Динамічна модель програми представлена у вигляді діаграми послідовності. У магістерській роботі представлені діаграми послідовності за деякими варіантами використання.

Діаграма послідовності (sequence diagram) - діаграма, на якій показані взаємодії об'єктів, упорядковані за часом їхнього прояву. Нижче наведено декілька діаграм послідовності для розроблюванного ПЗ на рисунку 2.5.

У варіанті використання «Нормалізація» користувач спочатку проходить етап розрахунку параметрів вихідних вибірок. Потім вводить параметри початкового наближення. Система перевіряє наявність та допустимість введених параметрів. Якщо дані коректні, то повертає розраховані оптимальні параметри. Діаграма послідовності для варіанту використання «Нормалізація» представлена на рисунку 2.5.

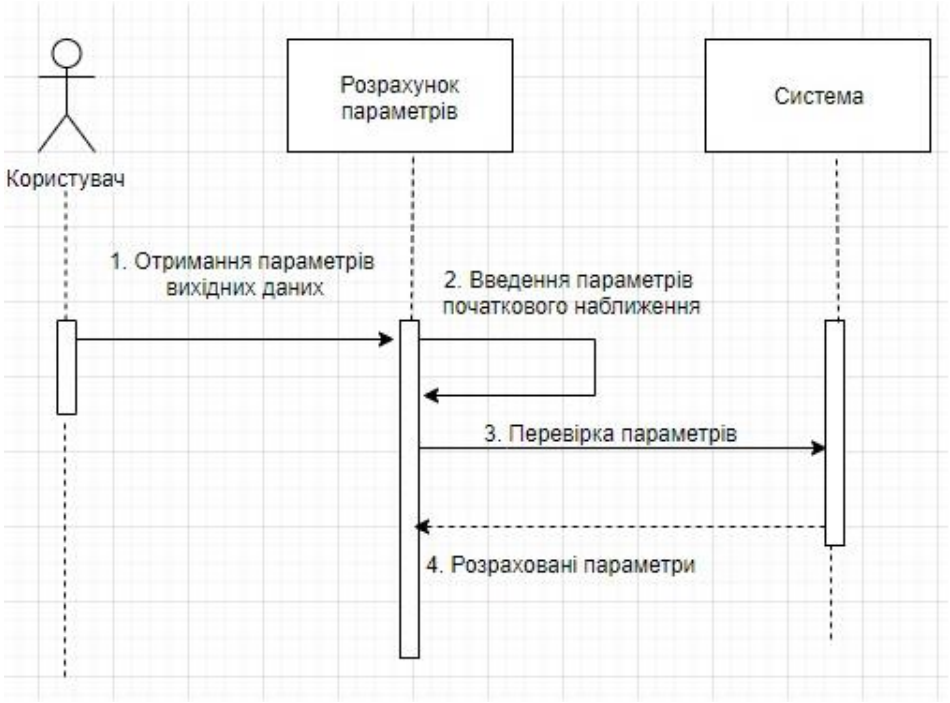

Рисунок 2.5 - Діаграма послідовності для варіанту використання «Нормалізація»

Діаграма діяльності (Activity diagram) дозволяє моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. Зазначені послідовності можуть являти собою альтернативні галузей процесу обробки даних або галузям, які можуть виконуватися паралельно. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму. Вони, як і діаграми станів та переходів, відображаються у вигляді орієнтованого графу, вершинами якого є дії, а ребрами – переходи між діями.

Діаграми діяльності доцільно використовувати для аналізу:

- змісту сценаріїв застосування проектованої системи;

- взаємодії потоків робіт різних сценаріїв;

- виконання сценаріїв у багатопроцесорних обчислювальних середовищах.

Ці діаграми широко використовуються в описі поведінки, що включає велику кількість паралельних процесів.

Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан виконується тільки після

завершення цієї операції.

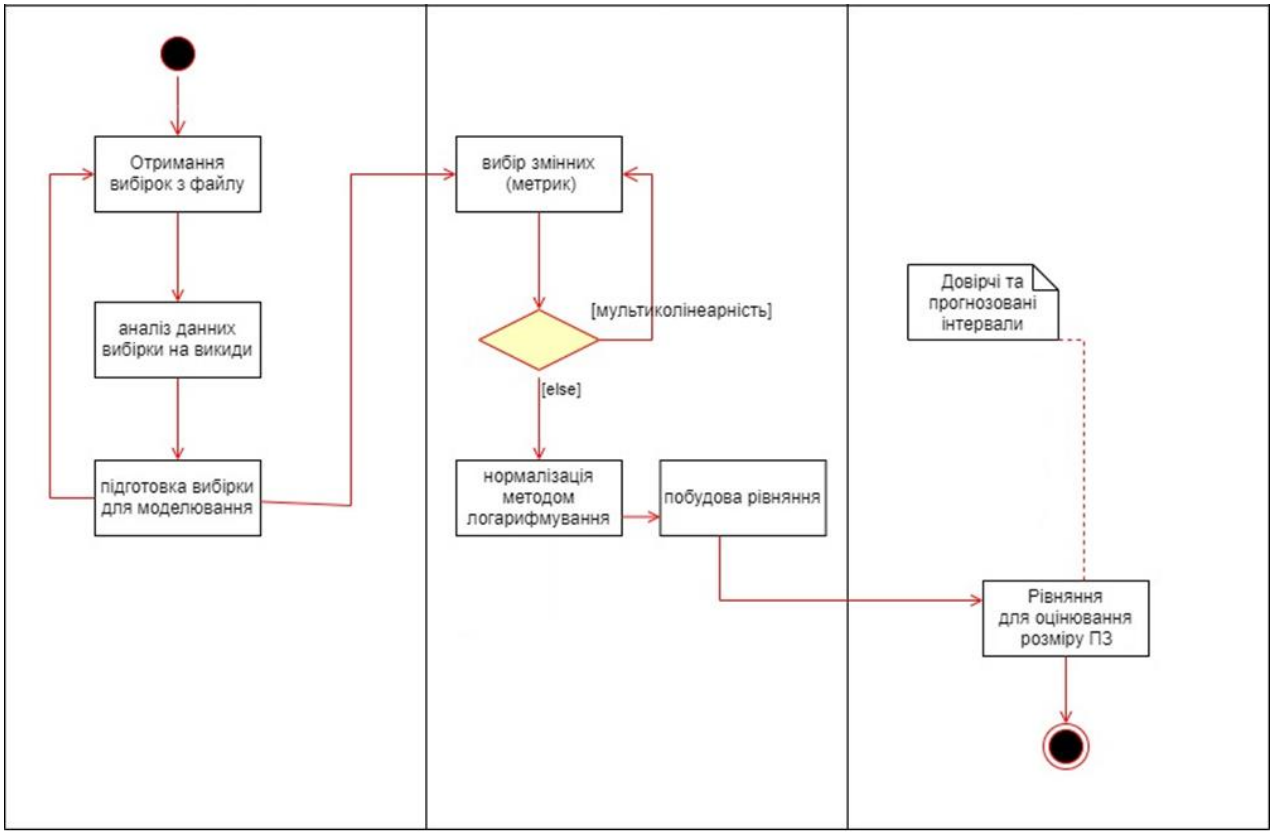

Рисунок 2.6 – Діаграма діяльності ПЗ оцінювання розміру проекту

2.3 Робочий проект програмного забезпечення

2.3.1 Обгрунтування вибору мови програмування

Для розробки програмного забезпечення для оцінювання розміру веб–застосунків на базі фреймворку Django, було обрано мову програмування Python (PyCharm).

PyCharm - це найінтелектуальніша Python IDE з повним набором засобів для ефективної розробки на мові Python. Випускається в двох варіантах - безкоштовна версія PyCharm Community Edition і PyCharm Professional Edition, що підтримує більший набір можливостей. PyCharm виконує інспекцію коду «на льоту», автодоповнення, в тому числі ґрунтуючись на інформації, отриманої під час виконання коду, навігацію по коду, забезпечує безліч рефакторингов. [27].

PyCharm підтримує сучасні фреймворки для веб-розробки: Django, Flask,

Google App Engine, Pyramid і web2py, що робить його універсальною IDE для швидкої розробки додатків. Можливості інтегрованого модульного тестування, перевірки коду, інтегрованого контролю версій, інструменти рефакторинга коду, набір інструментів для навігації проекту, виділення і автоматичного завершення.

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі: чистий синтаксис (для виділення блоків слід використовувати відступи); переносність програм (що властиве більшості інтерпретованих мов); можливість використання Python в діалоговому режимі; стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python; зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор); відкритий код. [7].

Розглянемо модель (схему) роботи типового веб-застосунку. При цьому не ставимо за мету повторити повністю його функціональність, а схематично відобразимо процеси, що відбуваються у програмі при застосуванні фреймворку Django. Моделювалися тільки ті функції, які є найбільш типовими для більшості веб-застосунків даного типу:

1) Веб-сервер отримує HTTP-запит від відвідувача сайту. Django не містить повноцінного веб-сервера і передбачається, що буде задіяна якась програма-веб-сервер або WSGI-сервер (Apache, nginx, LightHTTPd, Caddy, uWSGI і т.д.).

2) Веб-сервер передає запит Django через точку входу.

3) Django створює об'єкт HTTPRequest і пропускає його через ланцюжок Middleware.

4) Django переглядає urlconf і шукає уявлення, яке здатне обробити запит. Якщо його немає, повертається помилка HTTP 404 Not Found.

5) Django запускає знайдене представлення, передаючи йому Request і параметри з адреси.

6) Подання взаємодіє зі сховищем даних (через моделі), виконує певні (будь-які) дії, читає / записує файли. В результаті подання, має сформувати HTTP-відповідь. Дуже часто для цього застосовуються шаблони HTML-файлів, щоб не створювати веб-сторінку з нуля, а використовувати готові фрагменти.

7) Django пропускає створену поданням відповідь через ланцюжок Middleware.

8) Відповідь надається веб-серверу, а той, у свою чергу, передає відповідь відвідувачеві сайту. [1].

Схему роботи такого застосунку приведена на рис. 2.7. Зеленим відзначені блоки, які програмує розробник. Все інше - це готові компоненти Django.

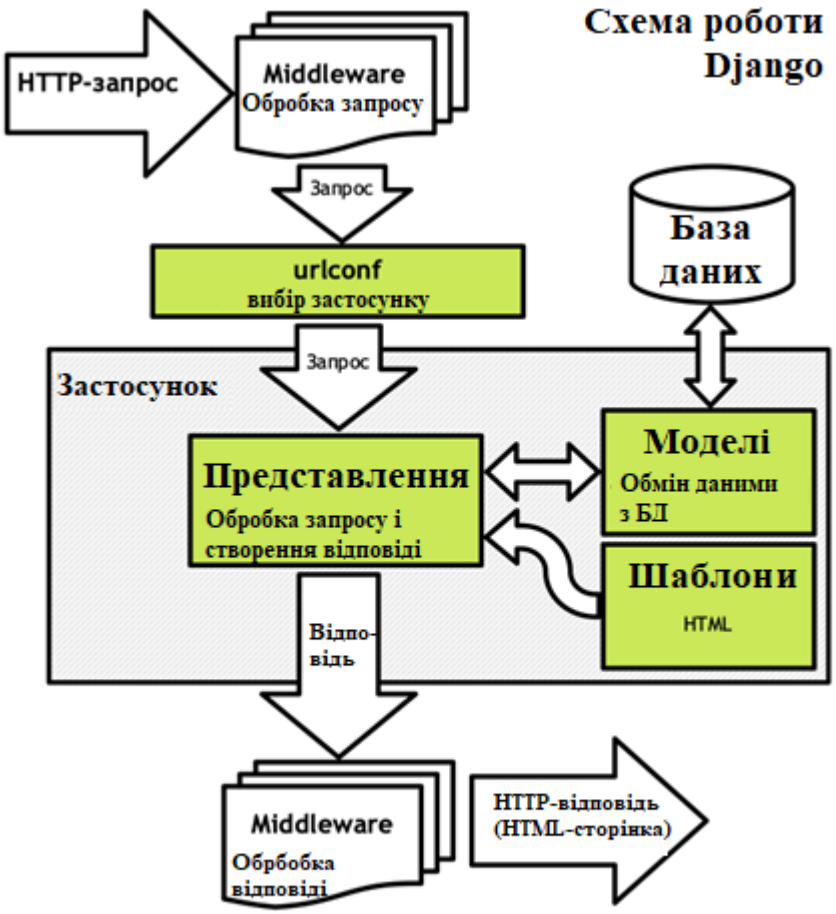

Рисунок 2.7 – Схема роботи типового веб-застосунку, запрограмованого з використанням фреймворку Django

Django - безкоштовний і відкритий фреймворк для створення веб-додатків, написаний мовою програмування Python. Веб-фреймворк - це набір компонентів, які оптимізують розробку веб-сайтів.

При побудові веб-сайту, завжди потребується певний постійний набір компонентів: спосіб управління авторизацією користувача (створення акаунту, вхід, вихід), панель управління, форми, спосіб завантажити файли, тощо.

Веб-розробники зіштовхуються з подібними проблемами під час розробки нових веб-продуктів, що змусило в певний час систематизувати частини коду у так звані фреймворки, що надають вже готові до використання програмні компоненти.

Рейтинг популярності мов програмування за 2020 рік, зведений до гістограми спеціалістами з аналізу інформації DOU.UA шляхом аналізу трендів основних пошукових систем та сайтів для пошуку проектів і фріласерів (Рис.2.8). Взагалі існує кілька рейтингів, що оцінюють популярність мов програмування. Серед них – TIOBE Software, Red Monk, PYRL, IEEE Spectrum. Незважаючи на деякі відмінності в розподілі місць мов, бузумовними лідерами в даному розрізі є саме Java, JavaScript, Python, C#.

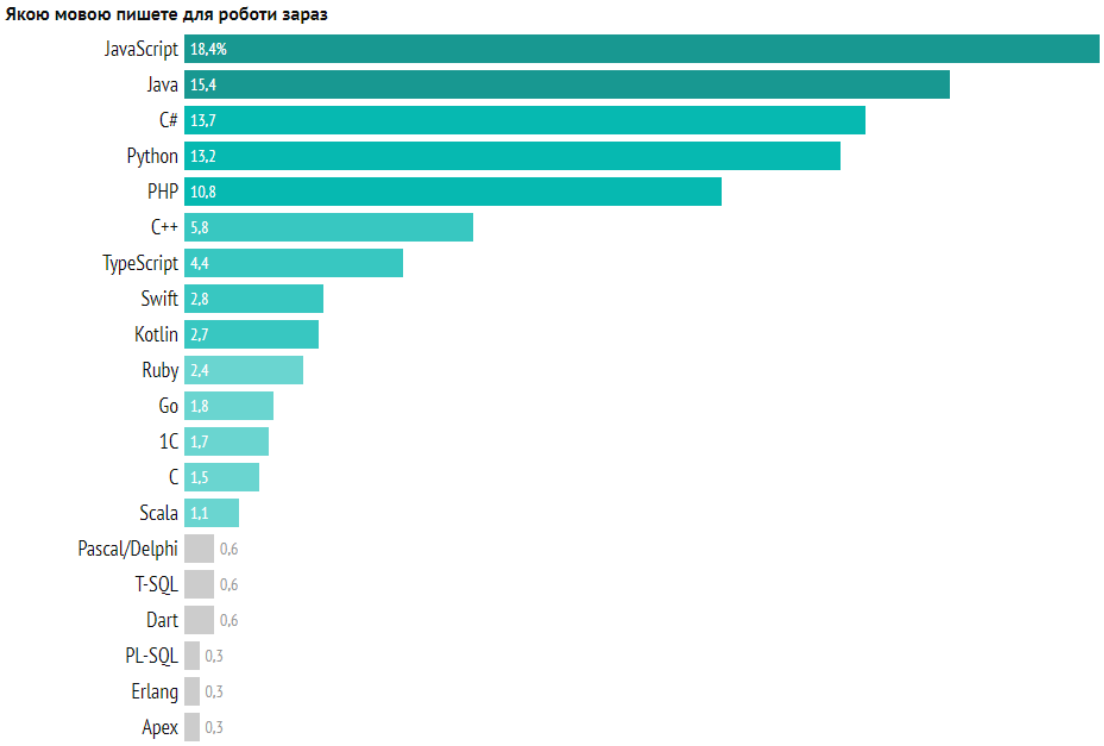

Рисунок 2.8 – Рейтинг популярності мов програмування за 2020 рік серед фрілансерів за версією DOU.UA

Рейтинг мов програмування (рис. 2.8) не дає відповіді на запитання, яка мова є кращою в плані написання коду. Однак, цей показник визначає популярність мови програмування. Незважаючи на деякі відмінності в розподілі місць мов, найбільш стрімкої популярності набирають саме Python та JavaScript. Саме ці мови програмування займають близько 50% всього обсягу використання мов програмування в світі.

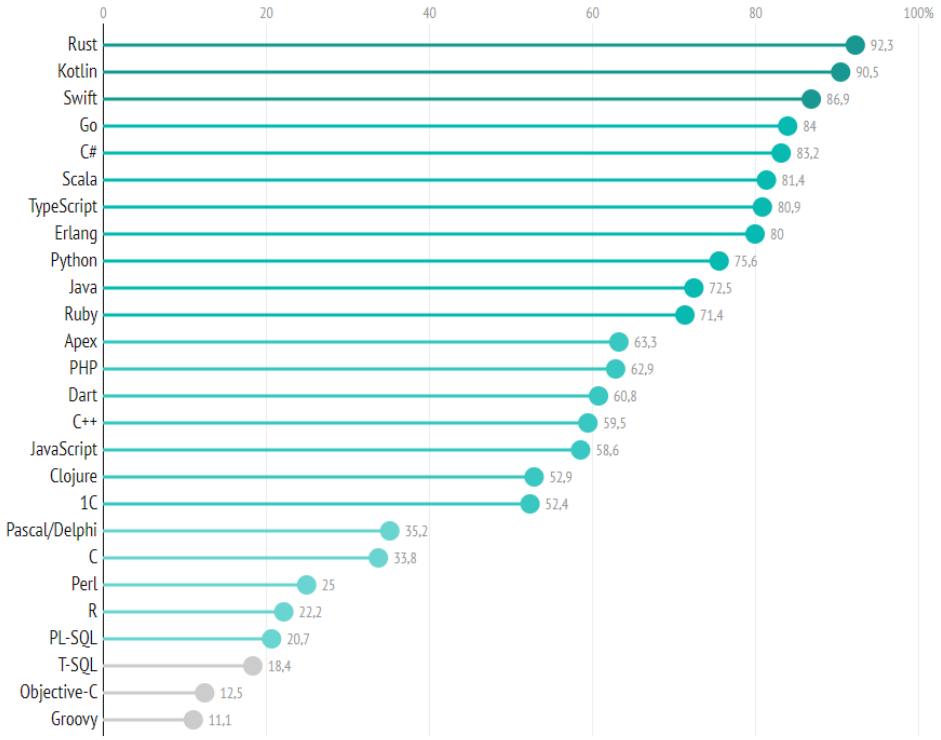

Рисунок 2.9 – Рейтинг мов програмування за версією компанії dou.ua за індексом вподобання на 2020 р.

На рисунку 2.9 представлено рейтинг мов програмування компанії dou.ua за індексом вподобання на 2020 р. [28]

Django - вільний фреймворк для розробки веб-додатків на мові Python, що використовує шаблон проектування MVC. Проект підтримується організацією Django Software Foundation.

Сайт на Django будується з одного або декількох додатків. Це одне з істотних архітектурних відмінностей цього фреймворка від деяких інших (наприклад, Ruby on Rails). Один з основних принципів фреймворка - DRY. Це принцип розробки програмного забезпечення, націлений на зниження повторення інформації різного роду, особливо в системах з безліччю шарів абстрагування.

Також, на відміну від інших фреймворків, обробка URL в Django конфігурується явно за допомогою регулярних виразів, а не виводиться автоматично зі структури моделей контролерів.

Для роботи з базою даних Django використовує власний ORM, в якому модель даних описується класами Python, і за нею генерується схема бази даних.

Проект Django цілком налаштовується користувачем середовища розробки. Головні компоненти фреймворку Django наступні:

- Об'єктно-реляційне відображення (object-relational mapping) для створення моделей;

- Бездоганний інтерфейс адміністратора, спеціально створений для кінцевих користувачів;

- Зручно спроектований механізм адресування (URL);

- Мова шаблонів для дизайнерів;

- Система кешування.

Веб-фреймворк Django використовується в таких великих і відомих сайтах, як Instagram, Disqus, Mozilla The Washington Times, Pinterest та ін.

Також Django використовується в якості веб-компонента в різних проектах, таких як Graphite - система побудови графіків і моніторингу, FreeNAS - вільна реалізація системи зберігання та обміну файлами і ін.

Зазвичай, статистичну обробку даних, зручніше писати на мові програмування R. Вона має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. R легко розбудовується завдяки використанню додаткових функцій і пакетів доступних на сайті Comprehensive R Archive Network (CRAN). Більша частина стандартних функцій R, написана мовою R, однак існує можливість підключати код написаний C, C++, або Фортраном. Також за допомогою програмного коду на C або Java можна безпосередньо маніпулювати R об'єктами. [29]

2.3.2 Реалізація та тестування основних класів програмного забезпечення

Найбільшої популярності набули дві стратегії тестування програм.

Перша, названа стратегією «чорного ящика», є тестуванням з управлінням по даним, або тестуванням з управлінням по входу-виходу. При використанні цієї стратегії програма розглядається як чорний ящик. Таке тестування має на меті з'ясування обставин, у яких поводження програми не відповідає її специфікації.

Тестові ж дані використаються тільки у відповідності зі специфікацією програми (тобто без урахування знань про її внутрішню структуру). При такому підході виявлення всіх помилок у програмі є критерієм вичерпного вхідного тестування. Останнє може бути досягнуто, якщо в якості тестових наборів використовувати всі можливі набори вхідних даних.

Стратегія «білого ящика», або стратегія тестування, що управляє логікою програми, дозволяє досліджувати внутрішню структуру програми. У цьому випадку особа одержує тестові дані шляхом аналізу логіки програми. Цей метод характеризується ступенем, у який тести виконують або покривають логіку (вихідний текст) програми. Вичерпне тестування за принципом білого ящика припускає виконання кожного шляху в програмі.

Існує декілька методів тестування, що відносяться до цієї стратегії: покриття операторів, покриття рішень, покриття умов, покриття рішень/умов, комбінаторне покриття умов. Для надійного тестування програмного забезпечення необхідно виконати тестування логіки кожної окремої процедури, модуля та перевірити їхню взаємодію.

Так як методи структурного тестування вимагають більших затрат ресурсів, то в дипломній роботі виконувалось тестування за специфікаціями, а саме - розбиття на класи еквівалентності.

Програму розглядали як чорний ящик. Випробування зводилися до послідовного вводу тестових наборів даних й аналізу отриманих результатів. В програмі користувач вводить дані вручну лише на одному етапі роботи ПЗ: на етапі розрахунку нормалізованих параметрів (необхідно) ввести чотири параметри для початкового наближення.

В таблиці 2.3 представлені класи еквівалентності та граничні значення для них даних: Параметр 1; Параметр 2; Параметр 3, Параметр 4.

Таблиця 2.3 - Класи еквівалентності вхідних даних для класу «Нормалізація вибірки»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Параметр 1	Чисельний	Символьні, логічні, графічні зображення
Параметр 2	Чисельний	Символьні, логічні, графічні зображення
Параметр 3	Чисельний	Символьні, логічні, графічні зображення
Параметр 4	Чисельний	Символьні, логічні, графічні зображення

Почнемо з тестування для правильних класів еквівалентності класу нормалізація», наведених в табл. 2.4.

Таблиця 2.4 - Тести з правильних класів еквівалентності для класу «Нормалізація вибірки»

№	Вхідні умови	Правильні класи еквівалентності	Вихідні дані
1	Параметр 1	Введенно число «70000,21»	Результат вірний
2	Параметр 2	Введено число «883»	Результат вірний
3	Параметр 3	Введено число «4,19»	Результат вірний
4	Параметр 4	Введено число «0,0031»	Результат вірний

Таблиця 2.5 - Тести неправильних класів еквівалентності класу «Нормалізація вибірки»

	Вхідні дані	Вихідні дані
1	В поле «Параметр 1» введемо/скопіюємо символ ASCII	Символ не вставляється. Програма далі працює правильно. Результат підтверджено
2	В поле «Параметр 2» введемо графічний елемент - (скопіюємо з будь-якого джерела).	Графічний елемент не вставляється. Програма далі працює правильно. Результат підтверджено

3	Поле «Параметр 3» залишимо порожнім	Повідомлення про помилку вводу. Програма далі працює правильно. Результат підтверджено
4	В поле «Параметр 4» введемо/скопіюємо символ ASCII	Символ не вставляється. Програма далі працює правильно. Результат підтверджено

Згідно результатів тестування неправильних класів еквівалентності класу «Нормалізація вибірки», програмне забезпечення працює правильно, не допускаючи до введення будь-яких інших елементів, окрім стандартних символів вводу. Варто зазначити, що тестування інших класів виконується аналогічно, тому сенсу приводити їх в даному пункті Робочого проекту немає.

2.3.3 Випробування програмного забезпечення

Під час випробувань було проведено повне функціональне тестування, також навантажувальне тестування і тестування на відмову всього ПЗ. Випробування програми було виконано згідно Додатку Д - «Програма та методика випробувань».

Випробування ПЗ проводилося на цільовому обладнанні (комп’ютері) в тій конфігурації, що заявлена в Додатку А - Технічне завдання. Всі виявлені недоліки ПЗ були зафіксовані та усунуті розробником до моменту впровадження ПЗ в дію. Випробування було проведено за стратегією «чорного ящика». Представимо деякі функції для тестування ПЗ. В повному обсязі дану інформацію і результати тестування можна у Додатку Д - «Програма та методика випробувань»

1) Функція «Імпорт файлу з вхідними даними».

На головному вікні зайшли в меню «Файл» - «Обрати файл». Відкривається вікно вибору файлу з файлової системи. Для тестування було обрано звичайний текстовий файл *.txt. З’явилось вікно з повідомленням про недопустимий формат файлу з вхідними даними.

1.1) Після натискання кнопки «ОК» знову відкривається діалогове вікно з можливістю вибору файлів з файлової системи. Обираємо файл 1.csv. В

головному вікні з’являється надпис: «Обрано файл: D:/1.csv» та стає активною кнопка «Перевірка коректності».

1.2) Після натискання кнопки «ОК» знову відкривається діалогове вікно з можливістю вибору файлів з файлової системи. Натискаємо «Відміна». Напису про обраний файл у вікні програми немає. Усі кнопки неактивні.

Функція працює правильно.

2) Функція «Перевірка коректності»

Після вдалого імпорту файлу з вхідними даними, натискаємо кнопку «Перевірка коректності». У разі відповідності даних вимогам початкових даних, з’являється надпис про вдале проходження перевірки та стає ативною кнопка «Перевірка на нормальність».

У разі некоректних початкових даних (таких, що не відповідають вимогам), з’являється повідомлення про необхідність зміни вхідних даних. Кнопки «Перевірка на нормальність» та «Нормалізувати дані» залишаються неактивними.

Функція працює правильно.

3) Функція «Перевірка вихідних вибірок на нормальність розподілу»

На головному вікні після завантаження файлу та розрахунку початкових параметрів вихідних вибірок натиснули кнопку «Перевірка на нормальність». Отримали параметри χ^2 (розрахункового та критичного), а також повідомлення про ненормальність розподілу вихідних вибірок. Кнопка «Нормалізувати дані», поле вибору к-сті параметрів та поля вводу параметрів нормалізації стають активними.

Функція працює правильно.

4) Функція «Нормалізувати дані»

Вводимо параметри розподілу до відповідних полів. Натискаємо кнопку «Нормалізувати дані». Отримали результати: кількість значень, середнього, дисперсії, середньоквадратичного відхилення, асиметрії, квадрату асиметрії, ексцесу. Дані розраховані і збережені до зовнішнього файлу.

Функція працює правильно.

5) Функція «Розрахувати параметри»

Після нормалізації дані зберігаються до зовнішнього файлу. Для його зчитування необхідно натиснути кнопку «Розрахувати параметри». Дані розрахунків показників зчитуються з зовнішнього файлу та відображаються під кнопкою. Функція працює правильно.

2.4 Удосконалення математичної моделі для оцінювання розміру програмного забезпечення

Вихідні дані для оцінювання розміру ПЗ, як правило, не мають нормального розподілу, що в результаті призводить до помилок у розрахунках та негативно впливає на достовірність отриманих результатів, у нашому випадку, на оцінювання розміру ПЗ. Щоб уникнути цієї проблеми, перед тим, як будувати математичну модель, потрібно нормалізувати вихідні дані. Для побудови нелінійного регресійного рівняння використовуємо вибірку шестивимірних негаусових даних з фактичний розмір ПЗ в тисячах рядків коду, загальна кількість класів x_1 , загальна кількість зв'язків x_2 та інші у концептуальній моделі даних з 54 одиниць ПЗ з відкритим кодом веб–застосунків, написаних з використанням фреймворку Django і взятих з репозиторіїв та Open-Source проектів: GitHub, SourceForge, AwesomeOpenSource.

Оскільки досить велика кількість з усіх 54-х програмни проектів призначені для платформи контейнеризації Docker, то для визначення їх метрик недостатньо скопіювати проект з репозиторію – необхідна інтеграція з платформою. [30] Фактори обирались за похідними метриками LOC/SLOC, що найбільш підходять під характеризацію коду, написаного на основі похідного фреймворку від мови програмування Python - Django. Варто зазначити, що обрані фактори характеризації коду частково притаманні для веб-застосунків на інших мовах програмування, таких як, наприклад, Java.

Використання нормалізуючих перетворень дозволяє перейти до лінійної регресії для нормалізованих даних, для неї побудувати довірчий інтервал та

інтервал прогнозування традиційним способом і, шляхом застосування відповідного зворотного перетворення, перейти до реалізації нелінійної регресії. Тим самим ми отримаємо більш точну математичну модель для оцінки розміру веб–застосунків, написаних з використанням фреймворку Django. [1]

2.5 Рівняння множинної регресії

Більшість соціально-економічних показників формується під впливом не одного, а багатьох факторів. Метод побудови моделі такого зв'язку має назву багатофакторного кореляційно-регресійного аналізу. В цьому випадку результативна ознака (*Y)* пов'язується з допомогою рівняння множинної регресії з декількома факторними ознаками *(x_1,...,x_m)*. Найважливішими умовами побудови багатофакторної моделі є достатня кількість одиниць у сукупності (як мінімум у 5 разів більше, ніж число факторів) та відсутність мультиколінеарності факторів (близького до функціонального зв'язку між ними). В тому випадку, якщо два факторних показники мультиколінеарні, один з них повинен бути виключений з моделі. [31]

Для побудови цього рівняння були використані дані з 54 програмних продуктів, кількість яких фактично вказує на невелику вибірку.

Рівняння множинної регресії відображає кореляційний зв'язок результативної (залежної) змінної *y* і декількох уточнюючих (незалежних) змінних $x_1,x_2,..,x_6$:

$$y = f(x_{1,}x_{2,}x_{p,}\varepsilon) \qquad (2.1)$$

де *y* - результативна змінна (залежна, яка пояснюється); $x_1,x_2,..,x_6$ - пояснюючі змінні (незалежні); ε - випадковий залишок; f – певна математична функція.

Якщо в якості опції в формулі (2.1) обрана лінійна, рівняння регресії називається рівнянням множинної лінійної регресії і має вигляд

$$Y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_p x_p + \varepsilon \qquad (2.2)$$

де $\alpha_0,\alpha_1,\ldots,\alpha_p$ - параметри.

У ряді випадків зручніше користуватися матричної записом цього рівняння

$$Y=X*\alpha\pm\varepsilon, \quad (2.3)$$

де X - матриця значень незалежних змінних

$$X=\begin{pmatrix}1 & x_{11} & \ldots & x_{p1}\\ 1 & x_{12} & \ldots & x_{p2}\\ \ldots & \ldots & \ldots & \ldots\\ 1 & x_{1n} & \ldots & x_{pn}\end{pmatrix} \quad (2.4)$$

Перший стовпець цієї матриці складається з одиниць, які розглядаються як значення додаткової змінної, на яку множиться вільний член. У матрицю X входить p незалежних змінних, що приймають n значень.

Y, α, ε - матриці-стовпці значень залежної змінної (матриця Y), параметрів регресії (матриця α) і випадкових залишків (матриця ε):

$$Y=\begin{pmatrix}y_1\\ y_2\\ \ldots\\ y_n\end{pmatrix};\ \alpha=\begin{pmatrix}a_0\\ \alpha_1\\ \ldots\\ \alpha_p\end{pmatrix};\quad \varepsilon=\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \ldots\\ \varepsilon_n\end{pmatrix} \quad (2.5)$$

Вектори Y і ε включають в себе по n значень залежної змінної і випадкових залишків, вектор a складається зі значень $(p+1)$ параметра - вільного члена α_0 і p коефіцієнтів регресії α_j.

Позначення Y - число рядків коду програми; x_1 - загальна кількість класів; x_2 - середня кількість атрибутів; x_3 - середня кількість методів; x_4 - середня кількість методів get; x_5 - середня кількість методів post; x_6 - середня кількість конструкторів; x_7 - кількість відношень між класами.

Статичний аналіз коду це процес виявлення помилок і недоліків у вихідному коді програм. Статичний аналіз можна розглядати як автоматизований процес огляду коду. Рішенням із збору статистичного матеріалу стали такі програмні продукти як: Prometheus [32], coveralls-python

[33], Radon [34], що статистично обробляють задані вихідні тексти програм і надають вичерпну інформацію по метрикам програм після аналізу.

Вибірка веб–застосунків на основі Django-фреймворку перевірена на викиди методом Евклідової відстані та обмежена 27 (з початкових 54 одиниць) програмами, дані яких наведені в таблиці 2.6. При дослідженні також розглядалися різні види рівняннь регресії. За коефіцієнтами детермінації було обрано експоненціальну регресію і надалі досліджувалося рівняння, в якому залежна змінна нормалізовувалася за основою натурального логарифму:

$$\ln y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 + \alpha_5 x_5 + \alpha_6 x_6 + \alpha_7 x_7 \quad (2.6)$$

Для побудови нелінійного регресійного рівняння використовуємо вибірку шестивимірних даних з фактичним розміром ПЗ в тисячах рядків коду. Рівняння множинної регресії може бути представлено у вигляді (2.7):

$$Y=X*\alpha\pm\varepsilon, \quad (2.7)$$

де $X = X(x_1, x_2, ..., x_m)$ - вектор незалежних (пояснюють) змінних;

α - вектор параметрів (що підлягають визначенню);

ε - випадкова помилка (відхилення);

Y - залежна змінна. [35]

Таблиця 2.6 – Вибірка 27 програм за 7 факторами

Назва ПЗ	Y	$Ln(y)$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Wooey	490	7.84	6	7	5.5	1	0.9	1	2.75
ci-guide	502	7.84	7	2.75	4	1	0.2	1	2.75
Gradebook	514	7.83	8	5	5.7	1.89	0.9	1	2.68
DetectDojo	532	7.82	3	6.58	7.33	0.33	0.14	1.02	2.81
cicd-tutorial	579	7.81	7	2	5.5	1.91	0.56	1	2.45
labhamster	580	7.81	8	2.89	2.26	0.44	0	1	2.45

Назва ПЗ	*Y*	*Ln(y)*	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Sphene Community	611	7.79	8	3	5.56	2	0.43	1	2.61
Wger	752	7.73	8	3	5.5	1.1	1	1.07	2.61
Gerapy	776	7.72	10	4.26	4.25	1.3	0.4	1	2.54
the-calendar	853	7.69	11	3.55	7.27	0.82	0.73	1	2.61
python-gitlab	894	7.67	11	7.9	1.6	0.23	0	1	2.75
Picket	950	7.64	17	4.63	3	0.77	0.43	1.05	2.54
OnlineJudge	972	7.63	9	4.26	5.56	2	0.42	1.02	2.61
Django Oscar	1092	7.57	6	8	6.5	0.71	0.6	1	2.81
graphene-django	1242	7.49	8	4	3.94	0.59	0.44	1	2.54
Pongo2	1248	7.49	7	10	9.08	0.83	0.67	1	2.61
DjangoLOs	1254	7.48	9	11	9.08	1	0.68	1	2.61
Liked feed	1269	7.48	10	5.18	7	1	0.56	1.09	2.45
RestAuth	1473	7.35	11	7	6.5	0.59	0.42	1	2.54
django-gitlab-logging	1499	7.34	14	5	7.27	3.47	1.39	1.05	2.36
pyadselfservice	1627	7.25	14	4.93	8.29	1.93	0.57	1.21	2.61
CyberGateMgmtSystem	6202	8.06	101	6.11	3.2	1	0.4	1.05	3.70
Dj-Shop	8022	8.51	137	4.33	4.25	0.59	0.18	1.02	3.70
Django PM	10734	8.95	76	6.33	7.59	2.1	1.85	1.05	3.48
Wag Tail	10790	8.96	84	4.26	4.92	1.1	0.4	1.07	3.84
Web FinanceMgr	16022	9.47	219	2.77	5.5	0.75	0.67	1.1	4.60
Django Shop	17023	9.55	159	5.28	5.56	1	0.64	1.1	5.09

Перш ніж перейти до визначення знаходження оцінок коефіцієнтів регресії, необхідно перевірити ряд передумов МНК:

1) Математичне сподівання випадкового відхилення ε_i дорівнює 0 для всіх спостережень ($M(\varepsilon_i) = 0$).

2) Дисперсія випадкових відхилень ε_i постійна:

$D(\varepsilon_i) = D(\varepsilon_j) = S^2$ для будь-яких i та j.

3) Відсутність автокореляції.

4) Випадкове відхилення повинно бути незалежним від незалежних змінних.

5) Модель є лінійною щодо параметрів.

6) Відсутність мультиколінеарності. Між незалежними змінними відсутня сильна лінійна залежність.

7) Помилки ε_i мають нормальний розподіл. Виконання цієї причини важливе для перевірки статистичних гіпотез і побудови довірчих інтервалів. [35]

2.5.1 Оцінка рівняння регресії.

Визначимо вектор оцінок коефіцієнтів регресії. Відповідно до методу найменших квадратів, вектор s виходить з виразу: $s = (X^T X)^{-1} X^T Y$.

Таблиця 2.7 – Шість незалежних параметрів та нормалізований Y

Параметри							
#	X1	X2	X3	X4	X5	X6	Ln(Y)
1	6	7	5.5	1	0.9	1	7.84
2	7	2.75	4	1	0.2	1	7.84
3	8	5	5.7	1.89	0.9	1	7.83
4	3	6.58	7.33	0.33	0.14	1.02	7.82
5	7	2	5.5	1.91	0.56	1	7.81
6	8	2.89	2.26	0.44	0	1	7.81
7	8	3	5.56	2	0.43	1	7.79
8	8	3	5.5	1.1	1	1.07	7.73
9	10	4.26	4.25	1.3	0.4	1	7.72
10	11	3.55	7.27	0.82	0.73	1	7.69
11	11	7.9	1.6	0.23	0	1	7.67
12	17	4.63	3	0.77	0.43	1.05	7.64

Таблиця 2.8 – Добуток матриці $X^T X$

53	1702	313.6	267.68	58.74	28.44	54.94
1702	142476	10453.6	8139.98	1714.66	945.01	1796.61
313.6	10453.6	3192.8996	1494.5299	303.4257	159.9291	324.1441
267.68	8139.98	1494.5299	1565.9792	341.8118	166.8737	277.3236
58.74	1714.66	303.4257	341.8118	94.0912	41.117	60.9403
28.44	945.01	159.9291	166.8737	41.117	24.3676	29.5346
54.94	1796.61	324.1441	277.3236	60.9403	29.5346	57.0864

Таблиця 2.9 –X^TX має наступні відповідності:

$\sum n$	$\sum y$	$\sum x_1$	$\sum x_2$	$\sum x_3$	$\sum x_4$	$\sum x_5$	$\sum x_6$
$\sum y$	$\sum y^2$	$\sum x_1 y$	$\sum x_2 y$	$\sum x_3 y$	$\sum x_4 y$	$\sum x_5 y$	$\sum x_6 y$
$\sum x_1$	$\sum y x_1$	$\sum x_1^2$	$\sum x_2 x_1$	$\sum x_3 x_1$	$\sum x_4 x_1$	$\sum x_5 x_1$	$\sum x_6 x_1$
$\sum x_2$	$\sum y x_2$	$\sum x_1 x_2$	$\sum x_2^2$	$\sum x_3 x_2$	$\sum x_4 x_2$	$\sum x_5 x_2$	$\sum x_6 x_2$
$\sum x_3$	$\sum y x_3$	$\sum x_1 x_3$	$\sum x_2 x_3$	$\sum x_3^2$	$\sum x_4 x_3$	$\sum x_5 x_3$	$\sum x_6 x_3$
$\sum x_4$	$\sum y x_4$	$\sum x_1 x_4$	$\sum x_2 x_4$	$\sum x_3 x_4$	$\sum x_4^2$	$\sum x_5 x_4$	$\sum x_6 x_4$
$\sum x_5$	$\sum y x_5$	$\sum x_1 x_5$	$\sum x_2 x_5$	$\sum x_3 x_5$	$\sum x_4 x_5$	$\sum x_5^2$	$\sum x_6 x_5$
$\sum x_6$	$\sum y x_6$	$\sum x_1 x_6$	$\sum x_2 x_6$	$\sum x_3 x_6$	$\sum x_4 x_6$	$\sum x_5 x_6$	$\sum x_6^2$

Таблиця 2.10 – Обернена матриця $(X^TX)^{-1}$

8.90503	0.002605	-0.012814	-0.038199	0.014553	0.0520276	-8.4363268
0.002605	1.29216E-05	-1.88949E-06	2.36242E-05	0.000109	-0.000205	-0.003029
-0.012814	-1.88949E-06	0.0008	0.000187	0.001276	-0.001116	0.006158
-0.038198	2.36242E-05	0.000187	0.007685	-0.007805	-0.011376	0.011821
0.014553	0.000109	0.001276	-0.007804	0.065089	-0.047708	-0.031598
0.052026	-0.000205	-0.001116	-0.011376	-0.047708	0.188975	-0.028853
-8.436327	-0.003029	0.006157	0.011821	-0.031598	-0.028853	8.188253

Добуток матриць X^TY представляє наступні значення: *-4388.138506; 83.79903217; 84.28212237; 262.4879011; 496.7781763; -200.1578765; 2376.037369.*

Вектор оцінок коеффіцієнтів регресії дорівнює: $Y(X) = (X^TX)^{-1}(X^TY) =$

$=(6.2477; 0.3568; 0.0201; 0.0602; 0.854; -0.5213; 0.1445)$

Далі за (2.6) та отриманним вектором будуємо регресійне рівняння:

$$\ln(Y) = 6.25 + 0.36x_1 + 0.02x_2 + 0.06x_3 + 0.85x_4 - 0.52x_5 + 0.15x_6 \quad (2.8)$$

Значення множинного коефіцієнта детермінації R^2 і середньої величини відносної похибки ρ дорівнюють *0,848* та *0,052* відповідно. Значення відсотка прогнозування $\tau(0;0,05)=0.897$.

Вид мультиколінеарності, при якому факторні змінні пов'язані деякою стохастичною залежністю, називається частковою. Якщо між факторними змінними є високий ступінь кореляції, то матриця (X^TX) близька до виродженої, тобто, чим ближче до 0 визначник матриці межфакторной кореляції, тим сильніше мультиколінеарності факторів і ненадійніше результати множинної регресії).

Аналіз мультиколінеарності показує:

$det(X^TX)=1.56625E^{13} \geqq 0$.

Знайдемо парні коефіцієнти кореляції (табл.2.11):

Таблиця 2.11 – Матриця парних коефіцієнтів кореляції *R*:

	Y	x1	x2	x3	x4	x5	x6
Y	1.000	0.897	0.099	0.070	0.043	0.135	0.291
x1	0.897	1.000	0.035	-0.105	-0.108	0.035	0.296
x2	0.099	0.035	1.000	-0.167	-0.224	-0.076	-0.069
x3	0.070	-0.105	-0.167	1.000	0.573	0.526	-0.029
x4	0.043	-0.108	-0.224	0.573	1.000	0.591	0.025
x5	0.135	0.035	-0.076	0.526	0.591	1.000	0.048
x6	0.291	0.296	-0.069	-0.029	0.025	0.048	1.000

Таблиця 2.12 – Матриця міжфакторної кореляції R_{11}

без Y	x1	x2	x3	x4	x5	x6
x1	1.000	0.035	-0.105	-0.108	0.035	0.296
x2	0.035	1.000	-0.167	-0.224	-0.076	-0.069
x3	-0.105	-0.167	1.000	0.573	0.526	-0.029
x4	-0.108	-0.224	0.573	1.000	0.591	0.025

x5	0.035	-0.076	0.526	0.591	1.000	0.048
x6	0.296	-0.069	-0.029	0.025	0.048	1.000

Таблиця 2.13 – Обернена матриця кореляції $(R_{11})^{-1}$

1.1347667	-0.0204767	0.1024246	0.1755087	-0.1834146	-0.330328
-0.0204767	1.0701197	0.0997898	0.2511408	-0.1231602	0.0828589
0.1024246	0.0997898	1.6458991	-0.6147732	-0.5022551	0.0636358
0.1755087	0.2511408	-0.6147732	1.8869288	-0.775171	-0.0625987
-0.1834146	-0.1231602	-0.5022551	-0.775171	1.7209198	-0.0320373
-0.330328	0.0828589	0.0636358	-0.0625987	-0.0320373	1.1085968

Значення діагональних елементів менше значення критерію Фішера $F_{kp}=3,81456902$, що показує відсутність мультиколінеарності, однак існує необхідність більш детального дослідження.

2.5.2 Аналіз мультиколінеарності

Якщо факторні змінні пов'язані сильною функціональною залежністю, то говорять про повну мультиколлінеарність. У цьому випадку серед стовпців матриці факторних змінних X є лінійно залежні стовпці і, за властивостями визначників матриці, $det(X^TX=0)$.

Значення визначника $det\ (X^TX) = 1.56625E^{13} \geqq 0$.

Вид мультиколінеарності, при якому факторні змінні пов'язані деякою стохастичною залежністю, називається частковою. Якщо між факторними змінними є високий ступінь кореляції, то матриця (X^TX) близька до виродженої, тобто: $det(X^TX \geqq 0)$ (чим ближче до 0 визначник матриці міжфакторної кореляції, тим сильніше мультиколінеарність факторів і ненадійніше результати множинної регресії).

Якщо в матриці коефіцієнт кореляції $r_{xjxi} > 0.7$, то в даній моделі множинної регресії існує мультиколінеарність. У нашому випадку всі парні

коефіцієнти кореляції $| r | <0.7$, що говорить про відсутність мультиколінеарності факторів.

Аналіз першого рядка цієї матриці дозволяє зробити відбір факторних ознак, які можуть бути включені в модель множинної кореляційної залежності. Факторні ознаки, у яких $| r_{yxi} | <0.5$ виключають з моделі. Можна дати наступну якісну інтерпретацію можливих значень коефіцієнта кореляції (за шкалою Чеддока): якщо $| r |> 0.3$ - зв'язок практично відсутній; $0.3 \leq | r | \leq 0.7$ - зв'язок середній; $0.7 \leq | r | \leq 0.9$ - зв'язок сильний; $| r | > 0.9$ - зв'язок вельми сильний.

Перевіримо значущість отриманих парних коефіцієнтів кореляції за допомогою t-критерію Стьюдента. Коефіцієнти, для яких значення t-статистики по модулю більше знайденого критичного значення, вважаються значущими.

$$T_{табл}(27-6-1;0,025)=2,09 \qquad (2.9)$$

t-критерій Стьюдента для змінних: $t_{x1}=14,83392622$; $tx_2=1,895896608$; $t_{x3}=1,904733654$; $t_{x4}=1,239027186$; $t_{x5}=-0,292985145$; $t_{x6}=0,528363742$.

Таким чином, зв'язок між y та x_i є суттєвим. Більш об'єктивну характеристику щільності зв'язку дають приватні коефіцієнти кореляції, що вимірюють вплив на результат фактора x_i при незмінному рівні інших факторів.

Колінеарність - залежність між факторами. В якості критерію мультиколінеарності може бути прийнято дотримання наступних нерівностей:

$$r (x_j y)> r (x_k x_j); r (x_k y)> r (x_k x_j).$$

Для відбору найбільш значущих чинників x_i враховуються такі умови:

- зв'язок між результативною і факторними ознаками повинна бути вище міжфакторного зв'язку;

- зв'язок між факторами має бути не більше значення 0.7. Якщо в матриці є міжфакторний коефіцієнт кореляції $r_{xjxi}> 0.7$, то в даній моделі множинної регресії існує мультиколінеарність;

- при високому міжфакторному зв'язку для ознаки відбираються фактори

з меншим коефіцієнтом кореляції між ними.

У нашому випадку всі парні коефіцієнти кореляції | *r* | *<0.7*, що говорить про відсутність мультиколінеарності факторів. [36]

Найбільш повним алгоритмом дослідження мультиколінеарності є алгоритм Фаррара-Глобера [37]. З його допомогою тестують два види мультиколлінеарності:

1) Всіх факторів ($FG_{набл}$).

2) Кожного фактора з іншими (критерій Фішера).

Перевіримо змінні на мультиколінеарності методом Фаррара-Глобера по першому виду статистичних критеріїв (критерій "хі-квадрат").

Формула для розрахунку значення статистики Фаррара-Глобера:

$$FG_{\text{набл}} = -[\, n - 1 - \frac{2m+5}{6}]\ln(\det[R]) \qquad (2.10)$$

$$FG_{\text{набл}} = \text{ - [27-1- (2*6 + 5) /6] ln (0.050) = 69.41}$$

де $m = 6$ - кількість факторів, $n = 27$ - кількість спостережень, *det [R]* - визначник матриці парних коефіцієнтів кореляції *R*. Порівнюємо його з табличним значенням при $v = (m / 2) - 1 = 13$ ступенях свободи і рівні значущості α. Якщо $FG_{\text{набл}} > \chi^2_{табл}$, то в векторі факторів присутня мультиколінеарність.

$\chi^2_{табл}\ (13;\ 0.05) = 22.36203 < 69.41$ мультиколінеарність відсутня.

2) Перевіримо змінні на мультиколінеарності за другим видом статистичних критеріїв (критерій Фішера). [38]

Таблиця 2.14 – Обернена матриця (до табл. 3.7) R^{-1}:

6,55978	-5,9673	-0,7406	-0,9228	-0,6427	0,14514	-0,2101
-5,96726	6,56303	0,65325	0,9419	0,76018	-0,3155	-0,1392
-0,74062	0,65325	1,15374	0,20398	0,32371	-0,1396	0,10658
-0,92279	0,9419	0,20398	1,77571	-0,5244	-0,5227	0,09319

-0,64272	0,76018	0,32371	-0,5244	1,9499	-0,7894	-0,042
0,14514	-0,3155	-0,1396	-0,5227	-0,7894	1,72413	-0,0367
-0,2101	-0,1392	0,10658	0,09319	-0,042	-0,0367	1,11533

Обчислюємо *F-критерій Фішера:*

$$F_k = (d_{kk} - 1) * \frac{n-m}{m-1}, \quad (2.11)$$

де d_{kk} - діагональні елементи матриці. Розраховані значення критеріїв порівнюються з табличними при $v_1 = n\text{-}m$ і $v_2 = m\text{-}1$ ступенях свободи, і рівні значущості α. Якщо $F_k > F_{табл}$, то *k*-а змінна мультіколлінеарності з іншими.

$$v_1 = 27\text{-}6 = 21;\ v_2 = 6\text{-}1 = 5.$$

$$F_{табл}\ (21;\ 5) = 2.68 \quad (2.12)$$

$F_1 = 19{,}45923453 > F_{табл}$, змінна у мультиколінеарності з іншими

$F_2 = 19{,}47060843 > F_{табл}$, змінна x_1 у мультиколінеарності з іншими

$F_3 = 0{,}538084615 \leq F_{табл}$, змінна x_2 немультиколінеарна з іншими

$F_4 = 2{,}714988032 > F_{табл}$, змінна x_3 у мультиколінеарності з іншими

$F_5 = 3{,}324658829 \leq F_{табл}$, змінна x_4 немультиколінеарна з іншими

$F_6 = 2{,}534459082 \leq F_{табл}$, змінна x_5 немультиколінеарна з іншими

$F_7 = 0{,}403636445 \leq F_{табл}$, змінна x_6 немультиколінеарна з іншими.

2.5.3 Аналіз параметрів рівняння регресії

Перейдемо до статистичного аналізу отриманого рівняння регресії: перевірці значимості рівняння і його коефіцієнтів, дослідженню абсолютних і відносних помилок апроксимації. Для незміщенної оцінки дисперсії:

$\varepsilon = Y\text{-}Y(x) = Y - X*s$ виконаємо наступні обчислення.

Таблиця 2.15 – Незміщена похибка ε (абсолютна похибка апроксимації)

Y	Y(x)	ε = Y - Y(x)	ε^2	|ε : Y|	(Y-Yx)/Y
7,84	7,808714	0,03	0,001062	0,004156	0,004156
7,84	7,794783	0,04	0,001746	0,005332	0,005332
7,83	7,770731	0,06	0,003732	0,0078	0,0078

7,82	7,856474	-0,03	0,001013	0,004067	0,004067
7,81	7,706672	0,10	0,009803	0,012684	0,012684
7,81	7,737562	0,07	0,004585	0,008675	0,008675
7,79	7,760498	0,03	0,001028	0,004114	0,004114
7,73	7,760498	-0,03	0,000779	0,003609	0,003609
7,72	7,718595	0,00	1,17E-05	0,000443	0,000443
7,69	7,713099	-0,03	0,000665	0,003354	0,003354
7,67	7,732888	-0,06	0,004167	0,008419	0,008419
7,64	7,582811	0,06	0,003481	0,00772	0,00772
7,63	7,745388	-0,11	0,01304	0,014964	0,014964
7,57	7,817677	-0,25	0,06074	0,032552	0,032552
7,49	7,749485	-0,26	0,066911	0,034532	0,034532
7,49	7,774995	-0,29	0,082677	0,038402	0,038402
7,48	7,745388	-0,26	0,068275	0,034913	0,034913
7,48	7,706672	-0,23	0,053381	0,030906	0,030906
7,35	7,702086	-0,35	0,122023	0,047508	0,047508
7,34	7,622491	-0,29	0,082096	0,039058	0,039058
7,25	7,658423	-0,41	0,167737	0,056499	0,056499
8,06	8,291944	-0,23	0,05333	0,028648	0,028648
8,51	8,441931	0,07	0,005323	0,008568	0,008568
8,95	8,099203	0,85	0,722222	0,094964	0,094964
8,96	8,210852	0,75	0,55567	0,08323	0,08323
9,47	8,778106	0,69	0,481237	0,07324	0,07324
9,55	8,710895	0,84	0,697504	0,087488	0,087488

Середня помилка апроксимації складає: *2.874%*

Оцінка дисперсії: $s_e^2=(Y-Y(X))^T(Y-Y(X))=3.264$ (2.13)

Незміщенна оцінка дисперсії: *0.384*

Оцінка середньо квадратичного відхилення (стандартна похибка для оцінки *Y*): [39]

$$S = \sqrt{S^2} = \sqrt{0.384} = 0.619 \quad (2.14)$$

Знайдемо оцінку коваріаційної матриці вектору $k = S^2 * (X^TX)^{-1}$

Таблиця 2.16 – Коваріаційна матриця

6,5597813	-5,967263	-0,740622	-0,922787	-0,642724	0,145142	-0,21008
-5,967263	**6,563031**	0,653248	0,94186	0,760178	-0,315446	-0,13922
-0,740622	0,653248	**1,153738**	0,203976	0,323706	-0,139547	0,106578
-0,922787	0,94186	0,203976	**1,775711**	-0,524359	-0,522673	0,093189
-0,642724	0,760178	0,323706	-0,524359	**1,949903**	-0,789392	-0,04202

0,1451415	-0,315446	-0,139547	-0,522673	-0,789392	**1,724131**	-0,03669
-0,21008	-0,139223	0,106578	0,093189	-0,042015	-0,036686	**1,115325**

Дисперсії параметрів моделі визначаються співвідношенням $S^2_i = K_{ii}$, тобто це елементи які знаходяться на головній діагоналі:

S: 2.561; 2.562; 1.074; 1.333; 1.396; 1.313; 1.056

2.5.4 Розрахунок множинного коефіцієнту кореляції

Щільність спільного впливу чинників на результат оцінює індекс множинної кореляції. На відміну від парного коефіцієнта кореляції, який може приймати негативні значення, він приймає значення від 0 до 1. Чим щільніше фактичні значення y_i розташовуються щодо лінії регресії, тим менше залишкова дисперсія і, отже, більше величина $R_y (x_1, ..., x_m)$.

Таким чином, при значенні R близькому до 1, рівняння регресії краще описує фактичні дані і фактори сильніше впливають на результат. При значенні R близькому до 0 рівняння регресії погано описує фактичні дані і фактори чинять слабкий вплив на результат.

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

$$R = \sqrt{1 - \frac{3.264}{9.977}} = \sqrt{0.327} = 0.572 \qquad (2.15)$$

Коефіцієнт множинної кореляції: $R = 0.897$. Зв'язок між ознакою lnY і факторами X_i сильний. Коефіцієнт детермінації:

$$R^2 = 0.897^2 = 0.805. \qquad (2.16)$$

Для перевірки рівня значущості використовують F-критерій Фішера**.** При цьому обчислюють фактичну (що спостерігається) значення F-критерію, через коефіцієнт детермінації R^2, розрахований за даними безпосереднього

спостереження. За таблицями розподілу Фішера знаходять критичне значення F-критерію *($F_{кр}$).* Для цього задаються рівнем значущості α (зазвичай його беруть рівним 0,05) і двома числами ступенів свободи $k_1 = m$ *і* $k_2 = n-m-1$. [38]

Маємо: $k_1 = 6$ *і* $k_2 = n-m-1 = 27 - 6 - 1 = 20$. Табличне значення при цих ступенях свободи,

$$F_{kp}(6; 20) = 2.6$$

$$R^2 = 0.805 = 1 - \frac{detR}{detR_{11}} \quad (2.17)$$

де $detR=0.05$ – визначник матриці парної кореляції; $detR_{11}=0.33$ - визначник матриці міжфакторної кореляції.

Уточнене значення коефіцієнту детермінації $\overline{R^2}$.

$$F = \frac{R^2}{1-R^2} \cdot \frac{(n-m-1)}{m} = \frac{0.7146}{1-0.7146} \cdot \frac{32-6-1}{6} = 10.431$$

$$\overline{R}^2 = 1 - (1-R)\frac{n-1}{n-m-1} \quad (2.18)$$

$$\overline{R}^2 = 1 - (1-0.805)\frac{27-1}{27-6-1} = 1 - 0.195 * 1.3 = 0.747$$

$$F = \frac{R^2}{1-R^2} * \frac{n-m-1}{m} = \frac{0.805}{0.195} * \frac{20}{6} = 13.761$$

Оскільки фактичне значення $F=13.8 > F_{kp}=2.6$, то коефіцієнт детермінації статистично значимий і рівняння регресії статистично надійне (тобто коефіцієнти спільно значимі). [36,38]

2.6 Перевірка нормальності розподілу залишкової компоненти

*X*і - квадрат (критерій згоди Пірсона) є об'єктивною оцінкою близькості емпіричних розподілів до теоретичних. Використовується у тих випадках, коли необхідно встановити відповідність двох порівнюваних рядів розподілу - емпіричного і теоретичного, або двох емпіричних. При цьому порівнюються

частоти названих рядів розподілу, виявляються розбіжності між ними і визначається вірогідність цих розбіжностей.

Перевірка гіпотез про вид розподілу за Пірсоном. Перевіримо гіпотезу про те, що *X* розподілено по нормальному закону з допомогою критерію згоди Пірсона (табл. 2.10).

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - np_i)}{np_i}, \quad (2.19)$$

де p_i - ймовірність попадання в і-й інтервал випадкової величини, розподіленої по гіпотетичному (теоретичному) закону.

Визначимо кордон критичної області. Так як статистика Пірсона вимірює різницю між емпіричним і теоретичним розподілами, то чим більше її бачимо значення $K_{набл}$, тим сильніше аргумент проти основної гіпотези. Тому критична область для цієї статистики завжди правобічна: $[K_{кр}; +\infty]$.

Її границя $K_{кр} = \chi 2\ (k\text{-}t\text{-}1;\ \alpha)$ знаходимо за таблицями розподілу χ^2 і заданим значенням *s, k* (число інтервалів), *t = 2*.

$$K_{кр} = \chi 2\ (6\text{-}2\text{-}1;\ 0.05) = 7.8;\ K_{набл} = 1.54 \quad (2.20)$$

Спостережуване значення статистики Пірсона не влучає у критичну область: $K_{набл} < K_{кр}$, тому немає підстав відкидати основну гіпотезу. Справедливо припущення про те, що дані вибірки мають нормальний розподіл. [40]

Перевіримо дане припущення, побудувавши гістограму за результатами розрахунків (рис.2.2).

Таблиця 2.17 – Значення інтервалів (гістограма)

	Перше зн.	кіл-ть
1	-3,8213	1
2	-3,13354	1

3	-2,44578	6
4	-1,75801	13
5	-1,07025	4
6	-0,38249	2

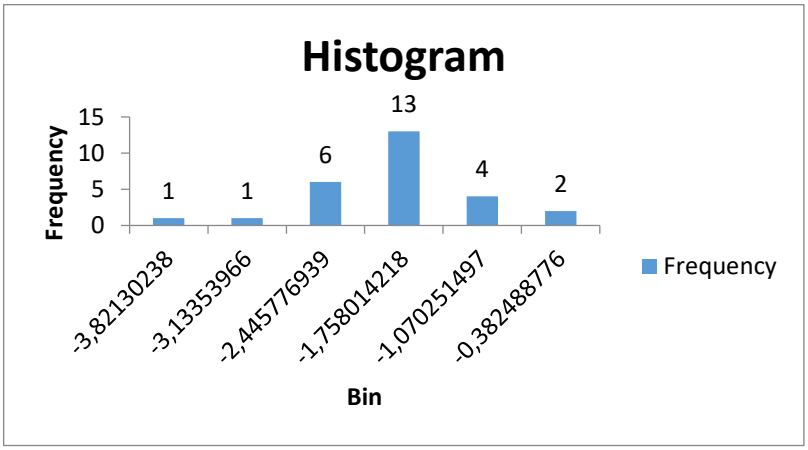

Рисунок 2.10 – Гістограма перевірки нормального розподілу

Побудована гістограма має ознаки нормального розподілу.

2.7 Довірчі та прогнозовані інтервали

Отримаємо з (2.3) нелінійну форму регресійного рівняння:

$$\hat{Y} = 516.8228 * e^{(0.32x_1+0.02x_2+0.06x_3+0.85x_4-0.52x_5+0.15x_6)} \quad (2.21)$$

$$\ln(Y) = 6.25 + 0.36x_1 + 0.02x_2 + 0.06x_3 + 0.85x_4 - 0.52x_5 + 0.15x_6$$

Для отриманого нелінійного рівняння $\hat{Y} = \omega_Y^{-1}\left[\overline{Z_Y} + (Z_X^+) * \hat{b}\right]$, будуємо довірчий та прогнозований інтервали з ймовірністтю 95%, відповідно $\alpha = 0{,}05$.

Індивідуальні довірчі інтервали для Y при даному значенні вектора $Z(\mathrm{x}_1, \ldots \mathrm{x}_6)$, де

$$\omega_Y^{-1}\left\{t_{\frac{\alpha}{2},v} * S * \sqrt{\frac{1}{n} + \left(\overline{Z_i}\right) * [R_{11}]^{-1} * (\overline{Z_l})^T}\right\} =$$

$$= 2.086 * 0.62 * \sqrt{\frac{1}{27} + \left(\overline{Z_i}\right) * [R_{11}]^{-1} * (\overline{Z_l})^T}; \quad (2.22)$$

де $t_{\frac{\alpha}{2},v}\left(\frac{\alpha}{2}; n-m-1\right) = t(0.025; 20) = 2.086;$

$S = 0.619$ – оцінка середньоквадратичного відхилення.

Індивідуальні інтервали прогнозування для Y при даному значенні вектора $Z(\mathrm{x}_1, \ldots \mathrm{x}_6)$,

$$\omega_Y^{-1}\left\{t_{\frac{\alpha}{2},v} * S_{Z_Y} * \sqrt{1 + \frac{1}{n} + \left(\overline{Z_i}\right) * [R_{11}]^{-1} * (\overline{Z_l})^T}\right\} =$$

$$= 2.086 * 0.62 * \sqrt{1 + \frac{1}{27} + \left(\overline{Z_i}\right) * [R_{11}]^{-1} * (\overline{Z_l})^T}. \quad (2.23)$$

$$S_{Z_Y}^2 = \frac{1}{n-m-1}\sum_{i=1}^{n}\left(\overline{Z_l} - \overline{Z_{sr}}\right)^2;$$

де $t_{\frac{\alpha}{2},v}\left(\frac{\alpha}{2}; n-m-1\right) = t(0.025; 20) = 2.086.$

Розрахунок довірчих та прогнозованих інтервалів зведено та представлено в таблиці Е.1 в Додатку Е.

З імовірністю 95% можна гарантувати, що значення Y при необмежено великому числі спостережень не вийде за межі знайдених інтервалів.

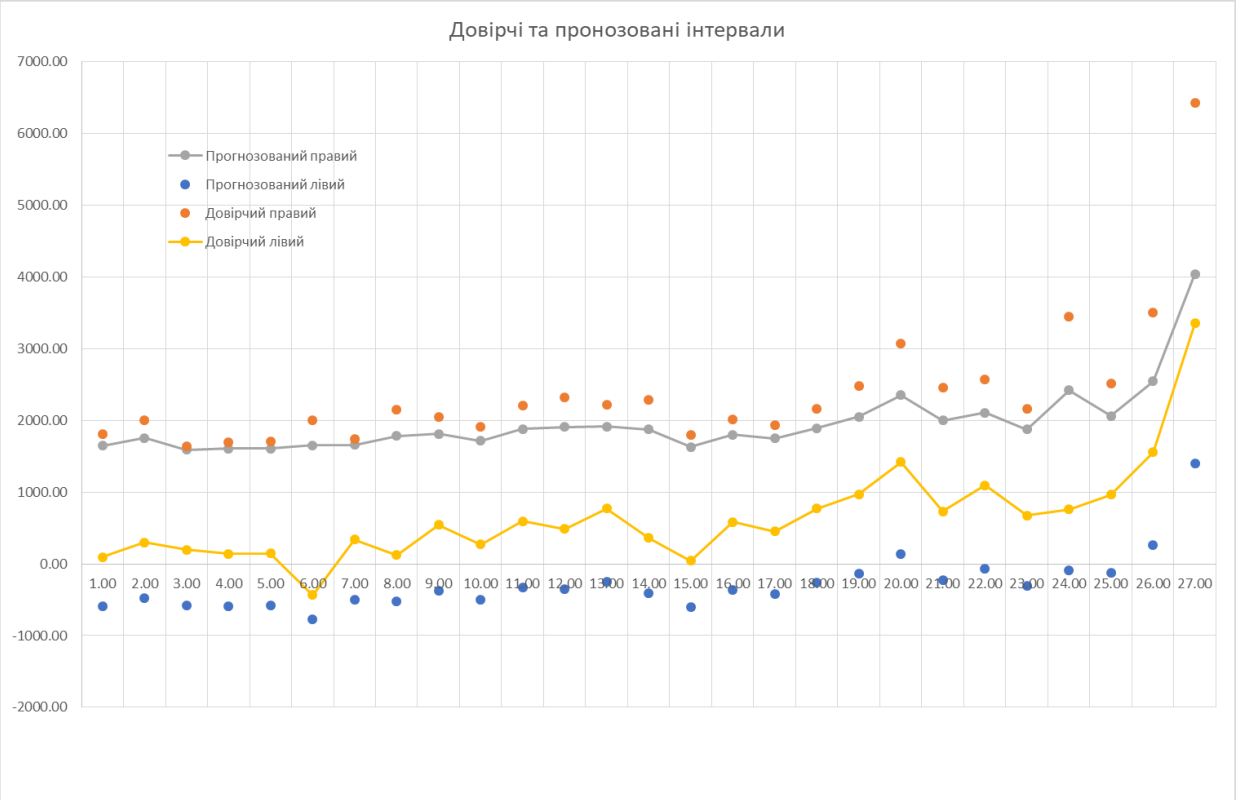

Рисунок 2.11 – Довірчі та прогнозовані інтервали для нелінійного рівняння

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Програмне забезпечення для оцінювання розміру коду веб-застосунків, написаних з використанням фреймворку Django» заснована на удосконаленій математичній моделі оцінювання розміру ПЗ за розрахунками, наведеними у Розділі 2 пояснювальної записки.

Нелінійне рівняння регресії та методика розрахунку довірчого інтервалу та інтервалу прогнозування нелінійного рівняння регресії дозволила підвищити достовірність оцінювання розміру веб–застосунків, написаних з використанням фреймворку Django.

Дане програмне забезпечення може використовуватись для оцінки програмних проектів програмістами, ентузіастами та організаціями, що пов’язані з розробкою програмних продуктів. Воно не потребує спеціальних знань та навичок й розраховане на звичайного користувача ПК з базовими знаннями предметної галузі.

Розроблене програмне забезпечення задовільняє вимогам, які поставлені в технічному завданні та випробувані під час етапу тестування. Створене ПЗ виконує наступні функції:

- імпорт файлу з вхідними даними у форматі *.csv;
- зчитування даних з файлу імпорту;
- перевірка даних на коректність;
- розрахунок параметрів для вихідних вибірок;
- нормалізація вихідних вибірок за допомогою логарифмічного перетворення;
- розрахунок параметрів для нормалізованих вибірок;
- перевірка вихідних та нормалізованих вибірок на нормальність розподілу;
- побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього.

Усі недоліки, які виникли на етапі тестування було зафіксовано та виправлено до моменту впровадження ПЗ в дію. Окрім цього було реалізовано ряд перевірок на коректність введених користувачем даних, а саме:

- перевірка на коректність формату файлу під час завантаження;
- перевірка на наявність файлу завантаження;
- перевірка на наявність введених початкових параметрів на етапі нормалізації;
- перевірка на послідовність дій користувача (деякі кнопки не є доступними, якщо не виконані попередні умови).

Інтерфейс був розроблений максимально зрозумілим та простим для користувача (user-friendly), кольорова гамма приємна та спокійна для очей, шрифт звичайний читабельний.

Також було реалізовано функцію завантаження файлу з розрахованими параметрами у форматі CSV.

Усі вимоги наведені у технічному завданні (див. Додаток А) були виконані. В якості технічних ресурсів був використаний ПК з комплектацією зазначеною у технічному завданні.

Розроблене програмне забезпечення дозволило автоматизувати обробку масивів статистичної інформації, пошуку і усунення "вузьких місць" у роботі, пов’язаної з оцінюванням розміру ПЗ, що призвело до скорочення часу відповідних розрахунків.

Для розробки програмного забезпечення для розміру коду веб-застосунків, написаних з використанням фреймворку Django, було обрано мову Python.

Для графічного зображення майбутнього ПЗ побудовано UML-діаграму варіантів використання. Потік подій варіанта використання має такі складові: короткий опис; передумови; основний потік подій; альтернативний потік подій;

постумови. Було розроблено специфікації варіантів використання.

У даному випадку програмне забезпечення, є продуктом такого типу структури, яке не потребує функціональності бази даних. Усі вхідні дані програма буде отримувати за допомогою імпортованого файлу та зчитування з нього інформації.

Інтерфейс користувача - це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. На рисунку 2.3 було представлено прототип інтерфейсу програмного забезпечення.

Діаграма класів - статичне представлення структури моделі. На побудованій діаграмі класів відображені наступні елементи: клас «Оцінювання розміру», клас «Нормалізація вибірки», клас «Регресія», «Нелінійне рівняння регресії», «Лінійне рівняння регресії». Вказано атрибути та їх типи, а також методи.

Динамічна модель ПЗ представлена діаграмами послідовності на рис.2.5-2.6. Потік подій варіанта використання має такі складові: короткий опис; передумови; основний потік подій; альтернативний потік подій; постумови.

За результатами тестування неправильних класів еквівалентності класу «Нормалізація вибірки», ПЗ працює правильно, не допускаючи до введення будь-яких інших елементів, окрім стандартних символів вводу. Тестування інших класів виконується аналогічно та не потребує додаткового опису.

Під час випробувань було проведено повне функціональне тестування ПЗ.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вступ

Найбільш важливим моментом при проектуванні програмного забезпечення для розробника, з економічної точки зору, є процес формування її майбутньої вартості. Створення програмного забезпечення вимагає одноразових витрат на її розробку, придбання необхідних технічних засобів, а також поточних витрат на функціонування продукту. Економія від функціонування програмного продукту визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень.

Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати. У зв’язку з актуальністю даного питання для студентів, що навчаються за напрямком «Програмне забезпечення автоматизованих систем» та з причини відсутності інформаційного забезпечення студентів з даної дисципліни, в рамках даної магістерської роботи виконується розробка програмного забезпечення для оцінювання трудомісткості коду веб-застосунків, що написані з використанням Django-фреймворку. Економічні показники визначаються діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

4.2 Розрахунок витрат на створення і експлуатацію програмного забезпечення

Витрати на розробку продукту складаються з витрат на заробітну платню розробника, на збірку (або закупівлю) комп’ютера для програмування, на експлуатацію ЕОМ, на засоби розробки та витрати на матеріали.

Розробка програмного забезпечення виконується програмістом, місячний оклад якого складає приблизно 8000 грн. Додаткова заробітна плата складає 20% від окладу.

Виходячи з вищевказаного, основна і додаткова заробітна плата розробника складає 9600 грн/міс , а вартість сучасного ПК - 13200 грн. (приведена вартість на базі Intel Core i5-10400). При вартості кіловат-години електроенергії рівної 1.68 грн. (згідно діючого тарифу) [41], розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	**Сума (грн.)**
Папір А4 (500 арк, упаковка, клас А)	120
Заправлення картриджа до принтера	160
CD-диск	15
Непередбачені витрати	300
Разом матеріали і комплектуючі	595

Вартість програми оцінювання розміру веб–застосунків на базі Django-фреймворку розрахуємо за наступною формулою:

$$C_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м},$$ де

$Т$ – тривалість розробки (міс);

$З_{зп}$ – основна і додаткова заробітна плата, грн.:

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

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

$З_{е}$ – витрати на електроенергію;

$З_{м}$ – витрати на основні і допоміжні матеріали;

$З_{е}$ при споживанні потужності 0,5 кВт, тривалості роботи за місяць, рівної 21*8=168 год., вартості кіловат-години електроенергії 1,68 грн складає:

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

Всі витрати на розробку ПЗ можна побачити в таблиці 4.2.

Таблиця 4.2 – Витрати на розробку ПЗ.

Найменування витрат	Одиниця	Кількість
Тривалість розробки	міс.	1,5
Основна і додаткова заробітна	грн.	9600
Відрахування на соціальні заходи	грн.	3072
Загальногосподарські витрати	грн.	800
Витрати на основні і допоміжні матеріали	грн.	595
Витрати на електроенергію	грн.	141,12

Відповідно вартість програми:

$$C_{пр} = (8000 + 3072 + 800 + 141,12) * 1,5 + 595 = 18614,68 \text{ грн.}$$

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

$$A_{об} = 13200 * 0,6 = 7920 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали (гнучкі диски, папір) визначаються в розмірі 5% вартості основного устаткування:

$$B_{м} = 13200 * 0,05 = 660 \text{ грн.}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_{м} = 264,5 * Тз$$

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

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи ПК складе:

$$Ф_{м} = 264,5 * 4 = 1058 \text{ годин}$$

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

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

Експлуатаційні витрати для ПК за рік складатимуть:

$$З_{зр} = 7920+660+888,72 = 9468,72 \text{ грн.}$$

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

$$З_{се} = 18614,68 + 9468,72 = 28083,40 \text{ грн.}$$

4.3 Економічна ефективність розробки і впровадження програмного забезпечення

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

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

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

—підвищені витрати на канцтовари;

—витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

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

—підвищення продуктивності праці;

—вивільнення робочого часу.

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

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

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

Зарплата 0,6 працівника в рік складає:

*8000 * 12 * 0,6 = 57600,00 грн.*

Річний економічний ефект розраховується по формулі:

$$Э_{год} = \Delta C_n - E_n * k ,$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту *57600,00 грн.)* мінус експлуатаційні витрати (*9468,72 грн.);*

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

k – одноразові витрати на впровадження продукту (*18614,68 грн*.).

$$Э_{год} = 57600,00 - 9468,72 – (0,6 * 18614,68) = 36942,47 \text{ грн.}$$

Строк окупності програми розраховується по формулі:

$$T = \frac{k}{\Delta C_n} = \frac{18614,68}{48131,28} = 0.39 \text{ (року)} \approx 5 \text{ (місяців)}$$

Отже строк окупності програмного продукту складає приблизно 5 місяців.

Проаналізувавши результати можна зробити висновок про доцільність розробки моделі якості ПЗ оцінювання розміру веб-застосунків на базі фреймворку Django. В даній роботі запропоновано використовувати методологію, що базується на розробленій моделі якості, специфічній для даного виду ПЗ.

Витрати на перший рік на створення й експлуатацію програми складуть: *28083,40 грн.* Строк окупності програмного продукту складає приблизно 5 місяців.

5 ОХОРОНА ПРАЦІ

На великих підприємствах станом на сьогодні діє система заходів, що спрямована на запобігання та попередження виникнення різних ситуацій, які можуть призвести до загрози життю і здоров’ю працівника. Дана система заходів називається охороною праці.

Під охороною праці розуміється сукупність способів, засобів і дій, спрямованих на скорочення в рамках підприємств або галузей травматизму, ситуацій, які можуть надати шкоду здоров’ю простого робітника.

Законодавство по охороні праці складається з дійсних Законів України «Про охорону праці», «Про охорону здоров’я», «Про пожежну безпеку», «Про забезпечення санітарного та епідеміологічного благополуччя населення», Кодексу законів про працю України та інших нормативних актів. У випадку, коли міжнародними договорами чи угодами, у яких бере участь Україна, установлені більш високі вимоги до охорони праці, чим ті, котрі передбачені законодавством України, застосовуються правила міжнародного договору чи угоди.

Визначення терміну «охорона праці» наведено в першій статті закону України «Про охорону праці». Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і заходів вкладених у збереження життя, здоров'я та перемоги працездатності людини під час праці. Закон України визначає основні напрямки по реалізації конституційного права на охорону життя і здоров'я в процесі трудової діяльності, регулюють при участі відповідних державних органів відносини між власником підприємства, чи установи організації, чи органом і роботодавцем з питань зовнішнього середовища і встановлює єдиний порядок організації охорони праці в Україні [42].

5.1 Аналіз небезпечних і шкідливих факторів у офісному приміщенні з персональними комп'ютерами

Цілком безпечних і нешкідливих виробництв не існує. Головна задача охорони праці - звести до мінімальної ймовірність ураження або захворювання працівника, а також виявлення і вивчення шкідливих факторів, їхній вплив на людину і навколишнє середовище.

Небезпечним виробничим фактором називається такий виробничий фактор, вплив котрого на працюючого у визначених умовах призведе до травми або до іншого раптового, різкого погіршення самопочуття. Прикладами небезпечних факторів можуть служити відкриті струмоведучі частини устаткування, що рухають деталі машин та інше. Прикладами шкідливих факторів являються шкідливі домішки в повітрі, несприятливі метеорологічні умови, шум, вібрації, недостатнє освітлення. [43]

Рівень безпеки є результатом взаємодії людини і того середовища, системи безпеки, що діє на виробництві. До факторів, які впливають на виконання виробничого процесу відносять мікроклімат приміщення, шум та вібрацію, освітлення, електробезпеку, пожежну безпеку, пил та інші.

Шумом є всякий небажаний для людини звук. Шум на виробництві завдає великої шкоди, шкідливо впливаючи на організм людини і знижуючи продуктивність праці. Шум навіть коли невеликий (при рівні 50-60 дБа), створює значне навантаження на нервову систему людини, роблячи на нього психологічний вплив. Під впливом шуму, що перевищує 85-90 дБа, у першу чергу знижується слухова чутливість на високих частотах.

Нормою виробничого шуму є рівень звуку до 85 дБ. Якщо рівень перешкод становить 20 дБ, то такий шум не заважає розбірливості мови. З підвищенням рівня перешкод до 70 дБ та вище мова стає нерозбірливою.

Джерелами шуму найчастіше є: виробниче устаткування для випікання, перемелювальні пристрої, двигуни, пневматичні та електричні інструменти, верстати, будівельна техніка тощо.

Небезпечним фактором для роботи програміста є також робота за комп’ютером. Найбільшому ризику піддаються зорова, опорно-рухова та нервово-психічна системи. Монітор випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювань розроблені гранично допустимі норми. Норми передбачають, що опроміненню піддається верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Відеоапаратура порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи загальний баланс атмосфери. Це також шкодить здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Ось чому необхідно, щоб до робочого місця за комп'ютером проникало свіже повітря.

Освітленість робочої зони комп'ютера повинна відповідати стандартам та бути вище яскравості монітора. Відстань від очей до екрана повинна бути не менше ніж півметра. Висота крісла має бути такою, щоб очі були на одному рівні з центром монітора. Фахівці підтверджують, що саме очі найбільш страждають при роботі з комп'ютером. Занадто відсунутий монітор призводить до перенавантаження м'язів голови та шиї, через що тиск на їх роботу зростає приблизно в три рази, судини шиї стискаються, погіршуючи кровопостачання до голови. Крім того, людині, що сидить у такій незручній позі, доводиться щоразу відкидати голову назад, для фіксування уваги на ближчих об’єктах. Це посилює вигин шийного відділу хребта. Через деякий час це може призвести до головної болі та болю у кінцівках, оскільки нерви, що відходять від спинного мозку в області шиї, простягаються до кінчиків пальців.

Малорухома та тривала сидяча поза за комп’ютером шкідлива для опорно-рухового апарату, що веде до застою крові в органах людини. Це особливо проявляється при фізіологічному положенні різних частин тіла і

тривало повторюваних одноманітних рухах. Під час роботи за комп'ютером людина сидить кілька годин поспіль в незручному становищі. Це не тільки загрожує втомою і загальним знесиленням організму, а й може призвести до розвитку остеохондрозу різних ділянок хребта - шийного, грудного, попереково-крижового.

Суттєвий вплив на стан організму ІТ-спеціаліста та його працездатність здійснює мікроклімат (метеорологічні умови) у відділі, під яким розуміють клімат внутрішнього середовища цих приміщень, що визначається діючою на організм людини сукупністю температури, вологи, руху повітря та теплового випромінювання нагрітих поверхонь. Несприятливі метеорологічні умови приводять до швидкої втоми працівника, частішу його захворюваністі та зниження продуктивності праці.

Світло впливає не лише на функції органів зору, а й на діяльність організму в цілому. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, зростає потенційна небезпека помилкових дій і нещасних випадків. Для створення оптимальних умов зорової роботи слід враховувати не лише кількість та якість освітлення, а й кольорове оточення. Так, при світлому пофарбуванні інтер’єру завдяки збільшенню кількості відбитого світла рівень освітленості підвищується на 20-40%, різкість тіней зменшується, покращується рівномірність освітлення.

При надмірній яскравості джерел світла та оточуючих предметів може відбутися засліплення працівника. Нерівномірність освітлення та неоднакова яскравість оточуючих предметів призводять до частої переадаптації очей під час виконання роботи, і як наслідок цього - до швидкого їх втомлення. Тому поверхні, що добре освітлюються і знаходяться в полі зору, краще фарбувати в кольори середньої світлості.

При освітленні ІТ-відділу використовують природне освітлення, що створюється світлом неба, штучне, здійснюване електричними лампами, і

сполучене, при якому у світлий час доби недостатній за нормами, природне освітлення доповнюється штучним. [44]

Для забезпечення здоров’я працюючих приймаються необхідні запобіжні міри захисту, що повністю або частково ізолюють доступ до зони, в якій діють шкідливі фактори, та виключають їх дію в разі проникнення людини у простір, де вони виникають. [45]

Велика кількість шуму негативно впливає на розумову роботу мозку, значно знижує продуктивність роботи програміста.

До методів боротьби із шумом відносяться:

- зменшення шуму в джерелі виникнення;
- зміна напрямку випромінювання шуму;
- акустична обробка приміщень;
- установка звукоізолюючих огороджень, кожухи, екрани, кабіни;
- установка глушителів шуму;
- використання засобів індивідуального захисту (вкладиші, навушники, шоломи).

Конструктивно вони можуть бути зроблені у вигляді гратчатих, сітчастих та непрозорих перешкод із металу, деревини тощо. Віброізоляція зменшує рівні вібрації, що передаються від джерела на тіло працюючого. Вона здійснюється введенням між джерелом вібрації та працюючим проміжного пружного зв’язку.

Заходи, щодо зниження негативного впливу мікроклімату:

1. впровадження раціональних технологічних процесів;
2. механізація та автоматизація виробничих процесів;
3. захист працівників різними типами екранів;
5. раціональна теплова ізоляція устаткування;
6. раціональне розміщення устаткування;
7. раціональне планування та конструкторське рішення виробничих приміщень;

8. раціональні вентиляція та опалювання;

Для збереження здоров’я очей, та запобіганню перевтоми, робоче приміщення спеціаліста ІТ-відділу повинно відповідати наступним вимогам:

- створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

- не повинно чинити засліплюючі дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

- забезпечити достатню рівномірність освітлення;

- не створювати на робочій поверхні тіней;

- повинно бути надійним в експлуатації, економічним та естетичним.

Робоче місце має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів:

1. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1 400мм, глибина – 800-1 000мм).

2. Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим.

3. Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку.

4. Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600-700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів. Розташування екрана

монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії погляду працівника.

5. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів.

6. Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають гігієнічний сертифікат.

На даний час, це поки що найреальніша можливість захистити людину, що працює біля комп'ютера від професійного захворювання.

Електрика широко застосовується у всіх галузях. Тому питанню електробезпечності необхідно приділяти велику увагу. Електробезпечність - система організаційних і технічних заходів та засобів, що забезпечують захист людей від шкідливого і небезпечного впливу електричного струму, електричної дуги, та інше. Проходячи через організм, електричний струм робить термічні, електролітичні і біологічні дії. Це різноманіття дій електричного струму нерідко приводити до різних електричних травм, що умовно можна звести до двох видів: місцеві електричні травми та загальні електричні травми.

Основні поразки струмом наступні:

- випадковий дотик або наближення на небезпечну відстань до струмопровідних частин, що знаходиться під напругою;

- поява напруги на металевих корпусних частинах електроустаткування;

- поява напруги на відключених струмоведучих частинах, на яких працюють люди, унаслідок помилкового включення установки;

Основними заходами захисту від поразки струмом є: забезпечення неприступності струмоведучих частин, що знаходяться під напругою, для

випадкового дотику; електричний поділ мережі; усунення небезпеки ураження з появою напруги на корпусах, кожухах і інших частинах електроустаткування, що досягається застосуванням малих напруг, використанням подвійної ізоляції, виявленням потенціалу, захисним заземленням, занулюванням, захисним відключенням і інше; застосування спеціальних електрозахисних засобів - переносних приладів і пристосувань; організація безпечної експлуатації електроустановок.

Занулюванням називається навмисне електричне з'єднання з нулем захисним провідником металевих не струмоведучих частин, що можуть виявитися під напругою. Призначення нульового захисного провідника - створення струмові короткого замикання ланцюга з малим опором, щоб цей струм був достатнім для швидкого спрацьовування захисту, тобто швидкого відключення ушкодженої установки від мережі.

Захисне відключення - швидкодіючий захист, забезпечує автоматичне відключення при виникненні в ній небезпеки ураження струмом. Така небезпека може виникнути, зокрема, при замиканні фази на корпус електроустаткування; при зниженні опору ізоляції фаз щодо землі нижче визначеної межі; появі в мережі більш високої напруги; дотик людини до струмоведучої частини, що знаходиться під напругою. Установлюють прилад захисного відключення ручний і автоматичний вимикач. Принцип дії - це швидке відключення від мережі установки, якщо напруга її корпусу щодо землі виявиться вище деякого гранично припустимого значення, унаслідок чого дотик до корпусу стає небезпечним.

Основним нормативним документом, що регламентує вимоги щодо пожежної безпеки є Закон України “Про пожежну безпеку”. Цей закон визначає загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулює відносини державних органів, юридичних і фізичних осіб у цій галузі незалежно від виду їх діяльності та форм власності. Пожежа - це неконтрольоване горіння поза спеціальним

вогнищем, що розповсюджується в часі і просторі та створює загрозу життю і здоров'ю людей, навколишньому середовищу, призводить до матеріальних збитків. Основними причинами пожежі на підприємстві є:

- небезпечне поводження з вогнем;

- незадовільний стан електротехнічних пристроїв та порушення правил їх монтажу та експлуатації;

- порушення режимів технологічних процесів;

- невиконання вимог нормативних документів з питань пожежної безпеки.

Власник підприємства, повинен:

- розробляти комплексні заходи щодо забезпечення пожежної безпеки;

- організувати навчання працівників правилам пожежної безпеки та пропаганду заходів щодо їх забезпечення;

- мають бути присутні засоби протипожежної безпеки.

Система протипожежного захисту - це сукупність організаційних заходів, а також технічних засобів, спрямованих на запобігання впливу на людей небезпечних факторів пожежі та обмеження матеріальних збитків від неї. На підприємстві у кожному цеху та кабінеті є по два вогнегасника.

У результаті проведеного дослідження можна зробити наступні висновки. Під час розумової або фізичної праці людина сприймає комплекс шкідливих виробничих чинників, які можуть позитивно або негативно відзначитись на її здоров'ї та продуктивності праці. Рівень цих факторів залежить від виду робочої діяльності, але можна сказати, що навіть робота в офісі може нашкодити здоров'ю працівника.

На великих підприємствах шкідливих чинників дуже багато, та їх дія безпосередньо негативно впливає на організм людини в цілому, та на окремі її органи. Постійний вплив цих факторів може призвести до професійних або хронічних захворювань.

Працівникам на потенційно небезпечному виробництві або з небезпечним обладнанням необхідно пам’ятати про засоби індивідуального захисту, правила поводження зі спеціальним обладнанням, адже від цього напряму залежить особисте здоров’я працівника.

Керівництву підприємства необхідно забезпечувати працівників та робітників необхідними засобами індивідуального захисту, вчасно слідкувати за станом робочих місць, що відповідають інструкції з техніки безпеки та не шкодить здоров’ю людини.

5.2 Розрахунок системи штучного освітлення у офісному приміщені з персональними комп’ютерами

У приміщені необхідно організувати змішане освітлення, тобто сполучення природного і штучного освітлення. Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітлені. У даному приміщені використовується загальне штучне освітлення. Розрахунок його здійснюється по методу світлового потоку з урахуванням потоку, відбитого від стін і стелі.

Довжина приміщення (А) = 5 м, ширина (В) = 4 м, висота (Н) = 2,5 м, висота робочої поверхні (h_p)= 1 м, для освітлення приймаємо світильник типу УПД, мінімальна освітлюваність лампи розжарювання за нормами E_{min}=100лк, коефіцієнт відображення стелі $\rho_п$ =70%, стін $\rho_с$ = 50%, робочої поверхні $\rho_р$=30%. Напруга мережі 210 В.

Розрахуємо відстань від стелі до робочої поверхні:

$$H_о = h – h_р = 2.5 – 1 = 1.5 \text{ м}, \quad (5.1)$$

де Н - висота приміщення, м;

h_p - висота робочої поверхні, м.

Визначимо відстань від стелі до світильника ($h_с$):

$$h_с= 0.2 * H_о = 0.2 * 1.5 = 0.3 \text{ м}, \quad (5.2)$$

Висота підвішування світильника над освітлюваною поверхнею (h):

$$h = H_o – h_c = 1.5 – 0.3 = 1.2 \text{ м}, \quad (5.3)$$

Висота підвішування світильника над підлогою ($H_п$):

$$H_п = h + h_p = 1.2 + 1 = 2.2 \text{ м}, \quad (5.4)$$

Для того, щоб досягти найбільшої рівномірності освітлення приймаємо відношення L/h = 1,5. Таким чином відстань між центрами світильників:

$$L = 1.5 * h = 1.5 * 1.2 = 1.8 \text{ м}.$$

Визначимо необхідну кількість світильників:

$$N = S/l^2 = 20 / 1.82^2 = 6.04, \quad (5.5)$$

приймаємо за 6 шт. Знаходимо індекс приміщення:

$$I = (A * B) / (h * (A+B)) = 20 / (1.2 * 9) = 1.85.$$

При i = 0,57, коефіцієнтах відображення стелі $\rho_п$=70%, стін ρ_c = 50%, робочої поверхні $\rho_р$=30% для світильника УПД коефіцієнт використання світлового потоку η = 0,28.

Світловий потік однієї лампи, лм:

$$\Phi = \frac{E_{min} \times S \times Z \times K_з}{N \times \eta \times \gamma}, \quad (5.6)$$

де E_{min} - рівень мінімальної освітленості за нормами, лк;

S - освітлювана площа приміщення, $м^2$;

$К_з$ - коефіцієнт запасу;

Z - коефіцієнт мінімальної освітленості;

N - число світильників;

η - коефіцієнт використання світлового потоку ламп, встановлених у світильнику.

Коефіцієнт запасу $К_з$ враховує експлуатаційне зниження освітленості, в порівнянні із запроектованою, внаслідок забруднення, а також зменшення світлового потоку ламп у процесі їх експлуатації. Для нашого приміщення коефіцієнт мінімальної освітленості ($К_з$) = 1,3.

Коефіцієнт мінімальної освітленості Z дорівнює відношенню середньої освітленості до мінімальної. Він враховує нерівномірність освітленості і залежить в основному від відношення відстаней між світильниками і від їх типів. Значення цього коефіцієнту для ламп розжарювання приймають Z =1,15.

$$\Phi = (E_{min} * S * K_З * Z) / (N* \eta) = (75* 20 * 1.3 * 1.15) / (6* 0.28) = 1334$$

лм.

За знайденим світловим потоком вибираємо лампу потужністю 200 Вт, що має світловий потік 3200 лм (Г125-135-200) найбільш близький до розрахункового.

При цьому фактична освітленість (лк) дорівнює:

$$E_ф = E_{min} \times \frac{\Phi_л}{\Phi_р}, \quad (5.7)$$

де $\Phi_л$ - світловий потік обраної лампи, лм;

$\Phi_р$ - світловий потік лампи, отриманої розрахунком,лм.

$$E_ф = 75* 3200/1334 = 179,91 \text{ лк.}$$

Загальна потужність освітлювальної установки $P_з$,Вт, визначається за формулою:

$$P_з = P_л \times N, \quad (5.8)$$

де $P_з$ - потужність обраної лампи,Вт.

$$P_з = 200 \times 6 = 1200 \text{ Вт} = 1,2 \text{ кВт.}$$

Загальна потужність освітлювальної установки становитиме при цьому 1200 Вт.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Основними нормативно-правовими актами України, спрямованими на забезпечення екологічної безпеки на її території, є Конституція, Закони „Про охорону навколишнього природного середовища”, „Про охорону атмосферного повітря”, „Про природно-заповідний фонд України” тощо, Земельний, Водний, Лісовий Кодекси, Кодекс України „Про надра”, постанови Кабінету Міністрів України в законодавстві передбачені права та обов’язки природокористувачів.

Закон України „Про охорону навколишнього природного середовища” [46] є основоположним законодавчим актом, визначає поняття екологічної безпеки та заходи щодо їх забезпечення, екологічні вимоги до розміщення, проектування будівництва, реконструкції, введення в дію підприємств та інших об’єктів, про застосування мінеральних добрив; передбачає заходи щодо охорони навколишнього природного середовища від шкідливого біологічного впливу; шкідливого впливу фізичних факторів та радіоактивного забруднення, від забруднення виробничими, побутовими та іншими відходами.

Даний закон не лише проголошує, але й передбачає систему гарантій екологічної безпеки людини, вносить певну впорядкованість в систему управління в галузі природокористування, він закріплює право громадян України на безпечне для життя навколишнє середовище. Це невід’ємне право реалізується шляхом участі громадян в обговоренні проектів законодавчих актів та інших рішень в галузі охорони навколишнього середовища; участі в розробці та здійсненні заходів щодо охорони природного середовища, раціонального використання природних ресурсів; об’єднання в громадські природоохоронні організації; отримання повної та достовірної інформації про стан навколишнього природного середовища.

Він надає громадянам України право звертатися до суду з позовом на підприємства, установи щодо відшкодування збитків, заподіяних здоров’ю та майну внаслідок негативного впливу на навколишнє середовище. Метою

екології є вивчення загальних закономірностей впливу антропогенної діяльності на навколишнє середовище, зокрема промисловості, сільського господарства, транспорту, комунального господарства тощо.

Охорону навколишнього середовища розглядають зазвичай як комплекс заходів міжнародного, державного, регіонального, локального рівнів, котрі спрямовані на збереження і забезпечення раціонального природокористування, відновлення, охорону та примноження природних ресурсів країни для блага людського суспільства і підтримання біологічної та екологічної рівноваги біосфери. Охорона довкілля включає джерела забруднення та їхній вплив на окремі екосистеми та біосферу з метою запобігання шкідливого впливу.

Найбільш раціональним заходом, спрямованим на зменшення забруднення атмосфери опалювальним обладнанням, є ліквідація пічних систем завдяки розвитку централізованого теплопостачання. При цьому внаслідок підвищення ККД котелень зменшується кількість спалюваного палива, а отже, і забруднення навколишнього середовища. Окрім того, при централізованому теплопостачанні в великих котельнях можливе очищення димових газів перед викидом їх в атмосферу.

Позитивне значення для розвитку не тільки централізації теплопостачання, але і каналізації, а також охорони природи може дати застосування колекторного прокладання інженерних комунікацій, - а саме - глибокого закладення. Таке прокладання здійснюється тунельним способом без зняття рослинного шару. Пошкодження рослин, порушення існуючих будівель і дорожнього покриття. А також велике значення для покращення екологічного стану повітряного басейну є відмова від використання вугілля в міських котельнях та перехід їх на природний газ.

Одним з потужних джерел забруднення міського повітря є автомобільний транспорт. У зв’язку з цим виникла необхідність розробки ряду заходів, що дозволяють запобігти забрудненню біосфери. Одним з таких заходів перехід автомобілів з бензиновими та дизельними двигунами на

електромобілі, що діють від під заряджених на станціях батареях – акумуляторах. Електромобілі мають кілька переваг: вони безшумні, бездимні та прості у використанні. Іншим засобом, який сьогодні найчастіше використовується, є встановлення на автомобілях фільтрів чи використання як палива природного газу, котрий в порівнянні з іншими видами палива менше забруднює повітря.

Щодо видалення побутових відходів, то останнім часом в Швеції почали застосовувати пневматичний транспорт для видалення сміття з сміттєпроводів по горизонтальних підземних каналах до станції, що надає послуги декільком будинкам (мікрорайон). В США, Великобританії, Італії та деяких інших країнах застосовується сплав в каналізацію посріблених відходів з квартир, готелів, ресторанів та інших об’єктів. З цією метою біля раковин ставлять механічні подрібнювачі, з котрих подрібнене сміття разом зі стічною водою видаляється в каналізацію, де воно знешкоджується в очисних установках. У нас в країні сміття збирається в контейнери та сміттєприймальні камери.

Для збору та тимчасового схоронності відходів організовують спеціальні площадки з твердим покривом, який забезпечує запобігти забрудненню ґрунту. Періодичність знешкодження накопичених відходів визначається згідно з діючими санітарними нормами та правилами та залежить від середньоденної температури повітря, при якій відбувається розкладання залишків органічних продуктів.

ВИСНОВКИ

Результатом дослідження в магістерській роботі є рівняння множинної регресії для оцінювання розміру коду веб-застосунків, написаних з використанням фреймворку Django та програмне забезпечення для реалізації поставленої задачі. Статистична значимість рівняння перевірена за допомогою коефіцієнта детермінації і критерію Фішера. Справедливим є припущення про те, що дані вибірки мають нормальний розподіл. Таким чином, модель є адекватною за нормальністю розподілу залишкової компоненти.

Отримані при аналізі метрики ПЗ кількісно визначають різні властивості програмних продуктів у вигляді чисельного відображення. Поставлена мета дослідження була у виведенні шістьох немультиколінеарних факторів особливостей ПЗ, що дало змогу порівнювати ці значення з подібними проектами, що мають специфічні стандарти.

Застосоване нормалізуюче логарифмування залежної змінної дозволило удосконалити лінійне рівняння регресії через нелінійне (експоненціальне) регресійне рівняння для оцінювання розміру веб-застосунків, написаних з використанням фреймворку Django.

Під час виконання роботи були вирішені такі завдання:

- проаналізовано та порівняно існуючі моделі для оцінювання розміру ПЗ;
- обґрунтовано необхідність удосконалення математичної моделі для оцінювання розміру ПЗ;
- досліджено та визначено веб-застосунки (на фреймворці Django), які були використані для перевірки побудованої моделі;
- побудовано діаграми класів та отримано необхідні метрики з кожного проекту;
- перевірено вихідні емпіричні дані на викиди;
- нормалізовано отримані емпіричні дані, використовуючи

нормалізуюче логарифмічне перетворення;

- побудовано лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для нормалізованих даних;
- побудовано нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для вихідних даних;
- розроблено ПЗ для оцінювання розміру веб-застосунків, написаних з використанням фреймворку Django.

Усі недоліки, які виникли на етапі тестування було зафіксовано та виправлено до моменту впровадження ПЗ в дію. Окрім цього було реалізовано ряд перевірок на коректність введених користувачем даних, а саме:

- перевірка на коректність формату файлу під час завантаження;
- перевірка на наявність файлу завантаження;
- перевірка на наявність введених початкових параметрів на етапі нормалізації;
- перевірка на послідовність дій користувача (деякі кнопки не є доступними, якщо не виконані попередні умови).

При порівнянні моделей отримали ближчий до одиниці коефіцієнт детермінаціїї: в лінійній моделі – **0.656,** в удосконаленій нормуванням **0.848**, $\boldsymbol{\tau}$**(0,05)** в лінійній моделі **0.172** в нелінійній моделі значення **0.805.** Тому можна сказати, що початкова модель удосконалена і ми отримали більш надійне рівняня для передбачення розміру програмного забезпечення.

Отримані при аналізі метрики ПЗ кількісно визначають різні властивості програмних продуктів у вигляді чисельного відображення. Поставлена ціль дослідження полягала у виведенні шістьох немультиколінеарних факторів особливостей ПЗ, що дало змогу порівнювати ці значення з подібними проектами, зі специфічними стандартамию З отриманих результатів можна прийти до висновку щодо прогнозування розміру ПЗ, його якості та всього програмного процесу, а також, якщо необхідно, планування необхідного часу для подальшої розробки.

Застосоване перетворення логарифмуванням за натуральною основою залежної змінної дозволило удосконалити лінійне рівняня регресії в нелінійне (експоненціальне) регресійне рівняння для оцінювання розміру веб–застосунків, написаних за допомогою Django-фреймворку.

Витрати на перший рік на створення й експлуатацію програми складуть: 28083,40 грн. Строк окупності програмного продукту складає приблизно 5 місяців.

Таким чином, мета роботи, щодо підвищення достовірності оцінювання розміру веб–застосунків на базі фреймворку Django та розробка програмного забезпечення для її реалізації, виконано в повному обсязі та проведено відповідні випробування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фреймворк Django [Електронний ресурс]. URL: https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov (дата звернення: 03.11.2019)

2. Аналіз функціональних точок [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Аналіз_функціональних_точок (дата звернення: 22.11.2019)

3. Модель COCOMO [Електронний ресурс]. URL: https://ru.wikipedia.org/wiki/COCOMO (дата звернення: 22.11.2019)

4. Дікопольцев І.О. Визначення метрик та довірчого інтервалу для побудови регресійного рівняння для оцінювання розміру веб-застосунків на базі фреймворка Django / Дікопольцев І.О., Кошкін В.К. // ІІІ Всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів «Сучасні інформаційні системи та технології», Херсон, 30 листопада 2020 року

5. Макарова Л.М. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою java / Л.М. Макарова, Н.В. Приходько, О.О. Кудін // Вісник ХНТУ. - 2019 р. - № 2(69) – С.145-154

6. Титов А. И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ / Титов А. И. // Intellectual Technologies on Transport. – 2016. – №1. –С.31-37

7. Мова програмування Python [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 03.11.2019)

8. Boehm B. Software Cost Estimation with Cocomo II. New Jersey, Prentice-Hall. 2000. 544 p

9. Титов А. И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ / Титов А. И. // Intellectual Technologies on Transport. – 2016. – №1. –С.31-37.

10. Prykhodko N. V. The non-linear regression model to estimate the software size of open source java-based systems / N. V. Prykhodko, S. B. Prykhodko

// Радіоелектроніка, інформатика, управління. - 2018. - № 3. - С. 158-166. - Режим доступу: http://nbuv.gov.ua/UJRN/riu_2018_3_19

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

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

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

14. Pardoe, Iain Applied regression modeling [Text] / Iain Pardoe. – Wiley, 2012. – 325 p.

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

16. Yan, Xin Linear regression analysis: theory and computing [Text] / Xin Yan, Xiao Gang Su. – Singapore: World Scientific Publishing Co. Pte. Ltd., 2009. – 328 p.

17. Айвазян, С. А. Прикладная статистика. Основы эконометрики: Учебник для вузов [Текст]: В 2 т. 2-е изд., испр. – Т. 1: Теория вероятностей и прикладная статистика / С. А. Айвазян, В. С. Мхитарян. – М.: ЮНИТИ-ДАНА, 2001. – 656 с.

18. Кобзарь, А. И. Прикладная математическая статистика. Для инженеров и научных работников [Текст] / А. И. Кобзарь. – М.: ФИЗМАТЛИТ, 2006. – 816 с.

19. Chatterjee, Samprit Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Simonoff. – Wiley, 2012. – 240 p.

20. Приходько С. Б. Інтервальне оцінювання статистичних моментів негаусівських випадкових величин на основі нормалізуючих перетворень [Текст]

21. Use Case [Електронний ресурс]. URL: https://systems.education/use-case (дата звернення: 25.12.2019)

22. UML. Сценарії варіантів використання [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Сценарій_використання (дата звернення: 25.12.2019)

23. Діаграма сутність-зв’язок [Електронний ресурс]. URL: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/latest/44197/index.html (дата звернення: 25.12.2019)

24. Діаграма діяльності [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення: 25.12.2019)

25. Моделювання форм Balsamiq [Електронний ресурс]. URL: https://balsamiq.com/ (дата звернення: 03.01.2020)

26. Діаграма класів [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Діаграма_класів (дата звернення: 03.01.2020)

27. Середовище програмування PyCharm [Електронний ресурс]. URL: https://itpro.ua/product/jetbrains-pycharm/?tab=description (дата звернення: 03.01.2020)

28. Рейтинг мов програмування серед розробників ПЗ за 2020 рік за версією Dou.ua [Електронний ресурс]. URL: https://dou.ua/lenta/articles/language-rating-jan-2020 (дата звернення: 05.01.2020)

29. Мова програмування R [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/R_(мова_програмування) (дата звернення: 05.01.2020)

30. Система контейнеризації Docker [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Docker (дата звернення: 05.01.2020)

31. Мультиколінеарність [Електронний ресурс]. URL: https://buklib.net/books/26193/ (дата звернення: 23.01.2020)

32. Проект для визначення метрик Prometheus [Електронний ресурс]. URL: https://github.com/prometheus/client_python (дата звернення: 01.02.2020)

33. Проект для визначення метрик Coveralls [Електронний ресурс]. URL: https://github.com/coveralls-clients/coveralls-python (дата звернення: 01.02.2020)

34. Проект для визначення метрик Radon [Електронний ресурс]. URL:, https://radon.readthedocs.io/en/latest/py-modindex.html (дата звернення: 02.02.2020)

35. Приходько, С. Б. Метод побудови нелінійних рівнянь регресії на основі нормалізуючих перетворень [Текст] : тези доп. міждерж. наук.-методич. конф. / С. Б. Приходько // Проблеми математичного моделювання. – Дніпродзержинськ: ДДТУ, 2012. – С. 31–33.

36. Множинна регресія і кореляція [Електронний ресурс]. URL: https://stud.com.ua/72639/ekonomika/mnozhinna_regresiya (дата звернення: 09.02.2020)

37. Алгоритм Фаррара-Глобера [Електронний ресурс]. URL: https://studfile.net/preview/5722599/page:2/ (дата звернення: 20.02.2020)

38. Критерій Фішера [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Критерій_Фішера (дата звернення: 02.03.2020)

39. Середньоквадратичне відхилення [Електронний ресурс]. URL: https://stud.com.ua/20681/statistika/serednye_kvadratichne_vidhilennya (дата звернення: 10.03.2020)

40. Критерій Пірсона [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Критерій_узгодженості_Пірсона (дата звернення: 19.03.2020)

41. Тариф на електроенергію (поточний) [Електронний ресурс]. URL: https://index.minfin.com.ua/ua/tariff/electric/ (дата звернення: 06.04.2020)

42. Закони України: «Про охорону праці»; «Про загальнообов’язкове державне соціальне страхування від нещасного випадку на виробництві і професійного захворювання, що спричинили втрату працездатності»; «Про забезпечення санітарного та епідемічного благополуччя населення»; Кодекс

цивільного захисту України; «Про дорожній рух»; «Про фізичний захист ядерних установок, ядерних матеріалів, радіоактивних відходів, інших джерел іонізуючого випромінювання». [Електронний ресурс]. – Режим доступу: http://zakon4.rada.gov.ua/laws

43. Жидецький В. Ц. Охорона праці користувачів комп'ютерів / В.Ц. Жидецький. – Львів: Афіша, 2000. – 176 с.

44. Безпека людини у життєвому середовищі: Навч. посібник / Голінько В.І., Шибка М.В., Безщасний О.В. За ред. В.І.Голінька.– 3-є вид., перероб. і доп. – Д.: Національний гірничий університет, 2004. – 187 с.

45. Безпека людини у надзвичайних ситуаціях: Навч. посібник / В.І.Голінько, С.О.Алексеєнко, М.Ф.Кременчуцький та ін.; За ред. В.І.Голінька. – 3-є вид., перероб. і доп. – Д.: Національний гірничий університет, 2004. – 160 с.

46. Закон України "Про охорону навколишнього природного середовища" N 1264-XII від 25 червня 1991 року.

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва програми: «Оцінка програмних проектів на базі Django-фреймворку».

Позначення програми: «Ocinka-Django».

Область застосування програми обмежена сферою веб-розробки програмного забезпечення.

1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на кваліфікаційну роботу магістра “Удосконалення математичної моделі для оцінювання розміру коду веб-застосунків, написаних з використанням фреймворку Django та розробка програмного забезпечення для її реалізації”.

2 Призначення програми

2.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою логарифмічного перетворення;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

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

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

2.2 Експлуатаційне призначення

Експлуатаційним призначенням програмного забезпечення є спрощення бізнес-процесів IT-компаній, покращення та легкість обробки масивів статистичних даних, полегшення оцінювання розміру веб-проектів.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу функцій, що виконуються

Програмне забезпечення повинно забезпечувати можливість виконування наступних функцій:

- можливість імпортувати файл з вихідними даними у форматі *.csv;
- можливість змінити файл з вихідними даними;
- зчитування даних з файлу та перевірка даних на коректність;
- розрахунок параметрів для вихідних вибірок;
- розрахунок параметрів логарифмічного перетворення;
- нормалізація вихідних вибірок;
- розрахунок параметрів для нормалізованих вибірок;
- перевірка вибірок на нормальність розподілу;
- побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;
- побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення вхідних даних здійснюється користувачем у відповідні поля діалогового вікна програмного забезпечення.

Вихідні дані (результат оцінювання, довірчий інтервал, інтервал передбачення) виводиться у діалоговому вікні програмного забезпечення.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

Вимоги, що забезпечують стійке функціонування програмного забезпечення повинні включати:

- Засоби перевірки введеної інформації (програмними засобами, повинні бути розроблені функції які контролюють введення інформації і не допускають ведення неповної чи невірної інформації, де це потрібно, і надають попередження чи підказки стосовно вірного введення інформації);

– Засоби, що забезпечують надійне збереження інформації (за рахунок використання операційної системи та створення резервних копій бази даних).

3.2.2 Відмова виконання програмного забезпечення через некоректні дії користувача

Відмова програмного забезпечення можлива внаслідок невірних дій користувача з операційною системою або середовищем виконання. Задля усунення вказаних відмов потрібно забезпечити стабільне функціонування операційної системи та середовища виконання програмного забезпечення.

3.3 Вимоги до умов експлуатації

Проектоване програмне забезпечення повинно функціонувати при наступних умовах її експлуатації:

– З програмним забезпеченням має працювати користувач, що ознайомлений з керівництвом користувача і має необхідні навички роботи з ПЕОМ в операційних системах Windows 10/8.x/7/XP.

– Умови експлуатації повинні відповідати вимогам до експлуатації апаратної частини ПЕОМ.

– Умови експлуатації носіїв інформації відповідають зазначеним вимогам у супровідній документації.

3.4 Вимоги до апаратного і програмного забезпечення

Система користувача повинна задовольняти вказаним мінімальним вимогам для коректного запуску:

- Процесор: Celeron N2840 2167 МГц і вище.
- Оперативна пам'ять: мінімум 2 Гб (рекомендується 4 Гб).
- Жорсткий диск: Для нормальної роботи системі потрібно 300 Мб вільного дискового простору.

Для програміста, розробника даного ПЗ, для подальшого вдосконалення системи та проведення тестів пропонується наступна конфігурація:

- Процесор: Intel Core i5-10400 2.9GHz

- Материнська плата: на базі чіпсету H410
- Оперативна пам’ять: 8-16 ГБ DDR4-2666
- Жорсткий диск: 1TB SATAIII 7200 rpm
- Твердотільний накопичувач (для ОС): 128 ГБ SATAIII 3D TLC.

3.5 Вимоги до документації

До програмної документації повинні входити наступні документи:

1) Технічне завдання;

2) Текст програми;

3) Опис програми;

4) Інструкція користувача;

5) Програма та методика випробувань.

Мова всієї документації – українська.

3.6 Вимоги до маркування та пакування

Вимоги до упаковки відсутні.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання відсутні.

4 Стадії та етапи розробки

Стадії та етапи розробки наведено в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадія створення ПЗ	**Етапи робот на стадії**	**Початок**	**Кінець**
Технічне завдання	Визначення призначення розробки та вимог	12.09.2019	22.10.2019
Ескізний проект	Прийняття принципових рішень по структурі ПЗ. Розробка концептуальної моделі даних.	23.10.2019	15.12.2019
Технічний проект	Вибір інструменту мови кодування. Розробка документації. Розробка логічної моделі даних.	16.12.2019	01.02.2020
Робочий проект	Виготовлення компонентів ПЗ. Розробка фізичної моделі даних.	02.02.2019	30.05.2019
Введення в дію	Дослідне функціонування ПЗ з метою перевірки дієздатності, корегування документації	01.06.2019	30.11.2020

5 Порядок прийому та контролю

Для контролю и прийому повинно бути надана інструкція користувача та опис програми.

Перевірка документації програми здійснюється представником замовника з метою зафіксувати факт відповідності (або невідповідності) створеного програмного забезпеченню всім пунктам технічної документації.

Порядок контролю и прийому даної розробки здійснюється представником замовника у присутності представника розробника згідно з програмою та методикою випробувань шляхом зіставлення характеристик системи з вимогами контракту.

За результатами прийому складається акт, який підписується представником замовника та представником розробника та затверджується керівниками організації-замовника і організації-розробника.

Якщо програма не пройшла випробування, складається акт про виявлені помилки та недоліки, що підписується представниками замовника та розробника, та протокол помилок, до якого вносять виявлені помилки. Виконавець зобов’язаний виправити помилки та недоліки у строк, не більш ніж 1 місяць з дня випробування та повідомити замовника про повторне проведенні перевірки не пізніше ніж 2 тижні до початку прийому програмного продукту. Після чого проводиться додаткове тестування тієї частини програми, де були виявлені помилки.

Після виправлення помилок або у разі відсутності таких і при відповідності програми встановленим вимогам підписується акт прийому програмного забезпечення.

ДОДАТОК Б - КОД ПРОГРАМИ

1) **Файл «main»**

```
import sys
from PyQt5 import QtWidgets
from controller import Controller

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    window = Controller()
    window.show()
    app.exec_()
```

2) **Функція розрахунку матриць**

```
import math
from copy import deepcopy

def get_column(array, column):
    result = []
    for rows in array:
        result.append(rows[column])
    return result

def remove_column(array, column):
    result = deepcopy(array)
    for i in range(result.__len__()):
        result[i].pop(column)
    return result

def remove_columns(array, columns):
    result = []
    for i in range(array.__len__()):
        result.append([val for j, val in enumerate(array[i]) if j not in
columns])
    return result

def add_ones_column(array):
    result = deepcopy(array)
    for i in range(array.__len__()):
        result[i] = [1] + array[i]
    return result

def ln_first_column(array):
    for i in range(array.__len__()):
        array[i][0] = math.log(array[i][0])

def matrix_size(matrix):
    rows = matrix.__len__()
    columns = matrix[0].__len__()
    return rows, columns

def print_2D(array):
    for row in array:
        for value in row:
            print(str(value), end=" ")
        print()
```

3) Файл «controller»

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QFileDialog, QTableWidget, QTableWidgetItem,
QCheckBox, QPushButton, QLineEdit, QLabel
import numpy as np
import view
from analyze.analyzer import Analyzer
from analyze.matrix import matrix_size

class Second(QtWidgets.QMainWindow):
    def __init__(self, parent, analyzer):
        super(Second, self).__init__(parent)
        self.setFixedHeight(500)
        self.setFixedWidth(800)
        self.table_widget = None
        self.display_table(analyzer._corr1)

    def display_table(self, matrix):
        matrix = np.round(matrix, 2)
        rows, columns = matrix_size(matrix)
        self.layout().removeWidget(self.table_widget)
        self.table_widget = QTableWidget()
        self.table_widget.setRowCount(rows)
        self.table_widget.setColumnCount(columns)
        for i in range(rows):
            for j in range(columns):
                self.table_widget.setItem(i, j,
QTableWidgetItem(str(matrix[i][j])))
        self.table_widget.setFixedWidth(self.width())
        self.table_widget.setFixedHeight(self.height())
        self.table_widget.move(0, 0)
        print(self.table_widget)
        self.layout().addWidget(self.table_widget)

class Third(QtWidgets.QMainWindow):
    def __init__(self, parent, analyzer):
        super(Third, self).__init__(parent)
        self.setFixedHeight(450)
        self.setFixedWidth(700)
        self.table_widget = None
        self.display_table(analyzer._corr2)

    def display_table(self, matrix):
        matrix = np.round(matrix, 2)
        rows, columns = matrix_size(matrix)
        self.layout().removeWidget(self.table_widget)
        self.table_widget = QTableWidget()
        self.table_widget.setRowCount(rows)
        self.table_widget.setColumnCount(columns)
        for i in range(rows):
            for j in range(columns):
                self.table_widget.setItem(i, j,
QTableWidgetItem(str(matrix[i][j])))
        self.table_widget.setFixedWidth(self.width())
        self.table_widget.setFixedHeight(self.height())
        self.table_widget.move(0, 0)
```

```
        print(self.table_widget)
        self.layout().addWidget(self.table_widget)

class Result(QtWidgets.QMainWindow):
    def __init__(self, parent, analyzer):
        super(Result, self).__init__(parent)
        self.setFixedHeight(200)
        self.setFixedWidth(900)
        self.table_widget = None
        self.display_table(analyzer)

    def display_table(self, analyzer):
        matrix = analyzer._reg[np.newaxis]
        matrix = np.round(matrix, 2)
        rows, columns = matrix_size(matrix)
        self.layout().removeWidget(self.table_widget)
        self.table_widget = QTableWidget()
        self.table_widget.setRowCount(rows)
        self.table_widget.setColumnCount(columns)
        for i in range(rows):
            for j in range(columns):
                self.table_widget.setItem(i, j,
QTableWidgetItem(str(matrix[i][j])))
        self.table_widget.setFixedWidth(self.width())
        self.table_widget.setFixedHeight(self.height())
        self.table_widget.move(0, 0)
        print(self.table_widget)
        self.layout().addWidget(self.table_widget)

        n = analyzer.active.__len__()
        mre = "{0:.4f}".format(analyzer.mre / n)
        pred = "{0:.4f}".format(analyzer.predk / n)

        labelMre = QLabel(self)
        labelMre.move(100, 50)
        labelMre.setText('MMRE: ' + mre)
        self.layout().addWidget(labelMre)

        labelL = QLabel(self)
        labelL.move(100, 70)
        labelL.setText('l: ' + str(analyzer.l))
        self.layout().addWidget(labelL)

        labelK = QLabel(self)
        labelK.move(100, 90)
        labelK.setText('k: ' + str(analyzer.predk))
        self.layout().addWidget(labelK)

        labelN = QLabel(self)
        labelN.move(100, 110)
        labelN.setText('n: ' + str(n))
        self.layout().addWidget(labelN)

        labelPred = QLabel(self)
        labelPred.move(100, 130)
        labelPred.setText('PRED: ' + pred)
```

```
        self.layout().addWidget(labelPred)

        labelR = QLabel(self)
        labelR.move(100, 150)
        labelR.setText('R^2: ' + str(analyzer.r))
        self.layout().addWidget(labelR)

class Controller(QtWidgets.QMainWindow, view.Ui_MainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)
        self.actionOpen.triggered.connect(self.open_file)
        self.table_widget = None
        self.textbox = None
        self.analyzer = None
        self.dialogs = list()

    def open_file(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        file_name, _ = QFileDialog.getOpenFileName(self, "Choose
statistic file", "", "Statistic file (*.txt)", options=options)
        if file_name:
            self.analyzer = Analyzer(file_name)
            self.input()

    def input(self):
        columns = self.display_table(self.analyzer.observations)

        names = ["x1", "x2", "x3", "x4", "x5", "x6", "x7", "Y"]
        for i in range(columns):
            checkbox = QCheckBox(names[i], self)
            checkbox.setChecked(True)
            checkbox.toggled.connect(lambda checked, v=i:
self.analyzer.disable(v, checked))
            checkbox.move(50 + 100 * i, self.height() - 75)
            self.layout().addWidget(checkbox)

        button = QPushButton('Regression', self)
        button.setToolTip('Make analyze')
        button.move(50 + 100 * columns, self.height() - 75)
        button.clicked.connect(self.next)
        self.layout().addWidget(button)

        label = QLabel(self)
        label.move(850, 550)
        label.setText('l:')
        self.layout().addWidget(label)

        self.textbox = QLineEdit(self)
        self.textbox.move(900, 550)
        self.textbox.resize(60, 30)
        self.textbox.setText("0.05")
        self.layout().addWidget(self.textbox)

    def checkbox_click(self, col, checked):
```

```
        if checked:
            self.analyzer.disables.add(col)
        else:
            self.analyzer.disables.remove(col)

    def display_table(self, matrix):
        matrix = np.round(matrix, 2)
        rows, columns = matrix_size(matrix)
        self.layout().removeWidget(self.table_widget)
        self.table_widget = QTableWidget()
        self.table_widget.setRowCount(rows)
        self.table_widget.setColumnCount(columns)
        for i in range(rows):
            for j in range(columns):
                self.table_widget.setItem(i, j,
QTableWidgetItem(str(matrix[i][j])))
        self.table_widget.setFixedWidth(self.width())
        self.table_widget.setFixedHeight(self.height() - 100)
        self.table_widget.move(0, 0)
        self.layout().addWidget(self.table_widget)
        return columns

    def next(self):
        if self.analyzer:
            self.analyzer.analyze(self.textbox.text())
            dialog1 = Second(self, self.analyzer)
            dialog2 = Third(self, self.analyzer)
            dialog3 = Result(self, self.analyzer)
            self.dialogs.append(dialog1)
            self.dialogs.append(dialog2)
            self.dialogs.append(dialog3)
            dialog1.show()
            dialog2.show()
            dialog3.show()
```

4) Файл «analyzer»

```
import math
from copy import deepcopy
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt
import copy

from analyze.matrix import print_2D, add_ones_column, remove_column,
get_column, remove_columns, matrix_size, \
    ln_first_column

class Analyzer:
    def __init__(self, file_name):
        self.observations = []
        self.active = []
        self.active_diff = []
        self.disables = []
        self.calculated = []
        self.freedom = 0
        self.student = 0
```

```
        self.fisher = 0
        self._corr1 = []
        self._corr2 = []
        self._reg = []
        self.mre = 0
        self.predk = 0
        self.r = 0
        with open(file_name, "r") as file:
            for line in file:
                values = [float(value) for value in line.split("\t")]
                self.observations.append(values)
        self.rows = self.columns = 0
        self.set_size(self.observations)
        ln_first_column(self.observations)

    def set_size(self, matrix):
        self.rows, self.columns = matrix_size(matrix)

    def disable(self, col, val):
        if val:
            self.disables.remove(col)
        else:
            self.disables.append(col)

    def analyze(self, l):
        self.l = float(l)

        self.active = remove_columns(self.observations, self.disables)
        self.set_size(self.active)

        self.calc_average()
        self.fit_by_average(1.0)
        self.set_size(self.active)

        self.active.sort()
        print_2D(self.active)
        regression_coef = self.regression()

        self.correlation()

        self.hist(regression_coef)

        self.set_size(self.active)
        self.criteria()
        self.intervals()

    def calc_average(self):
        self.active_diff = deepcopy(self.active)
        for column in range(self.columns):
            self.calc_average_diff(column)

    def calc_average_diff(self, column):
        s = 0
        mi = ma = self.active[0][column]
        for i in range(self.rows):
            v = self.active[i][column]
```

```
            s += v
            if v < mi:
                mi = v
            if v > ma:
                ma = v
        mid = s / self.rows
        for i in range(self.rows):
            self.active_diff[i][column] = abs(self.active[i][column] -
mid) / (ma - mi)

    def fit_by_average(self, limit):
        for i in range(self.rows - 1, -1, -1):
            s = sum(self.active_diff[i])
            if s > limit:
                self.active.pop(i)

    def regression(self):
        print("\nregression")
        x = add_ones_column(remove_column(self.active, 0))
        y = get_column(self.active, 0)
        xt = np.transpose(x)
        first = np.dot(xt, x)
        second = np.dot(xt, y)
        third = np.linalg.inv(first)
        coefficients = np.dot(third, second)
        print(coefficients)
        np.savetxt('regression.txt', coefficients, fmt='%.4f')
        self._reg = copy.deepcopy(coefficients)
        return coefficients.tolist()

    def correlation(self):
        print("\ncorrelation")
        a = np.array(self.active)
        a = a.transpose()
        matrix = np.corrcoef(a)
        matrix = np.linalg.inv(matrix)
        np.savetxt('correlation1.txt', matrix, fmt='%.4f')
        self._corr1 = copy.deepcopy(matrix)

        a = np.array(remove_column(self.active, 0))
        a = a.transpose()
        matrix = np.corrcoef(a)
        matrix = np.linalg.inv(matrix)
        np.savetxt('correlation2.txt', matrix, fmt='%.4f')
        self._corr2 = copy.deepcopy(matrix)

        self.r = 1 - np.linalg.det(self._corr2) /
np.linalg.det(self._corr1)

    def hist(self, regression_coef):
        print("\nhistogram")
        size = self.rows
        self.calculated = list()
        self.mre = 0
        self.predk = 0
        for i in range(0, size):
```

```
            s = regression_coef[0]
            for j in range(1, self.columns):
                s += self.active[i][j] * regression_coef[j]
            ds = s - self.active[i][0]
            self.calculated.append(ds)
            mrei = abs(ds / s)
            self.mre += mrei
            if (mrei < self.l):
                self.predk += 1
            # print(i, s, s - self.active[i][0])
        print(self.calculated)
        minv = min(self.calculated)
        maxv = max(self.calculated)
        step = (maxv - minv) / 6
        print("step =", step)
        counts = list()
        for q in range(0, 6):
            k = 0
            lower = q * step + minv
            upper = q * (step + 1) + minv
            for i in range(0, size):
                if (self.calculated[i] >= lower) and (self.calculated[i]
<= upper):
                    k += 1
            counts.append(k)
        print(counts)
        plt.bar(np.arange(6), height=counts)
        # plt.axis([minv, maxv, 0, max(counts)])
        plt.show()

    def criteria(self):
        print("\ncriteria")
        p = 0.05
        self.freedom = self.rows - self.columns - 1
        self.fisher = stats.f.ppf(q=1-p, dfn=self.columns,
dfd=self.freedom)
        self.student = 2.06
        # self.student = 1 - stats.t.cdf(0.05, df=self.freedom)
        print(self.fisher, self.student)

    def intervals(self):
        print("\nintervals")
        mean = math.exp(np.mean(get_column(self.active, 0)))
        a = np.array(remove_column(self.active, 0))
        x = remove_column(self.active, 0)
        xt = np.transpose(x)
        xt_x_r = np.dot(xt, x)
        xt_x = np.linalg.inv(xt_x_r)
        # print(xt_x)
        sqr = list()
        for val in self.active:
            sqr.append((math.exp(val[0]) - mean) ** 2)
        s = math.sqrt(sum(sqr) / self.freedom)
        plot = list()
        for i in range(0, self.rows):
            val = math.exp(self.active[i][0])
```

```
            # a_t = np.transpose(np.array(a[i])[np.newaxis])
            a_t = np.array(a[i])[np.newaxis]
            first = np.dot(a_t, xt_x)
            second = np.array(a[i])
            res1 = math.sqrt(np.dot(first, second) + 1. / self.rows)
            res2 = math.sqrt(np.dot(first, second) + 1. / self.rows + 1)
            diff1 = res1 * s * self.student
            diff2 = res2 * s * self.student
            # print(i, ":", diff1, diff2, ":", val - diff1, val + diff1,
"|", val - diff2, val + diff2)
            plot.append([val, val - diff1, val + diff1, val - diff2, val
+ diff2])

        colors = 'bggrr'
        x = np.arange(self.rows)
        for i in range(0, 5):
            y = list()
            for j in range(0, self.rows):
                y.append(plot[j][i])
            plt.plot(x, y, colors[i])
        plt.show()
```

5) <u>Клас «view»</u>

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1000, 700)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1000, 22))
        self.menubar.setObjectName("menubar")
        self.menuFile = QtWidgets.QMenu(self.menubar)
        self.menuFile.setObjectName("menuFile")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionX = QtWidgets.QAction(MainWindow)
        self.actionX.setObjectName("actionX")
        self.actionY = QtWidgets.QAction(MainWindow)
        self.actionY.setObjectName("actionY")
        self.actionOpen = QtWidgets.QAction(MainWindow)
        self.actionOpen.setObjectName("actionOpen")
        self.menuFile.addAction(self.actionOpen)
        self.menubar.addAction(self.menuFile.menuAction())

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
```

```
        MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
        self.menuFile.setTitle(_translate("MainWindow", "File"))
        self.actionX.setText(_translate("MainWindow", "Open"))
        self.actionY.setText(_translate("MainWindow", "Y"))
        self.actionOpen.setText(_translate("MainWindow", "Open"))
```

ДОДАТОК В - ОПИС ПРОГРАМИ

Програмне забезпечення автоматизованої обробки інформації для оцінювання розміру веб–застосунків на базі фрейворку Django, що розроблялась у рамках магістерської роботи, спрямоване на використання на підприємствах, які займаються розробкою програмного забезпечення.

Дане програмне забезпечення розроблене з метою спрощення бізнес-процесів ІТ-компаній, покращення прооцесу обробки масивів статистичних даних, полегшення оцінювання розміру подібних застосунків та підвищення достовірності оцінювання.

Функціональним призначенням програмного забезпечення є автоматизація процесів:

- вводу статистичних даних;
- нормалізації вихідних вибірок за допомогою логарифмування за основою натурального логарифма;
- розрахунку параметрів для вихідних та нормалізованих вибірок;
- побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;
- побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього.

Для розробки програмного забезпечення було обрану мову Python та середовище розробки PyCharm, а також написано декілька консольних модулів на мові програмування R для перевірки точності отриманих результатів.

При розробці додатку значну увагу було приділено користувацькому інтерфейсу. Важливим завданням було створити його якомога простим у використанні і в той самий час багатофункціональним і зручним, адже ним користуватимуться дуже часто.

Було проведено повне функціональне тестування, а також навантажувальне тестування. Усі виявлені недоліки ПЗ були зафіксовані в протоколах і усунуті розробником до моменту впровадження програми в дію.

Наступні випробування були успішними та показали повну готовність програми до використання.

ДОДАТОК Г - ІНСТРУКЦІЯ КОРИСТУВАЧА

Програмне забезпечення автоматизованої обробки інформації для оцінювання часу розробки веб-застосунків з використанням фреймворку Django в рамках магістерської роботи.

Після запуску на виконання файлу Django-ocinka.exe перед користувачем з'являється початкове вікно програми. Оберемо файл з даними за адресою в файловій системі: D:/1.csv (рисунок Г.1):

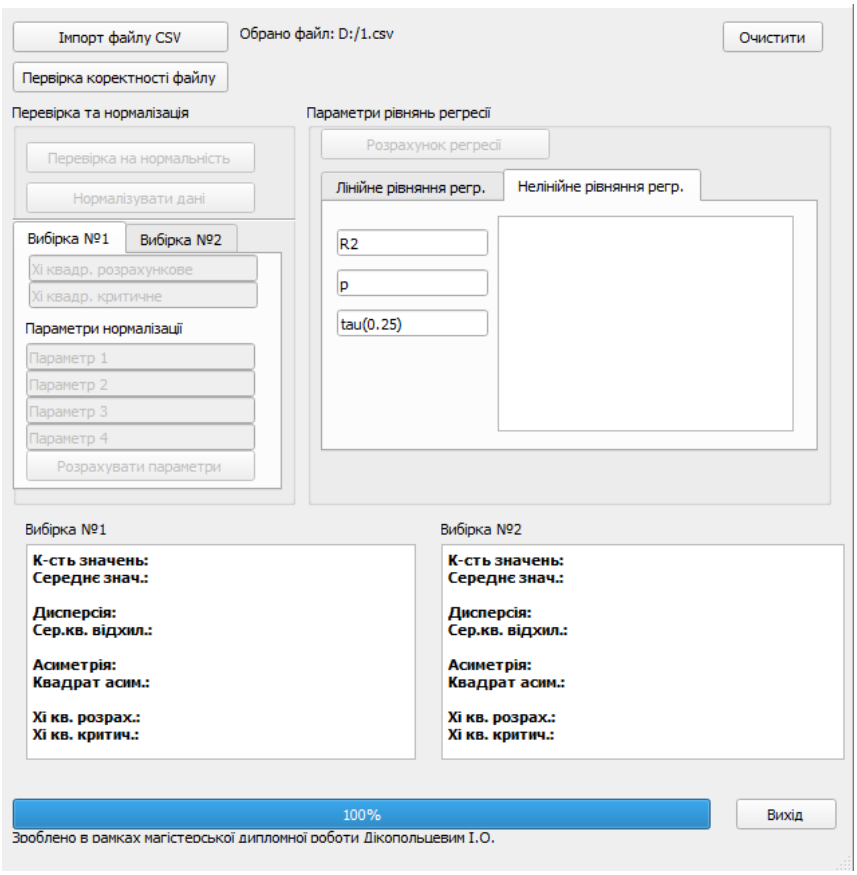

Рисунок Г.1 - Початкове вікно програми з обраним файлом для обробки даних

Програма опрацьовує інформацію поетапно, тому не всі кнопки є активними. Опис функцій та інформаційних блоків, наявних на початковому вікні програми наведено нижче:

1. Кнопка для імпорту файлу з даними.
2. Кнопка для очистки усіх полів.
3. Кнопка перевірки коректності даних в файлі імпорту.

4. Кнопка перевірки на нормальність даних.
5. Кнопка функції нормалізації вхідних даних.
6. Кнопка для розрахунку параметрів вихідних вибірок.
7. Кнопка розрахунку параметрів регресії.
8. Блок параметрів вихідних даних.

Порядок дій для оцінювання часу виконання веб-застосунків, розроблених за допомогою фреймворка Django:

1. Натиснути кнопку «Імпорт файлу *.csv», відкривається вікно вибору файлу. Обирається необхідний файл.
2. Натиснути кнопку «Перевірка коректності файлу». Відразу отримуємо розраховані значення у нижньому блоці вікна
3. Після натискання кнопки «Перевірка на нормальність»,.програма визначає, чи належать отримані дані до нормального закону розподілу.
4. Після перевірки на нормальність розподілу натискаємо «Нормалізувати». Відразу стають активними поля вводу параметрів.
5. Вводимо вручну параметри для початкового наближення в блок параметрів та натискаємо кнопку «Розрахувати» (8).
6. Нові результати розрахунків записуються у відповідні поля в нижньому блоці вікна. Перевіряємо нормалізовані вибірки на нормальність, натиснувши «Перевірити на нормальність». Цього разу, перевірка має пройти успішно. Всі дані записуються в окремий файл в каталозі з програмою.
7. Натискаємо кнопку «Розрахунок регресії». Стають аткивними поля вводу для лінійного та нелінійного рівнянь регресії.
8. Після вводу всіх параметрів, програма розраховує параметри для відповідного типу рівняння та будує графік (крайній блок праворуч) для відповідного рівнняня регресії, його довірчого інтервалу та інтервалу прогнозування.
9. Всі дані автоматично після кожного блоку розрахунків записуються в зовнішній файл.

ДОДАТОК Д - ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Об'єкт випробування

Об’єктом випробування є програмне забезпечення для оцінювання розміру веб-застосунків на базі Django-фреймворку, розроблене в рамках даної роботи.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного продукту, перевірка відповідності характеристик та вимог розробленої програми, викладених у технічному завдані, приведення прикладу роботи та отримання відповідних навичок.

3 Вимоги до додатку

При проведенні випробувань функціональні характеристики (можливості) програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

4.1 Склад програмної документації, пропонованої на випробування

Програмна документація повинна включати в себе наступні документи:

1) Технічне завдання;

2) Текст програми;

3) Опис програми;

4) Інструкція користувача;

5) Програма та методика випробувань.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

– CPU: Intel Pentium 4405Y (1.2 ГГц);
– RAM: 1 GB;
– Жорсткий диск 500 ГБ;
– Роздільна здатність екрану – 1920×1080 пікселів;
– Клавіатура 101/102-х клавішна рус / лат;
– Маніпулятор «миша».

5.2 Програмні засоби, що використовуються під час випробувань

Програмне забезпечення не пред'являє ніяких вимог до апаратної платформи.

Склад програмних засобів, що використовувалися під час випробувань:

Операційні системи:

- Windows 10 Pro x64.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної

документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

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

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

В процесі тестування була перевірена функціональність програмного забезпечення для оцінювання часу виконання веб-застосунків. У таблиці Д.1, що наведена нижче, вказано перелік випробувань основних функціональних можливостей.

Таблиця Д.1 - Методика випробування

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	2	3	4	5
1	Випробування на імпорт непідтримуваного формату файлу	Виводиться повідомлення про помилку вхідного файлу	Виконано	
2	Випробування на імпорт коректного формату файлу	У результаті файл успішно завантажився та у вікні було виведено його назву. Кнопка «перевірка коректності» стала активною	Виконано	

Продовження таблиці Д.1

1	2	3	4	5
3	Випробування на розрахунок параметрів без завантаження файлу	Усі кнопки на екрані неактивні. Окрім головно меню програми	Виконано	
4	Випробування на розрахунок параметрів після завантаження файлу	У відповідних текстових блоках з’явились розраховані параметри	Виконано	
6	Випробування на розрахунок параметрів рівняння регресії при некоректному вводі	У результаті з’явилось відповідне системне повідомлення, про необхідність ввести коректні параметри для розрахунку	Виконано	
7	Випробування на розрахунок параметрів рівняння регресії при коректному вводі	Відбувається підрахунок даних, що списком виводяться внизу діалогового вікна	Виконано	
8	Перевірка правильності вводу змінних для визначення рівнянь регресії	У результаті з’явилось вікно, в якому виведено розраховані рівняння регресії та побудовано графік	Виконано	

ДОДАТОК Е – РОЗРАХУНОК ДОВІРЧИХ ТА ПРОГНОЗОВАНИХ ІНТЕРВАЛІВ

Таблиця Е.1 – Розрахункова таблиця довірчих та прогнозованих інтервалів

Y	**ln(Y)**	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**Y_solv**	**$E^{(Y\_solv)}$**	**Довірчий**		**Прогнозований**	
										лівий	правий	лівий	правий
490	7.84	6	7	5.5	1	0.9	1	7,81	2461,96	92,70	1815,36	-1054,88	2962,95
502	7.84	7	2.75	4	1	0.2	1	7,79	2427,90	298,00	2005,62	-853,90	3157,51
514	7.83	8	5	5.7	1.89	0.9	1	7,77	2370,20	194,67	1635,49	-1037,58	2867,73
532	7.82	3	6.58	7.33	0.33	0.14	1.02	7,85	2582,39	139,81	1700,78	-1055,31	2895,91
579	7.81	7	2	5.5	1.91	0.56	1	7,70	2223,13	146,89	1702,77	-1049,77	2899,43
580	7.81	8	2.89	2.26	0.44	0	1	7,73	2292,87	-434,67	2005,76	-1401,41	2972,50
611	7.79	8	3	5.56	2	0.43	1	7,76	2346,07	340,74	1741,93	-904,10	2986,76
752	7.73	8	3	5.5	1.1	1	1.07	7,76	2346,07	123,05	2152,37	-941,57	3216,98
776	7.72	10	4.26	4.25	1.3	0.4	1	7,71	2249,79	545,81	2052,10	-666,02	3263,91
853	7.69	11	3.55	7.27	0.82	0.73	1	7,71	2237,46	272,29	1914,00	-898,76	3085,04
894	7.67	11	7.9	1.6	0.23	0	1	7,73	2282,18	594,57	2211,20	-583,88	3389,65
950	7.64	17	4.63	3	0.77	0.43	1.05	7,58	1964,14	488,72	2317,57	-629,11	3435,38
972	7.63	9	4.26	5.56	2	0.42	1.02	7,74	2310,89	774,30	2217,74	-457,11	3449,15
1092	7.57	6	8	6.5	0.71	0.6	1	7,82	2484,13	363,81	2289,42	-727,86	3381,08
1242	7.49	8	4	3.94	0.59	0.44	1	7,75	2320,38	43,88	1803,32	-1093,27	2940,48
1248	7.49	7	10	9.08	0.83	0.67	1	7,77	2380,33	585,83	2010,59	-651,50	3247,92
1254	7.48	9	11	9.08	1	0.68	1	7,75	2310,89	454,78	1932,10	-766,02	3152,90
1269	7.48	10	5.18	7	1	0.56	1.09	7,71	2223,13	770,02	2162,08	-477,75	3409,85
1473	7.35	11	7	6.5	0.59	0.42	1	7,70	2212,96	968,09	2484,26	-240,70	3693,04
1499	7.34	14	5	7.27	3.47	1.39	1.05	7,62	2043,65	1418,55	3072,11	250,99	4239,68
1627	7.25	14	4.93	8.29	1.93	0.57	1.21	7,66	2118,42	729,26	2455,00	-417,44	3601,71
6202	8.06	101	6.11	3.2	1	0.4	1.05	8,29	3991,59	1096,59	2577,93	-122,96	3797,48
8022	8.51	137	4.33	4.25	0.59	0.18	1.02	8,44	4637,50	674,03	2158,35	-544,59	3376,97
10734	8.95	76	6.33	7.59	2.1	1.85	1.05	8,09	3291,84	760,49	3444,82	-154,62	4359,93
10790	8.96	84	4.26	4.92	1.1	0.4	1.07	8,21	3680,68	963,07	2520,52	-233,11	3716,72
16022	9.47	219	2.77	5.5	0.75	0.67	1.1	8,78	6490,58	1557,68	3501,78	470,90	4588,55
17023	9.55	159	5.28	5.56	1	0.64	1.1	8,71	6068,67	3356,58	6432,80	2515,69	7273,69