

***БАКАЛАВРСЬКА
РОБОТА***

БР.ІП – 08.00.00.000 ІПЗ

Група ІП-22-1

Ільків Віталій

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ільків Віталій Володимирович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Аналіз дефектів програмного забезпечення

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Ільків В.В.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Процюк Василь Романович, доцент, к.т.н.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою

ІІЗ

доцент.

В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Ільків Віталію Володимировичу

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) "Аналіз дефектів програмного забезпечення "

керівник проекту (роботи) доц., к.т.н., Процюк В.Р.

затверджені наказом вищого навчального закладу від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз вимог до програмного забезпечення

2. Аналіз вимог і постановка задачі

3. Проектування системи

4. Реалізація проекту

5. Тестування та впровадження

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1 Методи відбору ознак (рис. 1.1, ст. 15)

2 Експериментальна установка (рис. 2.1, ст. 29)

3 Запропонований метод (рис. 2.2, ст. 31)

4 Кореляція ознак (рис. 2.3, ст. 36)

5 Залежність інерції від кластера (рис. 2.4, ст. 37)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання 2026 р.

Керівник

_____ (підпис)

Завдання прийняв до виконання

_____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	17.03.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	21.04.2026	виконано
3	Побудова моделі або алгоритму власного рішення	01.05.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	21.05.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	09.06.2026	виконано

Студент

_____ (підпис)

Керівник роботи

_____ (підпис)

АНОТАЦІЯ

Бакалаврська робота містить 83 сторінки, 12 рисунків, 15 таблиць, список використаних джерел із 40 найменувань.

Метою роботи є дослідження методів аналізу дефектів програмного забезпечення.

Об'єкт дослідження: процес виявлення, аналізу та прогнозування дефектів програмного забезпечення.

Предмет дослідження: методи аналізу дефектів програмного забезпечення, алгоритми прогнозування дефектів, засоби обробки даних та оцінювання якості програмних систем.

Результати дослідження: проведено аналіз сучасних підходів до виявлення та прогнозування дефектів програмного забезпечення, виконано обробку даних для побудови моделей прогнозування.

У першому розділі розглянуто теоретичні основи аналізу дефектів програмного забезпечення, методи вибору ознак для прогнозування дефектів та проаналізовано сучасні наукові дослідження у цій галузі.

У другому розділі досліджено методологію аналізу дефектів програмного забезпечення, охарактеризовано використовувані матеріали та методи, а також розглянуто категорії алгоритмів прогнозування дефектів.

У третьому розділі виконано підготовку та обробку даних, реалізовано методи вибору ознак для прогнозування дефектів програмного забезпечення та проведено аналіз отриманих результатів.

Висновок: результати дослідження підтверджують ефективність використання методів аналізу даних та прогнозування дефектів для підвищення якості програмного забезпечення.

КЛЮЧОВІ СЛОВА: ДЕФЕКТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ПРОГНОЗУВАННЯ ДЕФЕКТІВ, АНАЛІЗ ДАНИХ, ВИБІР ОЗНАК, МАШИННЕ НАВЧАННЯ, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, МОДЕЛІ ПРОГНОЗУВАННЯ.

ANNOTATION

The thesis consists of 82 pages, 12 figures, 15 tables, and a reference list containing 40 sources.

The aim of the thesis is to study software defect analysis methods.

Research object: the process of software defect detection, analysis, and prediction.

Research subject: software defect analysis methods, defect prediction algorithms, data processing techniques, and software quality assessment approaches.

Research results: modern approaches to software defect detection and prediction were analyzed, feature selection methods were investigated, data processing techniques for building prediction models were implemented, and the effectiveness of the obtained results was evaluated.

The first section presents the theoretical foundations of software defect analysis, feature selection methods for defect prediction, and a review of related research in this field.

The second section describes the methodology of software defect analysis, the materials and methods used in the research, and the main categories of defect prediction algorithms.

The third section focuses on data preprocessing and manipulation, implementation of feature selection methods for software defect prediction, and analysis of the obtained results.

Conclusion: the conducted research confirms the effectiveness of data analysis and defect prediction methods for improving software quality, reducing the number of defects during software development, and optimizing software testing processes.

KEY WORDS: SOFTWARE DEFECTS, DEFECT PREDICTION, DATA ANALYSIS, FEATURE SELECTION, MACHINE LEARNING, SOFTWARE TESTING, SOFTWARE QUALITY, PREDICTION MODELS.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	9
1.1 Основи аналізу дефектів програмного забезпечення	9
1.2 Методи вибору ознак для аналізу та прогнозування дефектів програмного забезпечення	12
1.3 Огляд існуючих досліджень та підходів до аналізу дефектів програмного забезпечення	15
1.4 Висновок по розділу	21
РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	22
2.1 Методологія аналізу дефектів програмного забезпечення	22
2.2 Матеріали та методи дослідження дефектів програмного забезпечення	28
2.3 Категорії алгоритмів аналізу та прогнозування дефектів програмного забезпечення	36
2.4 Висновок по розділу	40
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МЕТОДІВ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	41
3.1 Підготовка та обробка даних для аналізу дефектів програмного забезпечення	41
3.2 Реалізація методів вибору ознак для аналізу та прогнозування дефектів програмного забезпечення	53
3.3 Результати аналізу дефектів програмного забезпечення та їх обговорення	69
3.4 Висновок по розділу	77
ВИСНОВКИ	79
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА	81

					БР.ІП – 08.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Аналіз дефектів програмного забезпечення Пояснювальна записка	Літ.	Арк.	Акрушів
Розроб.	Ільків В.В.						7	
Перевір.	Процюк В.Р.							
Реценз.	Юрчишин В.М.							
Н. Контр.	Піх М.М.							
Затверд.	Бандура В. В.					ІФНТУНГ ІП-22-1		

ВСТУП

У сучасному світі програмне забезпечення використовується практично в усіх сферах діяльності людини, починаючи від бізнесу та освіти й закінчуючи критично важливими системами охорони здоров'я, транспорту та фінансів. Зростання складності програмних систем, збільшення обсягів коду та високі вимоги до надійності призводять до необхідності своєчасного виявлення та усунення дефектів програмного забезпечення. Навіть незначні помилки можуть спричиняти фінансові втрати, зниження продуктивності систем, порушення безпеки та погіршення користувацького досвіду. Сучасні виклики, пов'язані з розробкою складних програмних продуктів, вимагають застосування ефективних методів аналізу дефектів і прогнозування їх виникнення на ранніх етапах життєвого циклу програмного забезпечення.

Актуальність теми зумовлена необхідністю підвищення якості програмного забезпечення шляхом використання сучасних підходів до аналізу дефектів, обробки даних та прогнозування потенційних помилок. Існуючі методи тестування та контролю якості не завжди дозволяють своєчасно виявити всі проблеми, тому особливої уваги набувають методи аналізу даних, вибору ознак та алгоритми прогнозування дефектів, що дають змогу підвищити ефективність процесу розробки програмних продуктів.

Метою роботи є дослідження методів аналізу дефектів програмного забезпечення, вивчення підходів до прогнозування дефектів та оцінка ефективності сучасних методів обробки даних для підвищення якості програмних систем.

Завданнями дослідження є аналіз теоретичних основ аналізу дефектів програмного забезпечення, дослідження методів вибору ознак для побудови точних моделей прогнозування, вивчення сучасних підходів та алгоритмів прогнозування дефектів, аналіз методологій обробки даних, реалізація методів вибору ознак для прогнозування дефектів, оцінка отриманих результатів та формулювання висновків щодо ефективності застосованих підходів.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є процес виявлення, аналізу та прогнозування дефектів програмного забезпечення на різних етапах його життєвого циклу.

Предметом дослідження є методи аналізу дефектів програмного забезпечення, алгоритми прогнозування дефектів, підходи до вибору ознак та інструменти обробки даних, що використовуються для підвищення якості програмних продуктів.

Методи дослідження включають аналіз наукових джерел, порівняльний аналіз існуючих підходів до прогнозування дефектів, методи обробки та підготовки даних, методи вибору ознак, статистичний аналіз результатів та експериментальні дослідження ефективності моделей прогнозування.

Очікується, що результати роботи сприятимуть підвищенню якості програмного забезпечення, зменшенню кількості дефектів на етапах розробки та тестування, а також оптимізації процесів забезпечення якості програмних систем.

Бакалаврська робота містить 83 сторінки, 13 рисунків, 15 таблиць, 3 розділи, список використаних джерел із 40 найменувань.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Основи аналізу дефектів програмного забезпечення

Моделі прогнозування дефектів програмного забезпечення (SDP) будуються з використанням метрик програмного забезпечення, які базуються на даних, зібраних з попередньо розробленої системи або аналогічних програмних проєктів [1]. За допомогою такої моделі можна передбачити схильність до дефектів програмних модулів, що розробляються. Метою SDP є досягнення високої якості та надійності програмного забезпечення з ефективним використанням доступних обмежених ресурсів. Іншими словами, SDP передбачає виявлення програмних модулів або компонентів, схильних до дефектів. Це дозволить інженерам-програмістам визначити пріоритети використання обмежених ресурсів на кожному етапі розробки програмного забезпечення [2]. Отже, надійність та якість оцінки програмного забезпечення.

Окрім забезпечення якості програмного забезпечення, гарантуються [3]. Метрики програмного забезпечення, які включають складність вихідного коду програмного забезпечення та історію розробки, зазвичай використовуються для аналізу ефективності процесу розробки програмного забезпечення, якості та надійності програмних продуктів. Крім того, розробники програмного забезпечення використовують ці метрики програмного забезпечення для оцінки ризиків, а також для прогнозування дефектів з метою виявлення та покращення якості програмних продуктів [4]. Зокрема, метрики Маккейба та Халстеда, процедурні метрики, метрики процесу тощо – це типи інженерних метрик програмного забезпечення, що використовуються для визначення рівня якості та надійності програмної системи [5]. Програмний модуль або компонент містить набір ознак (метрик) та мітку класу. Мітка класу відображає стан програмного модуля, як дефектний чи без дефектів, тоді як похідні ознаки використовуються для розробки моделей SDP [6]. Тобто, SDP використовує

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

історичні дані, отримані з репозиторіїв програмного забезпечення, для визначення якості та надійності програмних модулів або компонентів.

SDP можна розглядати як завдання класифікації, яке включає категоризацію програмних модулів як дефектних або недефектних на основі історичних даних та метрик або ознак програмного забезпечення [7]. Функції або метрики програмного забезпечення відображають характеристики програмних модулів. Однак кількість згенерованих метрик зазвичай велика. Це пов'язано з різними типами механізмів метрик програмного забезпечення, що використовуються для визначення якості та надійності програмної системи. Поширення цих механізмів, відповідно, генерує велику кількість значень метрик, що призводить до проблеми високої розмірності, коли генерується багато значень ознак. Крім того, деякі з цих ознак (метрик) можуть бути більш релевантними для класу (дефектні чи недефектні), ніж інші, а деякі можуть бути надлишковими або нерелевантними [8].

Відбір ознак (FS) може бути використаний для вибору високо некорельованих ознак з високовимірних ознак. Іншими словами, він може вибирати ті ознаки, які є більш релевантними та ненадлишковими для мітки класу набору даних серед ознак. Таким чином, впровадження методів FS в SDP може вирішити проблему високовимірності. FS є життєво важливим кроком попередньої обробки даних у процесах класифікації, оскільки він покращує якість даних і, як наслідок, покращує прогностичну ефективність моделей прогнозування. Існуючі дослідження показали, що нерелевантні ознаки, разом з надлишковими ознаками, можуть серйозно впливати на точність предикторів дефектів [9]. Таким чином, існує потреба в пошуку ефективного методу вибору ознак, який може ідентифікувати та видалити якомога більше нерелевантної та надлишкової інформації, що призведе до хорошої прогностичної ефективності з низькими обчислювальними витратами. Методи контролю вибору ознак оцінюють доступні характеристики ознак та виводять набір відповідних характеристик на основі позначених наборів даних. Критерії, що використовуються для визначення корисних характеристик ознак, залежать від

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

основних обчислювальних характеристик використовуваного методу. Методи ранжування фільтрувальних ознак (FFR), які є типами методів контрольованого FS, залежать від обчислювальних характеристик кожної ознаки за певними критичними факторами, а потім аналітик відбирає деякі ознаки, які відповідають певному набору даних. З іншого боку, методи вибору підмножини фільтрувальних ознак (FSS) (інший тип методів контрольованого FS) шукають підмножину ознак, які мають хороші прогностичні можливості разом. У цьому дослідженні досліджується порівняльний аналіз ефективності методів FFR та FSS, заснований на різних методах пошуку, щоб визначити їхню відповідну ефективність у відборі відповідного набору ознак.

Нещодавні дослідження порівнювали вплив методів FS на ефективність SDP [10]. Деякі дослідження роблять висновок, що деякі методи FS кращі за інші, тоді як деякі дослідження стверджують, що немає суттєвої різниці між ефективністю методів FS в SDP. Ця суперечність та невідповідність результатів існуючих досліджень можуть бути пов'язані з вибором механізму пошуку, що використовується в методах FS.

У цьому дослідженні було використано чотири різні методи FFR (інформаційний приріст (IG), полегшення (RFA), коефіцієнт посилення (GR) та варіація кластеризації (CV)), засновані на методі ранжування, та два методи FFS: вибір підмножини ознак на основі кореляції (CFS) та вибір підмножини ознак консистентності (CNS) на основі двох різних підходів до пошуку: вичерпний пошук (методи пошуку за принципом «найкращий перший пошук» (BFS) та жадібний покроковий пошук (GSS)) та евристичний пошук (генетичний пошук (GS), пошук кажанів (BAT), пошук мурах (AS), пошук світлячків (FS) та метод оптимізації рою частинок (PSO)). Чотири методи класифікації: наївний

Для оцінки ефективності цих методів FS було використано баєсівський (NB), дерево рішень (DT), логістичну регресію (LR) та метод К-найближчих сусідів (KNN). Відповідні моделі були використані на п'яти наборах даних про дефекти програмного забезпечення з репозиторію NASA, а їх прогностична ефективність була виміряна порівняно на основі точності.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Згідно з нашими експериментальними результатами, застосування FS покращує прогностичну ефективність моделей прогнозування, а ефективність методів FS варіюється залежно від набору даних та моделей прогнозування. IG зафіксував найкраще покращення моделей прогнозування порівняно з іншими методами FFR, тоді як CNS на основі BFS мав найкращий вплив на моделі прогнозування на основі методів FSS. Подальший аналіз показав, що моделі прогнозування на основі FFR є стабільнішими з точки зору точності продуктивності, ніж інші методи FS.

1.2 Методи вибору ознак для аналізу та прогнозування дефектів програмного забезпечення

Методи вибору ознак (FS) використовуються для визначення та збереження найбільш релевантних та інформативних ознак даних для побудови точних моделей прогнозування [11]. У літературі використовуються різні терміни, такі як атрибути, метрики та виміри були використані для програмних функцій. Дані містять численні ознаки, що призводить до появи шумної, нерелевантної та надлишкової інформації. В результаті, алгоритм навчання зазнає збільшення як часу обчислення, так і рівня помилок [12]. Зі зростанням обсягу даних у геометричній прогресії якість даних, необхідних для обробки, поступово знижується. Це обмеження зазвичай називають «прокляттям розмірності».

В останні роки методи вибору ознак набули більшої популярності завдяки їхній здатності підвищувати точність прогнозування та скорочувати час створення моделі. Ці методи знаходять застосування в різних галузях, включаючи охорону здоров'я та медицину, виявлення шахрайства, аналіз настроїв тощо [13]. Існують чотири основні категорії методів вибору ознак: фільтри, обгортки, вбудовані та гібридні. Короткий опис кожного методу вибору ознак наведено нижче.

Метод фільтрації: Метод фільтрації, підхід з відкритим циклом, є

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

одним із найперших методів вибору ознак. Він оцінює ознаки на основі їхніх внутрішніх характеристик перед завданнями навчання. Цей метод спирається на чотири критерії вимірювання: інформація, залежність, узгодженість та відстань. Він виконує вибір ознак незалежно від алгоритму машинного навчання та використовує статистичні стандарти для ранжування підмножин ознак [14]. Приклади включають інформаційний приріст, тест хі-квадрат та коефіцієнт кореляції тощо.

Метод обгортки: Метод обгортки, також відомий як підхід із замкнутим циклом, інкапсулює процес вибору ознак навколо алгоритму навчання та використовує точність або коефіцієнт помилок процесу класифікації для оцінки ознак. Його метою є вибір найбільш дискримінаційної підмножини ознак шляхом зменшення похибки оцінки конкретного класифікатора. Метод обгортки виконує вибір ознак на основі продуктивності алгоритму навчання, зрештою вибираючи найоптимальніші ознаки для процесу прогнозування. Через свою багатовимірну природу більшість методів обгортки вимагають тривалого часу обчислення для досягнення збіжності, що робить їх непрактичними для великих наборів даних. Приклади включають прямий вибір, зворотне виключення та рекурсивне виключення [15] .

Вбудований метод: Вбудований метод – це вбудований механізм вибору ознак, який інтегрує вибір ознак безпосередньо в алгоритм навчання, використовуючи його властивості для керівництва оцінкою ознак. Порівняно з методом-обгорткою, вбудований підхід є більш ефективним з обчислювальної точки зору, зберігаючи при цьому подібний рівень продуктивності. Поєднуючи якості методів фільтрації та обгортки, вбудований метод вибирає ознаки під час реалізації алгоритму, що призводить до зниження обчислювальної складності. Це досягається шляхом уникнення повторних виконань класифікатора та перевірки кожної підмножини ознак. Приклади включають методи на основі LASSO та дерев регуляризації.

Гібридний метод: Нещодавні досягнення у виборі ознак призвели до розробки гібридного методу.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

Підхід можна сформулювати шляхом поєднання двох різних методів, наприклад, інтегруючи як методи обгортки, так і фільтрації з ідентичними критеріями або методологіями вибору ознак. Застосовуючи гібридний метод, можна досягти переваг обох методів, використовуючи їхні додаткові сильні сторони для покращення процесу вибору ознак [16]. Рисунок 1.1 ілюструє ієрархічну структуру методів вибору ознак.

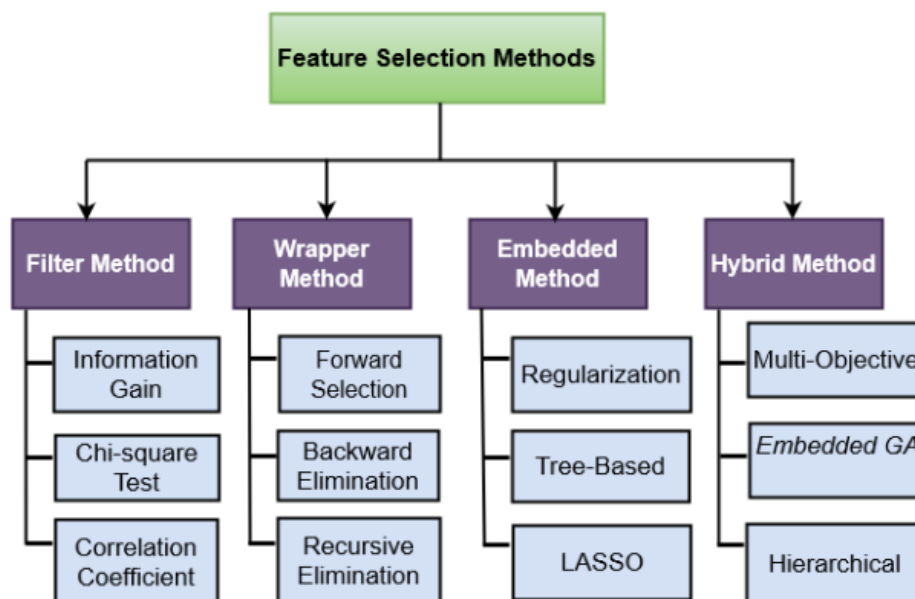


Рисунок 1.1 - Методи відбору ознак.

Прогнозування дефектів програмного забезпечення (SDP) використовує історичні дані та методи машинного навчання для прогнозування та виявлення потенційних дефектів у програмних системах на етапі розробки. Головною метою прогнозування дефектів програмного забезпечення є проактивне виявлення областей коду, які ймовірно містять недоліки, що дозволяє команді з контролю якості програмного забезпечення зосередити свої зусилля з тестування та налагодження на цих областях з високим ризиком [17]. Протягом останнього десятиліття кілька дослідників представили емпіричні докази того, що включення методу вибору ознак на етапі попередньої обробки призводить до підвищення точності класифікації в контексті прогнозування дефектів програмного забезпечення [18]. Систематичний огляд літератури (SLR) – це

структурований дослідницький процес для систематичного виявлення, оцінки та синтезу існуючої літератури з певної теми чи питання дослідження. Він має на меті надати всебічний та неупереджений огляд доступних доказів, пов'язаних з обраною темою.

Враховуючи важливість вибору ознак у прогнозуванні збоїв програмного забезпечення, варто зазначити, що попередній систематичний огляд надав цінні висновки з останнього десятиліття досліджень [19]. Цей попередній огляд, у якому розглянуто 15 первинних досліджень з 2007 по 2017 рік, заклав основу для вивчення ролі методів вибору ознак, але залишив простір для подальшого дослідження. Цей систематичний огляд намагається розширити це дослідження на ширший та більш актуальний спектр. Розглянувши вибірку з 49 первинних досліджень з авторитетних репозиторіїв з 2014 по 2023 рік, включаючи IEEE Xplore, Science Direct, ACM Digital Library та Springer Link, цей систематичний огляд всебічно розглядає останні досягнення в цій галузі. Він має на меті переглянути та оновити поточний стан методів вибору ознак у прогнозуванні помилок програмного забезпечення, надати нове розуміння їх застосування та відносної продуктивності, і таким чином розширити обсяг знань.

1.3 Огляд існуючих досліджень та підходів до аналізу дефектів програмного забезпечення

У SDP найпоширенішими підходами машинного навчання (ML) є кластеризація, класифікація та глибоке навчання. Використовуючи класифікатори на основі ML та статистичні підходи, дослідники запропонували різноманітні моделі SDP. У своєму дослідженні використовували кілька алгоритмів класифікації для прогнозування помилок програмного забезпечення з використанням дванадцяти наборів даних NASA. Класифікатори включають NB, базисну функцію повторного набору, K-найближчого сусіда, багатосаровий перцептрон, K-зірку, SVM, дерево рішень, одне правило та RF. Для оцінки продуктивності використовувалися точність, акуратність, повнота, f-міра та

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

AUC. Аналіз першопричин намагається знайти першопричини проблеми, щоб виправити її. Метод аналізу першопричин допомагає нам знаходити дефекти на ранній стадії. У моделі реалізовано різні типи методів кластеризації [20]. Кластеризація - це форма самостійного навчання, яка працює на подібності даних. Дослідники застосували різні типи методів кластеризації, щоб виявити дефект. Для реалізації методів кластеризації використовувався WEKA. Кластеризація К-середніх демонструє кращу продуктивність у цьому питанні.

Класифікація на основі машинного навчання також використовується для прогнозування помилок програмного забезпечення [21] . Для цього разом із машинним навчанням застосовується інший класифікатор. У цій класифікації на основі машинного навчання використовуються класифікатор штучних нейронних мереж (ANN), класифікатор NB та класифікатор дерева рішень, що дало хороші результати. П'ять класів є результатом цього підходу з вимірюваннями невеликих наборів даних. Експерименти реалізовано на трьох публічних наборах даних. WEKA 3.6.9 використовується як інструмент машинного навчання. Методи машинного навчання мають хороші результати в цьому завданні. Perreault та ін. використовували п'ять різних типів класифікаторів для виявлення дефектів програмного забезпечення у своєму дослідженні [22]. Для вимірювання продуктивності для класифікації було використано п'ять різних наборів даних з програми метрик NASA: нейронні мережі, логістична регресія, NB, SVM та k-найближчий сусід. Для деяких наборів даних NB та SVM працюють найкраще серед інших. Низька якість програмного забезпечення спричинена помилками програмного забезпечення. Як наслідок, критично важливо усунути помилки програмного забезпечення, щоб покращити якість програмного забезпечення.

Пропонується численні моделі для покращення якості програмного забезпечення шляхом вивчення факторів, що впливають на якість програмного забезпечення, та покращення продукту та якості з точки зору програмного забезпечення у своєму дослідженні. Вони розглядали різноманітні вимірювання розміру та складності, а також такі моделі, як байєсівські мережі довіри,

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

генетичні алгоритми та нейронні мережі, серед інших. Представлено ще одну гарну ідею. Основною метою роботи було визначити, скільки проблем існує загалом, щоб заощадити гроші та час. Обсяг, складність, зобов'язання та оцінка часу, а також тривалість програми, були використані для категоризації дефектів. Для цього вони застосували класифікатор дерева рішень. Для класифікації дефектів вони використовували алгоритм класифікації ID3. Потім вони застосували підхід інтелектуального аналізу шаблонів для класифікації помилкових шаблонів. Вони використовували JAVA для реалізації запропонованої парадигми.

Література з прогнозування моделей зосереджена на вилученні характеристик та використанні різних алгоритмів машинного навчання. У роботі представлено графі потоку керування (CFG) з нейронним мережевим підходом. Першим кроком цієї моделі є CFG, в якому вони представляють програмне забезпечення у вигляді графів. Після компіляції вихідного коду, CFG програми будується з коду асемблера. Використовується сильна нейронна мережа на графах міток, яка також відома як багатошарова згорткова нейронна мережа. Продуктивність цієї моделі оцінюється за допомогою чотирьох наборів даних.

Для прогнозування в цьому дослідженні представлена стандартна модель ADBBO (адаптивна оптимізація на основі розмірної біогеографії) у поєднанні з моделлю RBFNN (нейронна мережа з повторним набором базисних функцій). В експериментах використовуються п'ять наборів даних, які є загальнодоступними та є частиною програми даних NASA. Автори проводять порівняння з прогнозами попередніх досліджень для аналогічних наборів даних. Проблема дисбалансу класів знижує точність прогнозування дефектів. Навчання на основі ядра було введено. Ця проблема вирішується за допомогою AKPLSC (асиметричного класифікатора часткових найменших квадратів ядра) та AKPCAC (асиметричного класифікатора методом головних компонент ядра). Після цього обидва класифікатори піддаються дії функції ядра. Як ядро використовується функція Гауса. Експерименти проводяться з використанням баз даних SOFTLAB та NASA.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Використовується кластеризація за методом К-середніх та апріорний метод. Кластеризація використовується для досягнення дискретизації, а потім апріорний метод використовується для вилучення правил або шаблонів з даних. Експерименти проводяться з використанням даних про дефекти NASA. Нарешті, наведено порівняння з існуючими методами. Метод CART, який являє собою оптимізоване дерево регресії та класифікації [23]. Поряд з цим методом, для мінімізації розмірностей використовується аналіз основних компонентів. Вони стверджують, що час виконання оптимізації в цьому випадку є накладними витратами. Як результат, подальші покращення можуть бути реалізовані шляхом скорочення часу виконання.

Запропоновано генетичний алгоритм з оновленою глибокою нейронною мережею. Для оптимізації ознак застосовується генетичний алгоритм. В результаті для прогнозування дефектів з використанням чотирьох наборів даних використовується гібридний метод. Набір даних надано обіцяним репозиторієм даних. Для реалізації використовується інструмент MATLAB. Точність цього гібридного методу чудова.

У цьому дослідженні було використано метод ансамблю, заснований на виборі ознак [24]. Запропонований фреймворк було реалізовано як з вибором ознак (FS), так і без нього. NASA надало дванадцять очищених публічних наборів даних. Запропоновані результати порівнюються з результатами різних класифікаторів. Хоча результати покращилися, проблема дисбалансу класів у цій ситуації зберігається. У цій роботі розглядаються різні алгоритми машинного навчання [25]. Серед використаних алгоритмів – штучна нейронна мережа, дерево рішень, NB, PSO та лінійний класифікатор. Для проведення експериментів використовується інструмент KEEL, а також сім наборів даних з репозиторію даних NASA. У чотирьох наборах даних лінійний класифікатор перевершує решту.

Підхід ШНМ поєднується з підходом штучної бджолиної колонії (ABC). Для навчання штучної нейронної мережі використовується алгоритм ABC. Для оптимізації застосовується алгоритм ABC. У даних існує проблема дисбалансу

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

класів. Як результат, результати незбалансовані. Програма даних NASA надала п'ять окремих наборів даних для цього дослідження. Для оцінки продуктивності використовуються точність.

Обговорюються деякі метрики та набори даних програмного забезпечення для прогнозування дефектів. Вони здебільшого використовуються для пошуку дефектів у підході машинного навчання, який полягає у отриманні інформації з архівів програмного забезпечення, що містять повідомлення, а також вихідний код. Екземпляри містять метод, клас, вихідний код, пакети та зміни коду. Екземпляр також містить деякі функції, отримані з архівів програмного забезпечення. Значення метрик характеризують складність програмного забезпечення та розробку. Екземпляри розглядаються як позначені або непозначені. Метрики відіграють важливу роль у створенні будь-якої моделі прогнозування, яка допомагає покращити узгодженість програмного забезпечення, отримуючи якомога більше дефектів з програмного забезпечення. Метрики поділяються на двочастинний код, який описує складність коду, та метрики процесу, які описують складність розробки [25]. Найбільш використовуваними метриками коду є метрики рядків коду. Набір даних про дефекти важливий для прогнозування дефектів. Деякі ранні дослідження показують, що дослідники використовували неопублічний набір даних. Деякі загальнодоступні набори даних - це NASA, SOFTLAB, PROMISE, ReLink, AEEEM, ECLIPSE 1, ECLIPSE 2 тощо. Для прогнозування дефектів програмного забезпечення використовується багато оціночних показників, тобто ймовірність дефекту, коефіцієнт істинно позитивних (TP), коефіцієнт хибнонегативних (FN), коефіцієнт істинно негативних (TN) та хибнопозитивних (FP) результатів, точність, правильність, G-міра, F-міра, AUC. З новим розвитком досліджень також покращився алгоритм машинного навчання для SDP.

Для прогнозування несправностей використовується гібридний метод. Автори використовують підхід вибору ознак у поєднанні з різними класифікаторами машинного навчання. Для вибору ознак було використано метод оптимізованих штучних імунних мереж (Opt-aiNet). Для порівняння

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

результатів використовуються класифікатори машинного навчання. З репозиторію PROMISE використовується п'ять окремих наборів даних. Точність та площа під кривою (AUC) є показниками оцінки. Після вибору ознак продуктивність покращилася. Запропоновано рішення проблеми дисбалансу. Для вибору ознак було використано аналіз головних компонент. Для економічно чутливих проблем дисбалансу для прогнозування використовується ANFIS. Цей метод покращує площу ROC на 5%.

Щоб передбачити недоліки, вони використовували три класифікатори: дерево рішень, RF, SVM та ансамблевий підхід, включаючи Ada-boost та Bagging. Для даних дисбалансу використовувалася вибірка SMOTE. В експерименті використовуються десять наборів даних NASA. Серед усіх підходів RF та Ada-boost з RF показали найкращі результати. Використання багатьох класифікаторів у комбінованій стратегії є актуальним питанням у наші дні. Для прогнозування дефектів також використовувався змішаний або ансамблевий метод. Шість алгоритмів використовувалися на п'яти різних наборах даних з бази даних NASA MDP. Для покращення якості даних для вибірки використовувався SMOTE. Експерименти проводилися за допомогою інструменту WEKA. Найкращим ансамблевим алгоритмом є RF [26].

Використовується ансамблювання на основі дерев. Для чотирьох безкоштовних наборів даних NASA Metric Data Program використовується сім ансамблерів. Два з них використовують пакетування, а інші п'ять - бустинг. Багінг працює добре, проте Ada-boost працює погано. Використовуються різні алгоритми кластеризації, а отримані кластери потім інтегруються в єдину модель. Для підвищення якості програмного забезпечення використовується PSO. Для покращення якості програмного забезпечення класифікатор NB використовується разом з ARM, який використовується для вибору ознак [27]. Набори даних NASA використовуються для тестування п'яти різних наборів даних. Інші методи використовуються для оцінки продуктивності. У цьому огляді використовуються різні підходи та фреймворки машинного навчання з різними метриками для усунення проблем. Вони розглянули 40 досліджень,

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

опублікованих у різних журналах між 2009 і 2018 роками. Незважаючи на всю роботу, виконану в цій галузі, завжди є місце для вдосконалення через незбалансовану природу наборів даних та неоднозначностей.

Проаналізували п'ять публічних наборів даних з промісного репозиторію, використовуючи прогнози на основі машинного навчання та десять класифікаторів. В оцінюванні використовувалася точність вимірювань. У цьому опитуванні досліджувалися методи глибокого навчання для прогнозування дефектів [28]. Жоден з методів не давав послідовно результатів з високою точністю, повнотою та прецизійністю. SLR використовувався для моніторингу поточних розробок ансамблевих або гібридних методів. Була представлена стратегія підвищення продуктивності. З авторитетних онлайн-бібліотек було обрано 46 статей для формування короткого списку. Згідно з дослідженням, FS та вибірка даних покращують результати. Використовуючи метрики оцінювання, оцінюється продуктивність. Ансамблева стратегія перевершує інші.

Дослідження літератури показує, що досі застосовувалися різні методи машинного навчання (ML), але їхня ефективність варіюється залежно від набору даних, і з точки зору ML, їхня продуктивність менш точна. Тому ми хочемо підвищити точність, аналізуючи різні методи ML у поєднанні з FS та кластеризацією K-середніх. Метою дослідження є підвищення точності з урахуванням літературних досліджень.

1.4 Висновок по розділу

У першому розділі досліджено теоретичні основи аналізу дефектів програмного забезпечення. Розглянуто сутність дефектів, причини їх виникнення та вплив на якість програмних продуктів. Проаналізовано методи вибору ознак, що використовуються для виявлення та прогнозування дефектів, а також розглянуто сучасні наукові підходи до аналізу програмного забезпечення. Проведене дослідження підтвердило важливість своєчасного виявлення дефектів для підвищення надійності, якості та ефективності програмних систем.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Методологія аналізу дефектів програмного забезпечення

У цьому підрозділі описано методи FS, відповідні методи пошуку, алгоритми класифікації, експериментальну установку, набір даних про дефекти програмного забезпечення та показники продуктивності, що використовуються в цьому дослідженні.

Таблиця 2.1

Список методів ранжування фільтрувальних ознак (FFR).

Метод ранжування фільтрації ознак	Метод пошуку	Характеристики / Підхід
Оцінювач ознак за приростом інформації (IG) (Information Gain Attribute Evaluator)	Пошук за рангом (Ranker Search)	На основі теорії ймовірностей (Probability-based)
Рельєфний оцінювач атрибутів ознак (RFA) (Relief Feature Attribute Evaluator)	Пошук за рангом (Ranker Search)	На основі прикладів/екземплярів (Instance-based)
Оцінювач ознак за коефіцієнтом приросту (GR) (Gain Ratio Attribute Evaluator)	Пошук за рангом (Ranker Search)	На основі теорії ймовірностей (Probability-based)
Оцінювач ознак за варіацією кластеризації (CV) (Clustering Variation Attribute Evaluator)	Пошук за рангом (Ranker Search)	На основі статистичних методів (Statistics-based)

Метод ранжування функцій фільтра

Метод ранжування фільтрувальних ознак (FFR) використовує обчислювальні характеристики наборів даних для незалежної оцінки та ранжування атрибутів у наборах даних, які виявилися незалежними від моделі прогнозування. Він оцінює кожен атрибут на основі різних характеристик, таких як статистика, ймовірність, екземпляр або класифікатор. Потім атрибути вибираються на основі їхньої оцінки [29]. У цій роботі було розглянуто чотири методи FFR на основі різних функціональних характеристик. Для вибору ознак з найвищим рейтингом у цьому дослідженні було використано $\log^2 N$, де N –

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

кількість метрик програмного забезпечення у повному наборі даних про дефекти програмного забезпечення. У рисунку 2.1 представлено опис методів FFR, що використовуються в цьому дослідженні.

Метод вибору підмножини ознак фільтра

Вибір підмножини ознак (FSS), як і FFR, оцінює, ранжує та вибирає ознаки на основі деяких властивостей. Однак у випадку FSS основна увага приділяється методу пошуку, який використовується для створення підмножини ознак, що разом мають хороший прогностичний потенціал. Він враховує існування кращої прогностичної ефективності, коли ознака поєднується з іншими ознаками [30]. У цьому дослідженні було розглянуто два методи FSS. Як згадувалося раніше, FSS базується на різних методах пошуку. Таким чином, ці методи пошуку перетинають простір ознак, щоб створити підмножину з високим прогностичним потенціалом. Отже, продуктивність FSS залежить від методу пошуку [31]. У таблиці 2.2 наведено детальний опис методів FSS, що використовуються в цьому дослідженні, а в таблиці... На рис. 3 показано різні методи пошуку з їхніми параметрами, що використовуються в методах FSS.

Алгоритми класифікації

Для процесу класифікації було розглянуто чотири широко використовувані алгоритми класифікації для оцінки ефективності методів FS. Це найвний байєсівський метод (NB), дерево рішень (DT), ядерна логістична регресія (LR) та метод К найближчих сусідів (KNN). Це відповідає меті цього дослідження оцінити ефективність методів FFR та FSS, які повинні бути незалежними від методу класифікації.

Таблиця 2.2

Перелік методів відбору підмножин ознак фільтра (FSS).

Метод вибору підмножини ознак	Алгоритм / Метод пошуку
Вибір підмножини ознак на основі кореляції (CFS) (<i>Correlation-based Feature Subset Selection</i>)	Пошук за першим найкращим збігом (BFS / Best First Search)
	Покроковий жадібний пошук (GSS / Greedy Stepwise Search)

Продовження таблиці 2.2

	Покроковий жадібний пошук (GSS / Greedy Stepwise Search)
	Мурашиний алгоритм пошуку (AS / Ant Search)
	Алгоритм кажана (BAT / Bat Search)
	Алгоритм світлячка (FS / Firefly Search)
	Генетичний алгоритм пошуку (GS / Genetic Search)
	Пошук роєм часток (PSOS / PSO Search)
Вибір підмножини ознак на основі узгодженості (CNS) (Consistency Feature Subset Selection)	Пошук за першим найкращим збігом (BFS / Best First Search)
	Покроковий жадібний пошук (GSS / Greedy Stepwise Search)
	Мурашиний алгоритм пошуку (AS / Ant Search)
	Алгоритм кажана (BAT / Bat Search)
	Алгоритм світлячка (FS / Firefly Search)
	Генетичний алгоритм пошуку (GS / Genetic Search)
	Пошук роєм часток (PSOS / PSO Search)
Метод вибору підмножини ознак	Алгоритм / Метод пошуку

Таблиця 2.3

Різні методи пошуку та налаштування параметрів.

Метод пошуку(Search Methods)	Налаштування параметрів(Parameter Settings)
Пошук за першим найкращим збігом	Напрямок = Двонаправлений (Direction = Bi-directional)
Покроковий жадібний пошук	- Консервативний прямий вибір = True (Conservative Forward Selection = True) - Зворотний пошук = False (Search Backwards = False) - Кількість для вибору: $\text{NumToSelect} = \log_2 N$, де N — кількість ознак (N = Number of Features)
Мурашиний алгоритм пошуку	- Тип прискорення = Прискорений (AccelerateType = Accelerate) - Хаотичний коефіцієнт = 0.4 (Chaotic co-efficient = 0.4) - Тип хаотичного параметра = Хаотична карта: Параметр (Chaotic Parameter Type = Chaotic Map: Parameter) - Тип хаосу = Логістичне відображення (Chaotic Type = logistic map) - Розмір популяції = 200 (PopulationSize = 200) - Початковий феромон = 2.0 (Phromone = 2.0)
Алгоритм кажана	- Тип прискорення = Прискорений (AccelerateType = Accelerate) - Хаотичний коефіцієнт = 0.4 (Chaotic co-efficient = 0.4) - Тип хаотичного параметра = Логістичне відображення: Параметр (Chaotic Parameter Type = logistic Map: Parameter)

	• Тип хаосу = Логістичне відображення (Chaotic Type = logistic map) • Розмір популяції = 200 (PopulationSize = 200)
Метод пошуку(Search Methods)	Налаштування параметрів(Parameter Settings)
Пошук за першим найкращим збігом	Напрямок = Двонаправлений (Direction = Bi-directional)
Алгоритм світлячка	• Тип прискорення = Прискорений (AccelerateType = Accelerate) • Хаотичний коефіцієнт = 0.4 (Chaotic co-efficient = 0.4) • Тип хаотичного параметра = Логістичне відображення: Параметр (Chaotic Parameter Type = logistic Map: Parameter) • Тип хаосу = Логістичне відображення (Chaotic Type = logistic map) • Розмір популяції = 200 (PopulationSize = 200) • Поглинання світла = 0.001 (absorption = 0.001) • Мінімальна бета = 0.33 (betaMin = 0.33)
Генетичний алгоритм пошуку	• Розмір популяції = 200 (PopulationSize = 200) • Максимальна кількість поколінь = 20 (MaxGeneration = 20) • Ймовірність схрещування = 0.6 (crossoverProb = 0.6)
Пошук роєм часток	• Розмір популяції = 200 (PopulationSize = 200) • Індивідуальна вага = 0.34 (IndividualWeight = 0.34) • Інерційна вага = 0.33 (InertiaWeight = 0.33) • Соціальна вага = 0.33 (SocialWeight = 0.33)

Чотири класифікатори було обрано на основі різних характеристик: NB на основі теореми Байєса, DT - з методів на основі дерев, LR - з методу класифікації на основі функцій та KNN - з класифікаторів на основі навчання на основі екземплярів. Гетерогенність вибраних класифікаторів полягає в тому, щоб дослідити, як різні методи FS працюють на різних класифікаторах з різними характеристиками. У таблиці 2.4 наведено короткий опис цих алгоритмів класифікації щодо їхніх характеристик класифікації та налаштувань параметрів.

Таблиця 2.4

Алгоритми класифікації.

Класифікатор(Classifiers)	Опис метода(Description)	Налаштування параметрів(Parameter Setting)

Продовження таблиці 2.4

		= Нормальний розподіл (NumAttrEval = Normal Dist.)
Дерево рішень (DT)	Метод класифікації, що базується на деревоподібній структурі умов.	• Фактор довіри / впевненості = 0.25 (Confidence factor = 0.25) • Мінімальна кількість об'єктів у листку = 2 (MinNumObj = 2)
Ядерна логістична регресія (LR)	Метод класифікації, що базується на функціональному підході (з використанням ядерних функцій).	• Ядро = Поліноміальне ядро (Kernel = PolyKernel з параметрами ступеня \$E = 1.0\$ та константи \$C = 250007\$) • Параметр регуляризації лямбда = 0.01 (lambda = 0.01) • Квадратичний штрафметод оптимізації = Оптимізація BFGS (Quadratic penalty BFGS Optimization)
Метод k-найближчих сусідів (KNN)	Метод класифікації, що базується на навчанні за прецедентами (на основі прикладів із вибірки).	• Метод пошуку сусідів = Лінійний пошук найближчого сусіда (NNSearch = LinearNNSearch) • Метрика відстані = Евклідова відстань (based on Euclidean Distance)

Експериментальна установка

У цьому підрозділі розглядається експериментальна установка порівняльного аналізу ефективності методів FS, як зображено на рисунку 1. Експериментальну установку можна описати на основі трьох основних кроків.

Крок) 1 : Спочатку чотири методи FFR, як представлено в таблиці 2.1 застосовуються до вихідних повних наборів даних про дефекти програмного забезпечення. У цьому випадку кожен з методів FFR (IG, RFA, GR та CV на основі методу Ranker Search) оцінює та ранжує функції програмного забезпечення на основі їх відповідних характеристик. $\log 2 N$ ознак було вибрано зі списку ранжування, наданого кожним із методів FFR. Зокрема, перші шість ознак було вибрано зі списку ранжування, отриманого за допомогою методів FFR. $\log 2 N$ (де N – кількість ознак у кожному наборі даних про дефекти

програмного забезпечення) було прийнято відповідно до роботи, яка вказувала на те, що краще відсікати $\log_2 N$ ознак; багато емпіричних досліджень дотримувалися цієї процедури [32]. Згідно з таблицею 2.4, значення tC у цьому дослідженні становить ($tC \geq 21$), і це пов'язано з вибором різних наборів даних про дефекти програмного забезпечення для експерименту. Зрештою, генеруються зменшені дані в OERMS атрибутів.

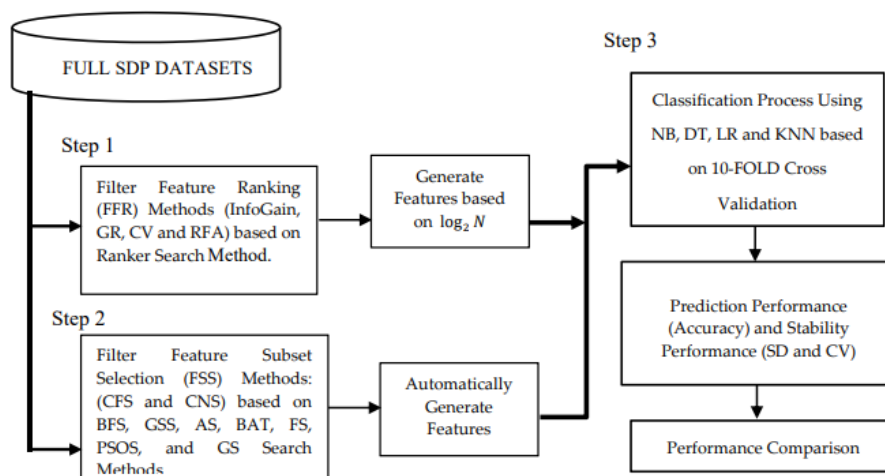


Рисунок 2.1 - Експериментальна установка.

Крок 2: По-друге, два методи FSS у таблиці 2.2 та відповідні методи пошуку у таблиці 3 застосовуються до вихідних повних наборів даних. Два методи FSS (CFS, CNS) з сімома методами пошуку (BFS, BAT, FS, AS, GSS, PSOS та GS), що утворює чотирнадцять методів FSS для застосування до набору даних. Кожен з FSS генерують підмножину ознак, які мають високий прогностичний потенціал. Відповідні налаштування параметрів методів пошуку, як показано в таблиці Було використано 3. Як і у випадку методів FFR, наприкінці цього кроку також генеруються скорочені набори даних.

Крок 3 : Цей крок включає процес прогнозування. Це застосування класифікаторів з Таблиці 2.4 до набору даних про дефекти програмного забезпечення (фільтровані набори даних з кроків 1 та 2). Чотири класифікатори (NB, DT, LR та KNN) були застосовані як до скорочених (фільтровані набори даних з кроків 1 та 2), так і до повних наборів даних. Суть цього кроку полягає в

тому, щоб показати ефективність скорочених функцій програмного забезпечення в SDP. Кожен з експериментів проводився з використанням методу 10-кратної перехресної перевірки (CV). Це зроблено для того, щоб уникнути проблеми зміщень та перенавчання моделей прогнозування, а також для зменшення впливу дисбалансу класів, що є однією з проблем якості даних в data mining [33]. Крім того, через випадковий характер методів пошуку методів FSS, кожен експеримент з використанням методів FSS проводився 10 разів для перевірки результату. Зрештою, у цьому дослідженні було проведено загалом 2900 ((4 методи FFR X 5 наборів даних X 4 класифікатори) + (14 методів FSS X 5 наборів даних X 4 класифікатори X 10 прогонів) + (4 класифікатори X 5 наборів даних)) різних експериментів.

2.2 Матеріали та методи дослідження дефектів програмного забезпечення

Дизайн моделі. ML є настільки значним досягненням у штучному інтелекті (ШІ), що очевидно, що модель для SDP, заснована на методах ML, необхідна для підтримки якості та економії коштів на тестуванні [33]. У літературі з використанням методологій ML ми знаходимо багато проблем, пов'язаних з прогнозуванням дефектів програмного забезпечення. На різних наборах даних автори використовували ряд алгоритмів ML [34]. Деякі з них є точними, тоді як інші відрізняються своєю продуктивністю та точністю. Метою цього дослідження є створення прототипу аналізу SDP шляхом економії коштів на тестуванні та одночасного підвищення точності запропонованої системи. Для цього ми аналізуємо різні методи ML з вибраними ознаками та кластеризацією для досягнення хорошої точності. Ми маємо намір досягти високої точності, використовуючи наші аналізовані методи, які використовують алгоритми ML, з використанням набору даних CM1, який має меншу точність у більшості методів.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

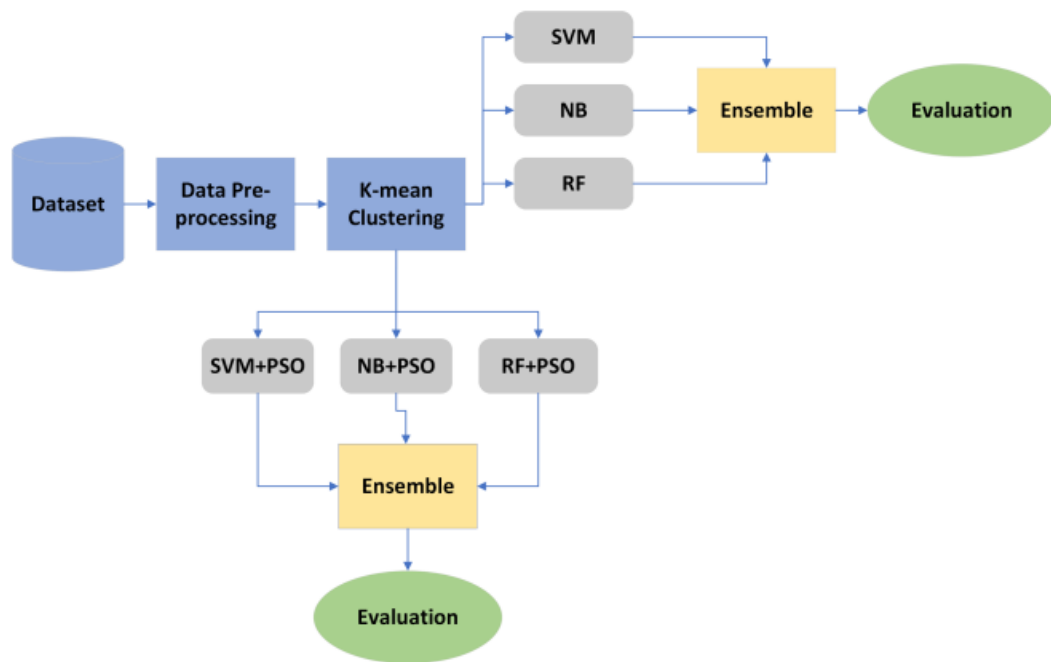


Рисунок 2.2 - Запропонований метод.

На рисунку 2.2 показано архітектуру запропонованої моделі. Ми розглянули різні відомі методи машинного навчання (ML) та оптимізували методи ML на вільно доступному наборі даних, щоб покращити точність наборів даних порівняно з попередніми дослідженнями. Для цього ми використали кластеризацію за методом К-середніх для категоризації позначок класів. Крім того, ми застосували моделі класифікації до вибраних ознак. Для оптимізації моделей ML було використано Particle Swarm Optimization. Ми оцінили продуктивність моделей за такими показниками, як точність, повнота, f-міра, метрики помилок продуктивності та матриця плутанини. Нижче ми детально обговоримо ці розділи рисунка. Метою вибору методів ML є підвищення точності цих добре зарекомендували себе методів ML, оскільки в літературі результати цих методів ML різняться та можуть бути додатково вдосконалені. Тому метою є покращення продуктивності або точності цих методів ML та аналіз результатів [35].

Набір даних

З репозиторію програмного забезпечення PROMISE , який знаходиться у вільному доступі та є частиною Програми даних метрик NASA (MDP). CM1 – це

інструмент космічного апарату NASA на основі мови C. У колекції є 498 екземплярів або модулів, а також 22 властивості, як зазначено в таблиці 2.5.

Таблиця 2.5

Параметри набору даних.

Метрика / Параметр	Значення показника
Назва датасету / Проекту (Title)	CM1
Мова програмування (Language)	C
Цільова система / Вихідний код (Source Code)	Космічний інструмент NASA (NASA Spacecraft Instrument)
Кількість модулів (Modules)	498
Кількість ознак / Атрибутів (Features)	22
Дефектні модулі (Defective)	49
Бездефектні модулі (Defect-Free)	449
Рівень дефектності (Defect Rate)	9.83%

Властивості набору даних цього дослідження базуються на чотирьох метриках Маккейба, дванадцяти вимірюваннях Холстеда та кількох інших метриках. Метрики Маккейба – це метрики рівня методу, що зосереджені на принципах програмування та легко збираються з вихідного коду. Інші метрики Холстеда – це числові дані, які можна легко зібрати за допомогою будь-якого програмного забезпечення [36]. Таблиця У розділі 2.6 наведено показники набору даних.

Таблиця 2.6

Опис атрибутів набору даних.

Категорія метрик	Скорочення / Ідентифікатор	Опис та розшифровка метрики
Метрики Маккейба	Loc	Кількість рядків коду (Line count of code)
	v(g)	Цикломатична складність (Cyclomatic complexity)
	ev(g)	Сутнісна / есенціальна складність (Essential complexity)
	iv(g)	Складність архітектури / дизайну (Design)
Метрики Голстеда	N	Загальна кількість операторів та операндів (Total operators + operands)
	V	Об'єм програми (Volume)

Продовження таблиці 2.6

	L	Довжина програми (<i>Program length</i>)
	D	Складність розуміння програми (<i>Difficulty</i>)
	I	Рівень інформативності / інтелект програми (<i>Intelligence</i>)
	E	Зусилля, необхідні для написання програми (<i>Effort to write program</i>)
	B	Очікувана кількість доставлених / наявних помилок (<i>Delivered bugs</i>)
	T	Оцінка часу, необхідного на розробку (<i>Time estimator</i>)
Інші структурні метрики	IOCode	Загальна кількість рядків безпосередньо коду (<i>Line count of code</i>)
	IOComment	Кількість рядків з коментарями (<i>Count of lines of comments</i>)
	IOBlank	Кількість порожніх рядків (<i>Count of blank lines</i>)
	IOCodeAndComment	Рівень рядків, що містять одночасно і код, і коментарі (<i>Lines of code and comments</i>)
	uniqOp	Кількість унікальних операторів (<i>Unique operators</i>)
	uniqOpnd	Кількість унікальних операндів (<i>Unique operands</i>)
	totalOp	Загальна кількість операторів (<i>Total operators</i>)
	totalOpnd	Загальна кількість операндів (<i>Total operands</i>)
	branchCount	Кількість розгалужень у графі потоку керування (<i>Flow of graph</i>)
	D	Бінарний прапорець: наявність дефектів у модулі — так/ні (<i>Module has defects or not</i>)

Методи. Попередня обробка даних

З аналізу набору даних зроблено висновок, що набір даних необхідно перетворити у стандартний формат перед застосуванням будь-яких моделей машинного навчання, оскільки в наборі даних є 498 кортежів та 22 ознаки. У кожному стовпці набору даних є широкий діапазон значень, наприклад, стовпець 'e' (e – зусилля, витрачені на написання програми) має максимальне значення 2 153 690,63 та мінімальне значення 0,0; крім того, стовпець 't' (t – час, витрачений на написання програми) має максимальне значення 119 649,48 та мінімальне значення 0,0. Стовпець 'I' (I – інтелект) має максимальне значення 1,30 та мінімальне значення 0,0. Між стовпцями та іншими стовпцями існує величезна різниця. Середнє значення стовпця 't' становить 1938,056124, а стандартне відхилення - 7453,591519, тоді як стовпець 'I' має середнє значення 0,146325 та стандартне відхилення 0,159337. Таким чином, для цієї мети в цьому дослідженні

для стандартизації набору даних використовується стандартний метод масштабування. Він упорядковує дані за стандартним нормальним розподілом.

Математично стандартний скаляр Z можна визначити як:

$$Z = \frac{x-u}{s} \quad (2.1)$$

де x – це спостереження, u – середнє значення навчальних вибірок, а стандартне відхилення навчальних вибірок дорівнює s . У наборі даних ми перевірили наявність нульових значень, але в жодному кортежі нульових значень не було.

Вибір функцій

Метод вибору ознак використовується для зменшення кількості ознак, що використовуються в навчанні та тестуванні прогностичної моделі. У цій роботі використовуються підхід фактора інфляції дисперсії та метод кореляції для визначення значущості значень та мультиколінеарності ознаки після підготовки набору даних [37]. Дві ознаки мають позитивну кореляцію, якщо збільшення значення однієї ознаки призводить до збільшення значення іншої ознаки, а зменшення значення однієї ознаки призводить до зменшення значення іншої ознаки. Якщо зміна значення однієї ознаки не впливає на значення іншої ознаки, ці дві ознаки не корелюють. Якщо збільшення значення однієї ознаки призводить до зменшення значення іншої ознаки або зменшення значення однієї ознаки призводить до збільшення значення ознаки, ці дві ознаки негативно корелюють [38]. На рисунку 2.3 показано кореляцію між усіма ознаками набору даних. Ми вибрали 10 найкращих атрибутів, які негативно пов'язані або не мають зв'язку на основі релевантності та високої значущості без позначок класів. Основними причинами зменшення кількості ознак є зменшення обчислювальних витрат та запобігання труднощам перенавчання для підвищення продуктивності моделі.

У цьому дослідженні ми використовували неконтрольоване машинне навчання (кластеризацію) для категоризації міток вихідних класів та

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

контрольоване машинне навчання (класифікацію) для цілей прогнозування.

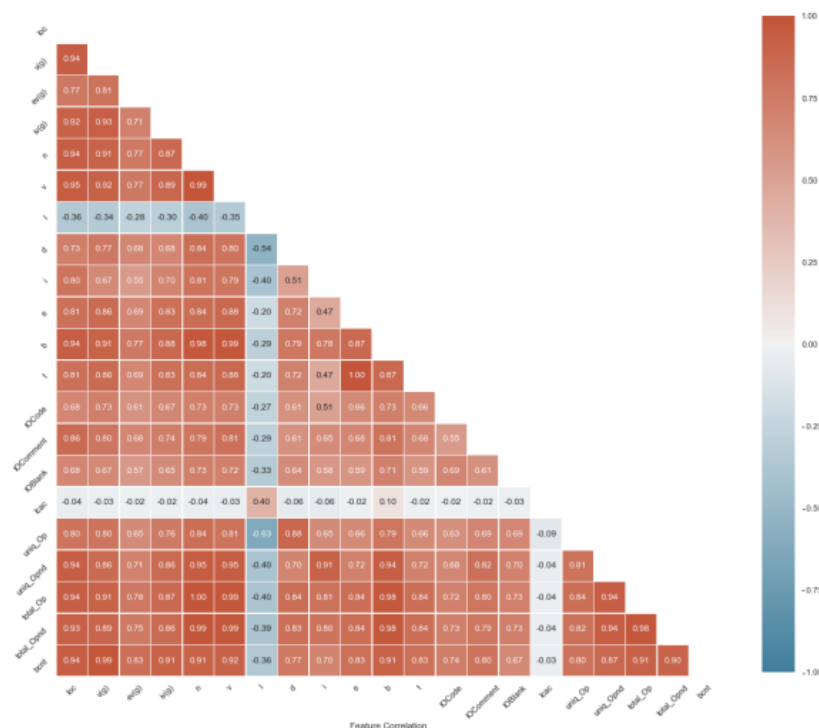


Рисунок 2.3 - Кореляція ознак

Класифікація

Далі, для визначення міток класів використовується кластеризація за методом К-середніх, а також інерція та метод ліктя для вибору найкращої кількості кластерів (K). У кластеризації за методом К-середніх кількість створених кластерів позначається символом K. Інерція – це метрика, яка показує, наскільки ефективно кластеризовано колекцію даних, і визначається шляхом піднесення відстані до квадрата між кожною точкою даних та її центроїдом, а потім додавання квадратів по всьому кластеру. Метод ліктя використовує ідею суми квадратів всередині кластера (WCSS). Для обчислення відстані між точками даних та центроїдом WCSS використовує формулу відстані. Тут значення K дорівнює 2, з інерцією 5947,92, як показано на рисунку 2.4.

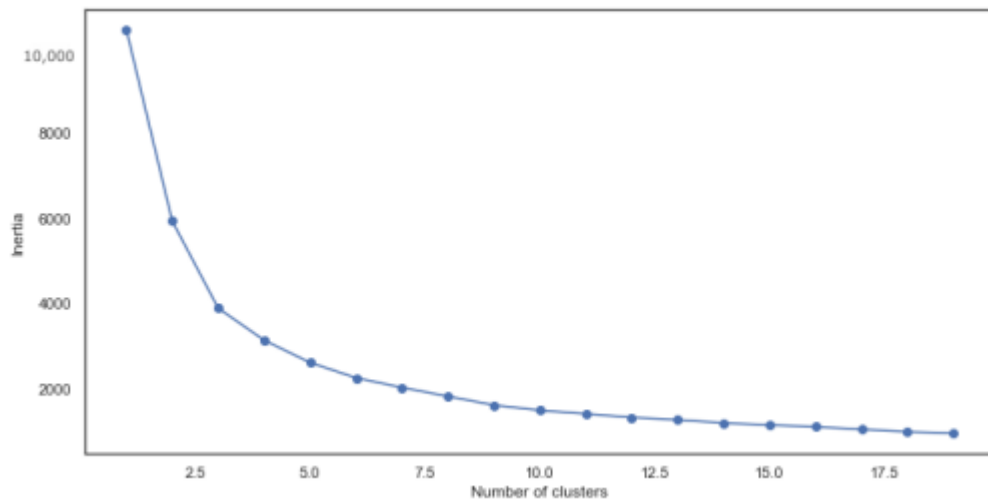


Рисунок 2.4 - Залежність інерції від кластера.

Класифікація

Контрольоване машинне навчання (ML) застосовується до даних з вихідними мітками класів. Набір даних складається з двох частин: навчання та тестування. Навчальний набір даних становить 67% від загальної кількості, тоді як тестовий набір даних – лише 33%. Спочатку даним надаються вихідні мітки класів для навчання, а потім для тестування надаються невидимі дані без вихідних міток класів. Для аналізу ми використовували три моделі класифікації ML, а саме SVM як лінійний класифікатор SVC, NB як гауссівський NB-класифікатор та RF як класифікатор випадкового лісу з ансамблевим методом як класифікатором стекування, щоб проаналізувати та об'єднати результати вищезазначених трьох моделей, використовуючи NB як базову модель та SVM та RF як моделі-члени. Кількість дерев у моделі RF становить 1000, а випадковий стан для всіх класифікаторів – 42. Потім класифікатори аналізуються з використанням методів PSO та без них. Кількість частинок під час експерименту підтримується на рівні 20.

SVM

SVM – це модель машинного навчання з учителем, яка здебільшого застосовується до даних з двома класами на виході [39]. Моделі SVM працюють краще з високою швидкістю, якщо набір даних має обмежений обсяг даних.

Модель SVM проводить межу прийняття рішень між даними для класифікації кожного класу. Найкраща гіперплощина в моделі SVM – це коли є максимальні запаси від обох класів. Модель SVM використовується в цій роботі для прогнозування мітки вихідного класу, і вона отримує найкращі результати. Випадковий стан – 42. Як класифікатор використовується лінійний SVC.

Примітка

Алгоритм NB – це метод категоризації на основі алгоритму Баєса. Він працює шляхом обчислення ймовірностей на основі правдоподібностей. Коли немає зв'язку між характеристиками набору даних, метод NB працює добре. Стан випадковості – 42. Класифікатор – це гауссівський NB.

RF – це модель класифікації, яка використовує поняття ансамблевого навчання, що передбачає поєднання численних класифікаторів для покращення результату. Модель RF складається з низки DT, які застосовуються до підмножин набору даних, а потім усереднюються для визначення показників продуктивності. Кількість використаних дерев має значний вплив на точність та інші показники. Однак покращення моделі стає постійним після певної кількості дерев. Знання правильної кількості дерев має вирішальне значення для цілей навчання. У цьому випадку використовується 1000 дерев, а випадковий стан дорівнює 42. Як класифікатор використовується випадковий ліс (Random Forest).

Ансамбль

У машинному навчанні ансамблевий метод поєднує численні моделі прогнозування для забезпечення найкращих результатів. В ансамблевому методі одна модель служить основою, а інші - членами. Ансамбль можна виконувати різними способами. Використовується підхід класифікації на основі стекування. Модель NB служить базовою моделлю, а моделі SVM та RF - моделями-членами. В ансамблевій моделі випадковий стан дорівнює 42.

Зрештою, ми проаналізували всі моделі, як без оптимізації, так і з оптимізацією. Для оцінки продуктивності моделі використовувалися точність, правильність, повнота, f-міра, міри помилок продуктивності та матриця плутанини.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3 Категорії алгоритмів аналізу та прогнозування дефектів програмного забезпечення

Методи машинного навчання є найпопулярнішими методами прогнозування дефектів [40]. З моменту розробки нових методів машинного навчання для побудови кращих моделей прогнозування дефектів були застосовані різні алгоритми, такі як словникове навчання, колективне навчання на основі представлень, багатоядерне ансамблеве навчання, глибоке навчання та трансферне навчання. З точки зору машинного навчання, ми приблизно поділяємо відповідну літературу з прогнозування дефектів на такі три категорії: методи з учителем, напів-керовані та некеровані методи. Методи навчання з учителем стосуються використання всіх позначених навчальних даних у проекті для побудови моделей прогнозування дефектів. Методи напів-керованого навчання будують моделі прогнозування дефектів, використовуючи лише невелику кількість позначених навчальних даних та велику кількість некерованих даних у проекті. Методи навчання без учителя не потребують позначених навчальних даних, вони безпосередньо використовують некеровані дані в проекті для навчання моделей прогнозування дефектів.

Контрольовані методи

Використовуючи нещодавно розроблений метод навчання словника, Розроблено підхід до дискримінативного навчання словника (CDDL), чутливий до витрат, для прогнозування дефектів програмного забезпечення. CDDL використовує інформацію про класи з історичних даних для покращення дискримінантної сили та призначає різні витрати на неправильну класифікацію, збільшуючи покарання за неправильну класифікацію II типу (тобто дефектний прогнозується як бездефектний) для вирішення проблеми дисбалансу класів.

Водночас, представили метод прогнозування дефектів (CSDP) на основі спільної класифікації представлень (CRC). CSDP використовує нещодавно запропонований метод CRC, основна ідея якого полягає в тому, що екземпляр може бути спільно представлений лінійною комбінацією всіх інших екземплярів.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

Представили метод прогнозування дефектів програмного забезпечення, який отримав назву багатоядерне ансамблеве навчання (MKEL). MKEL може краще представляти дані про дефекти у багатовимірному просторі ознак за допомогою багатоядерного навчання та зменшувати зміщення, спричинене класом без дефектів, шляхом складання серії слабких класифікаторів. Крім того, MKEL спеціально розробляє нову стратегію оновлення вектора ваг вибірки для вирішення проблеми дисбалансу класів.

Щоб подолати розрив між семантикою програм та функціями прогнозування дефектів, Автоматично вивчили семантичне представлення програм з вихідного коду, використовуючи методи глибокого навчання. Спочатку вони витягли абстрактні синтаксичні дерева програм, щоб отримати вектори токенів. На основі векторів токенів вони використали мережу глибоких довір для автоматичного вивчення семантичних ознак.

Використовуючи популярні методи глибокого навчання, Представили метод *Deeper* для прогнозування дефектів ЛІТ. *Deeper* спочатку використовує мережу глибоких довір для вилучення набору виразних метрик з початкового набору метрик змін, а потім навчає класифікатор за допомогою вилучених метрик для прогнозування дефектів.

Розробили новий алгоритм подвійного трансферного бустингу (DTB) для міжкомпанійного прогнозування. DTB спочатку використовує гравітацію даних для реконструкції розподілу міжкомпанійних (CC) даних, близьких до внутрішньокорпоративних (WC) даних. Потім DTB використовує метод навчання трансферним бустингом для видалення негативних випадків у даних CC, використовуючи обмежену кількість даних WC.

Розробили метод прогнозування дефектів між проектами, названий гібридним підходом реконструкції моделі (HYDRA), який містить дві фази: фазу генетичного алгоритму (ГА) та фазу навчання ансамблю. Після завершення цих двох фаз HYDRA створює масивну композицію класифікаторів, які можна застосовувати для прогнозування випадків дефектів у цільовому проекті.

З точки зору оптимізації, розглядали проблему прогнозування дефектів як

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

багатоцільову оптимізаційну задачу. Вони запропонували використовувати генетичний алгоритм (ГА) для навчання логістичної регресії та моделей дерев рішень, які називаються багатоцільовим предиктором дефектів (MODEP). MODEP дозволяє розробникам програмного забезпечення вибирати предиктори, що досягають певного компромісу, тобто вартості перевірки коду та кількості випадків, схильних до дефектів, або кількості дефектів, які може передбачити MODEP.

Представили метод бустингу, чутливого до вартості трансферу (TCSBoost), для вирішення проблеми дисбалансу класів для прогнозування дефектів між проектами. На основі характеристик розподілу TCSBoost призначає різні витрати неправильної класифікації для правильних/неправильних екземплярів класифікації в кожній ітерації алгоритму бустингу.

Використали методи дерева рішень та ансамбевого навчання для розробки двошарового методу ансамбевого навчання (TLEL) для прогнозування дефектів ЛІТ. TLEL спочатку будує модель випадкового лісу, поєднуючи дерево рішень та пакетування. Використовуючи алгоритм випадкової недовибірки, TLEL потім навчає кілька різних моделей випадкового лісу та знову збирає їх за допомогою ансамблю стекування.

Напівконтрольовані методи

Використовуючи та поєднуючи напівавторизоване навчання та ансамблеве навчання, представили метод невід'ємного розрідженого SemiBoost (NSSB) для прогнозування дефектів програмного забезпечення. З одного боку, NSSB краще використовує велику кількість немаркованих екземплярів та невелику кількість маркованих екземплярів завдяки напівавторизованому навчанню. З іншого боку, NSSB збирає низку слабких класифікаторів, щоб зменшити упередженість, спричинену бездефектним класом, завдяки ансамблевому навчанню.

Використовуючи методи напівавторизованого навчання на основі графів та навчання з розрідженим представленням, запропонували метод поширення міток на основі невід'ємних розріджених графів (NSGLP) для прогнозування

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

дефектів. NSGLP спочатку повторно вибірково аналізує позначені недефектні екземпляри для створення збалансованого навчального набору даних. Потім NSGLP будує ваги за допомогою алгоритму невід'ємних розріджених графів для кращого вивчення взаємозв'язку між даними. Нарешті, NSGLP ітеративно прогнозує немарковані екземпляри за допомогою алгоритму поширення міток.

Неконтрольовані методи

Забезпечення прогнозування дефектів для нових проектів або проектів без достатніх історичних даних є складною проблемою. Щоб вирішити це обмеження, представили кластеризацію екземплярів та маркування екземплярів у кластерах (CLA), а також кластеризацію екземплярів, маркування екземплярів, вибір метрик та вибір екземплярів (CLAMI) – два методи без нагляду. Ключова ідея цих двох методів полягає в маркуванні немаркованого набору даних за допомогою величини значень метрик. Отже, CLA та CLAMI мають переваги автоматизованого способу та відсутності потреби в ручних зусиллях.

З точки зору кластеризації даних, представили використання спектральної кластеризації (SC) для вирішення проблеми неоднорідності між вихідним проектом та цільовим проектом у прогнозуванні дефектів між проектами. SC – це метод неконтрольованої кластеризації на основі зв'язності, який не потребує жодних навчальних даних. Таким чином, SC не страждає від проблеми неоднорідності. Це перевага методів неконтрольованої кластеризації.

Короткий зміст: Порівнюючи та аналізуючи вищезазначені методи прогнозування дефектів на основі методів машинного навчання, ми можемо отримати такі спостереження: (i) Методи прогнозування дефектів зазвичай безпосередньо використовують або модифікують існуючі відомі алгоритми машинного навчання для різних контекстів прогнозування. (ii) Більшість методів прогнозування дефектів використовують методи навчання з учителем для побудови моделей класифікації. Як правило, методи прогнозування дефектів з учителем здатні покращити ефективність прогнозування, особливо точність класифікації. (iii) Більшість методів прогнозування дефектів базуються на програмних моделях моделювання (КМ), можливо тому, що їх легше збирати

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

порівняно з метриками процесу.

2.4 Висновок по розділу

У другому розділі розглянуто методи та інструменти аналізу дефектів програмного забезпечення. Визначено основні етапи методології аналізу дефектів, досліджено джерела даних та методи їх обробки. Особливу увагу приділено категоріям алгоритмів аналізу й прогнозування дефектів, включаючи статистичні методи та алгоритми машинного навчання. Результати дослідження показали, що використання сучасних методів аналізу даних дозволяє підвищити точність прогнозування дефектів і сприяти покращенню процесу тестування програмного забезпечення.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МЕТОДІВ АНАЛІЗУ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Підготовка та обробка даних для аналізу дефектів програмного забезпечення

У цьому розділі ми розглядаємо відповідні дослідження прогнозування дефектів з точки зору маніпулювання даними. Зокрема, ми порівнюємо та аналізуємо ці дослідження за такими трьома аспектами: атрибути прогнозування, використання даних та якість набору даних.

Атрибути для прогнозування

Нові показники програмного забезпечення: Такі показники програмного забезпечення, як вихідні СМ, відтік змін та кількість попередніх дефектів, активно вивчаються для того, щоб забезпечити прогнозування дефектів та полегшити забезпечення якості програмного забезпечення. Останнім часом існує кілька методів розробки нових показників програмного забезпечення для прогнозування дефектів [39].

Розглядаючи поведінку розробників, використали інформацію про взаємодію розробників для пропозиції метрик мікровзаємодії (МІМ). Вони окремо використовували МІМ, вихідні СМ та метрики історії змін для побудови моделей прогнозування дефектів та порівняли їхню ефективність прогнозування. Вони виявили, що МІМ значно покращують загальну точність прогнозування дефектів, поєднуючись з існуючими метриками програмного забезпечення, та добре працюють у економічно ефективному контексті. Отримані результати можуть допомогти розробникам програмного забезпечення краще виявляти власні неефективні моделі поведінки під час розробки програмного забезпечення.

Для вимірювання якості вихідного коду запровадили нову метрику під назвою «відсутність якості кодування» (LOCQ). Вони застосували баєсівські мережі для дослідження ймовірнісних впливових зв'язків між метриками

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

програмного забезпечення та схильністю до дефектів. Вони виявили, що відгук для класу, рядків коду та LOCQ є найефективнішими метриками, тоді як зв'язок між об'єктами, зважений метод для кожного класу та відсутність узгодженості методів є менш ефективними метриками для прогнозування дефектів.

Щоб дослідити прогностичну здатність прогнозування дефектів з урахуванням мутацій, визначено 40 метрик мутацій. Вони окремо побудували моделі прогнозування дефектів для порівняння ефективності цих метрик з урахуванням мутацій та 39 вихідних СМ (без усвідомлення мутацій), а також об'єднали їх разом. Вони виявили, що метрики з урахуванням мутацій можуть значно покращити ефективність прогнозування дефектів.

Щоб дослідити вплив проблем на якість програмного забезпечення, апроксимували проблеми програмного забезпечення як теми, використовуючи метод статистичного моделювання тем. Вони запропонували набір метрик на основі тем, включаючи кількість тем, кількість тем, схильних до дефектів, приналежність до тем та приналежність до тем, схильних до дефектів. Результати показують, що метрики на основі тем забезпечують додаткову пояснювальну силу порівняно з існуючими структурними та історичними метриками, що може допомогти краще пояснити дефекти програмного забезпечення.

Вибір функцій: Вибір ознак (метричний вибір) став предметом машинного навчання та аналізу даних, що також використовується в прогнозуванні дефектів програмного забезпечення [18]. Метою вибору ознак є вибір ознак, які є більш релевантними для цільового класу, з багатовимірних ознак та видалення ознак, які є надлишковими та некорельованими. Після вибору ознак покращиться ефективність класифікації моделей прогнозування.

Щоб перевірити позитивний вплив поєднання вибору ознак та ансамбового навчання на ефективність прогнозування дефектів [20] представили метод ансамблю середньої ймовірності (APE). APE може пом'якшити вплив, спричинений надмірністю метрик та дисбалансом даних, на ефективність прогнозування дефектів.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння методів прогнозування дефектів із застосуванням технологій
машинного навчання

Тип дослідження	Метод / Техніка	Набір даних (Датасет)	Рік
supervised (з вчителем)	навчання за словником, чутливе до витрат навчання (dictionary learning, cost-sensitive learning)	NASA	2014
	колаборативне представлення (collaborative representation)	NASA	2014
	ансамблеве навчання з кількома ядрами, бустинг (multiple kernel ensemble learning, boosting)	NASA	2016
	глибоке навчання (deep learning)	PROMISE	2016
	трансферне навчання, бустинг (transfer learning, boosting)	PROMISE	2015
	трансферне навчання, генетичний алгоритм (ГА), AdaBoost (transfer learning, GA, AdaBoost)	PROMISE	2016
	бустинг, чутливе до витрат навчання, трансферне навчання (boosting, cost-sensitive learning, transfer learning)	PROMISE	2017
	дерево рішень, беггінг, випадковий ліс (decision tree, bagging, random forest)	JIT	2017
semi-supervised (часткове)	SemiBoost, графове навчання, розріджене представлення (SemiBoost, graph learning, sparse representation)	NASA	2016
	розріджене представлення, розріджений граф, поширення міток (sparse representation, sparse graph, label propagation)	NASA	2017
unsupervised (без вчителя)	кластеризація, відбір ознак (cluster, feature selection)	ReLink	2015
	спектральна кластеризація (spectral clustering)	NASA, AEEEM, PROMISE	2016

Розроблено нову систему вибору ознак, яка отримала назву кластеризація ознак та ранжування ознак (FECAR). FECAR спочатку розділяє вихідні ознаки на k кластерів, а потім вибирає відповідні ознаки з кожного кластера. Результати показують, що FECAR ефективно вибирає ознаки для прогнозування дефектів.

Для вивчення методів вибору ознак з певною толерантністю до шуму, представили метод вибору ознак - кластеризацію ознак зі стратегіями вибору

(FECS). FECS містить дві фази: фазу кластеризації ознак та фазу вибору ознак. Вони виявили, що FECS ефективний як для вільних від шуму, так і для зашумлених наборів даних.

Представлено систему вибору ознак під назвою «максимальний інформаційний коефіцієнт з ієрархічною агломеративною кластеризацією» (MICNAS). MICNAS спочатку ранжує кандидатські ознаки, щоб відфільтрувати нерелевантні, використовуючи максимальний інформаційний коефіцієнт. Потім MICNAS групує ознаки за допомогою ієрархічної агломеративної кластеризації та видаляє надлишкові ознаки, вибираючи одну ознаку з кожної результуючої групи.

Короткий зміст: У таблиці 3.2 наведено порівняння досліджень прогнозування дефектів, що зосереджені на розробці нових метрик програмного забезпечення та пов'язаних з ними методів вибору ознак. Метрики програмного забезпечення дуже важливі для побудови моделей прогнозування дефектів. За останні десятиліття було запропоновано багато ефективних метрик, особливо СМ та метрики процесу. Для сприяння забезпеченню якості програмного забезпечення все частіше розробляються нові метрики прогнозування дефектів. Однак можуть існувати деякі надлишкові та нерелевантні метрики, які шкідливо впливають на ефективність прогнозування вивчених предикторів. Щоб вирішити цю проблему, можна використовувати методи вибору ознак для видалення цих надлишкових та нерелевантних метрик з метою покращення продуктивності.

Впровадження даних

Локальні та глобальні моделі: Першими представили локальні та глобальні моделі для прогнозування дефектів та оцінки зусиль. Локальні моделі стосуються кластеризації всього набору даних на менші підмножини з подібними властивостями даних та побудови моделей прогнозування шляхом їх навчання на цих підмножинах. Навпаки, глобальна модель навчається на всьому наборі даних.

Щоб дослідити ефективність трьох методів локальних, глобальних та багатовимірних адаптивних регресійних сплайнів (MARS), побудували для них

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

моделі прогнозування дефектів. Зокрема, вони побудували локальні моделі, розділивши дані на менші однорідні підмножини та вивчили кілька окремих статистичних моделей, по одній для кожної підмножини. Потім вони побудували глобальну модель на всьому наборі даних. Крім того, вони розглядали MARS як глобальну модель з локальними міркуваннями. З точки зору моделювання відповідності та прогностичної ефективності, вони виявили, що локальна модель може значно перевершувати глобальну модель. Для практичного застосування вони спостерігали, що MARS генерує загальні тенденції та може розглядатися як гібрид між глобальною та локальною моделями.

Проведено велике дослідження локальних моделей у контексті прогнозування дефектів між проектами. Вони оцінили та порівняли ефективність локальної моделі, глобальної моделі та методу перенесення навчання для прогнозування дефектів між проектами. Результати показують, що локальні моделі мають лише невелику різницю в ефективності прогнозування порівняно з глобальною моделлю та методом перенесення навчання для прогнозування дефектів між проектами.

Порівняли локальні та глобальні моделі прогнозування дефектів у контексті з урахуванням зусиль. Вони виявили, що принаймні одна локальна модель перевершує глобальну, але завжди існує інша локальна модель, яка працює гірше, ніж глобальна. Вони також зазначили, що гірша локальна модель навчається на тому, що підмножина даних має низький відсоток дефектів. На основі цих висновків вони рекомендували приділяти особливу увагу файлам меншого розміру в майбутніх дослідженнях прогнозування дефектів з урахуванням зусиль, і що корисно поєднувати переваги глобальних моделей, враховуючи локальні міркування на практиці.

Гетерогенні дані: Гетерогенне прогнозування дефектів стосується прогнозування схильності до дефектів екземплярів програмного забезпечення в цільовому проекті за допомогою гетерогенних метричних даних, зібраних з інших проектів. Це забезпечує нову перспективу прогнозування дефектів і отримало значний дослідницький інтерес. Нещодавно було запропоновано

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

кілька гетерогенних моделей прогнозування дефектів [16] для прогнозування дефектів у проектах з гетерогенними наборами метрик (тобто вихідні та цільові проекти мають різні набори метрик).

Таблиця 3.2

Порівняння досліджень з прогнозування дефектів, у яких розглядаються відповідні показники

Параметр Категорія	Дослідження 1	Дослідження 2	Дослідження 3	Дослідження 4
	Нові метрики	Нові метрики	Нові метрики	Нові метрики
Тема	Оцінка прогностичної здатності мікрометрик взаємодії	Оцінка метрики недостатньої якості коду	Оцінка прогностичної здатності мутаційних метрик	Оцінка пояснювальної здатності тематичних метрик (topic-based)
Метод / Техніка	Відбір підмножини ознак на основі кореляції, випадковий ліс (Random Forest)	Байєсівські мережі	Наївний Байєс, логістична регресія, випадковий ліс, J48	Латентне розміщення Діріхле (LDA), логістична регресія, кореляційний аналіз Спірмена
Датасет	Система Eclipse	PROMISE	3 реальні системи (real word systems)	4 реальні системи (real word systems)
Рік	2016	2014	2016	2017

Оскільки метрики, окрім загальних метрик, можуть мати сприятливу дискримінантну здатність, запропоновано метод прогнозування гетерогенних дефектів, а саме ССА+ (канонічний кореляційний аналіз), який використовує уніфіковане метричне представлення (UMR) та метод перенесення навчання на основі ССА. Зокрема, UMR складається з трьох типів метрик, включаючи загальні метрики вихідного та цільового проектів, метрики, специфічні для вихідного проекту, та метрики, специфічні для цільового проекту. Вивчаючи пару проєктивних перетворень, за яких кореляція вихідного та цільового проекту максимізується, ССА+ може зробити розподіл даних цільового проекту подібним до розподілу даних вихідного проекту.

Водночас, запропоновано інше рішення для прогнозування гетерогенних

дефектів. Спочатку вони застосували метод вибору метрик для видалення надлишкових та нерелевантних метрик для вихідного проєкту. Потім вони зіставили (наприклад, зіставлення на основі тесту Колмогорова-Смірнова, KSAnalysar) метрики вихідного та цільового проєктів на основі подібності метрик, таких як розподіл або кореляція. Після цих процесів вони нарешті отримали зіставлені набори метрик вихідного та цільового проєктів. З отриманими наборами метрик вони побудували гетерогенну модель прогнозування дефектів для прогнозування міток екземплярів у цілі.

Представлено CPDP з незбалансованими наборами ознак для вирішення проблеми гетерогенних наборів метрик у прогнозуванні дефектів між проєктами. Вони використовували вектори характеристик розподілу [13]. кожного екземпляра як нові метрики для забезпечення прогнозування дефектів.

Щоб вирішити проблему дисбалансу класів в умовах гетерогенного прогнозування дефектів між проєктами, Представили метод методу опорних векторів з перенесенням кореляції, чутливий до витрат (CCT-SVM), заснований на CCA+ [36]. Зокрема, щоб зменшити вплив незбалансованих даних, вони застосували різні витрати на неправильну класифікацію для дефектних та недефектних класів, включивши фактори вартості в модель методу опорних векторів (SVM).

Короткий зміст: У таблиці 3.3 показано порівняння досліджень прогнозування дефектів з акцентом на впровадженні даних. Локальні та глобальні моделі досліджують, як знаходити навчальні дані з подібним розподілом для тестових даних. Зазвичай це може досягти кращої продуктивності прогнозування для навчальних та тестових даних з подібним розподілом [12]. На практиці, маркування людиною великої кількості немаркованих модулів є дорогим та трудомістким, і може бути не ідеально точним. Отже, важко зібрати інформацію про дефекти для маркування набору даних для навчання моделі прогнозування. Гетерогенне прогнозування дефектів має хороший потенціал для використання всіх гетерогенних даних програмних проєктів для прогнозування дефектів у нових проєктах або проєктах з

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

обмеженими історичними даними про дефекти [18]. Однак основною проблемою гетерогенного прогнозування дефектів є подолання відмінностей у розподілі даних між вихідними та цільовими проектами. Таким чином, необхідно змінити розподіл вихідних даних, щоб він був подібним до цільових даних. Коротше кажучи, дуже корисно та цікаво розробляти високоякісні методи прогнозування гетерогенних дефектів для майбутніх досліджень.

Якість набору даних

Вирішення проблеми дисбалансу в класах: Набори даних про дефекти програмного забезпечення часто дуже незбалансовані [25]. Тобто, кількість дефектних екземплярів (меншість) зазвичай набагато менша, ніж кількість недефектних (більшість). Більшості звичайних алгоритмів класифікації важко працювати з даними, які мають незбалансований розподіл класів. Незбалансований розподіл може призвести до неправильної класифікації екземплярів у класі меншості, і це важливий фактор, що пояснює незадовільну ефективність прогнозування [1]. Останнім часом все більше дослідників звертають увагу на проблему дисбалансу класів при прогнозуванні дефектів [29].

Щоб одночасно вирішити проблему дисбалансу класів всередині проекту та між проектами, розробили єдину систему прогнозування дефектів. Спочатку вони використали вдосконалений дискримінантний аналіз підкласів (ISDA) для досягнення збалансованих підкласів для вирішення проблеми класифікації незбалансованих даних всередині проекту. Потім вони зробили розподіли вихідних та цільових даних подібними за допомогою напівконтрольованого аналізу компонентів передачі (SSTCA) та поєднали SSTCA з ISDA, щоб запропонувати підхід SSTCA + ISDA для навчання дисбалансу класів між проектами.

Щоб вирішити проблему дисбалансу класів у контексті прогнозування дефектів змін в режимі онлайн, запропонували використовувати методи повторної вибірки та оновлюваної класифікації. Вони застосували кілька методів повторної вибірки, включаючи просте дублювання, техніку синтетичної меншості з надмірною вибіркою (SMOTE), розсіяну підвибірку, повторну

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

вибірку з/без замін, щоб збільшити відсоток дефектних екземплярів у навчальному наборі для вирішення проблеми незбалансованих даних.

Досліджено можливість використання навчання дисбалансу класів для прогнозування дефектів між проектами. Вони розробили модель на основі бустингу, яка називається когнітивним бустингом за значеннями з машиною опорних векторів (VCB-SVM). Вона встановлює ваги подібності відповідно до характеристик розподілу та поєднує ваги з асиметричною вартістю неправильної класифікації, розробленою алгоритмом бустингу, для відповідної повторної вибірки.

Використовуючи методи зменшення перекриття класів та навчання ансамблевого дисбалансу, представили новий метод прогнозування дефектів програмного забезпечення, який називається навчання з очищенням сусідів (NCL) + ансамблева випадкова недостатня вибірка (ERUS). З одного боку, вони усунули перекриваючі недефектні екземпляри за допомогою алгоритму NCL. З іншого боку, вони повторно вибірково вибрали недефектні екземпляри для побудови збалансованих навчальних підмножин за допомогою алгоритму ERUS. Нарешті, вони зібрали серію класифікаторів для побудови моделі прогнозування ансамблю за допомогою адаптивного бустингу (AdaBoost).

Представлено метод локального колаборативного представлення (CLCR) з урахуванням вартості для прогнозування дефектів. CLCR досліджує інформацію про сусідство серед екземплярів програмного забезпечення для покращення прогностичної здатності моделі та включає різні фактори вартості для дефектних та недефектних класів для вирішення проблеми дисбалансу класів.

Представлено новий двоетапний метод навчання, чутливий до витрат, для прогнозування дефектів, який використовує інформацію про вартість як на етапі вибору ознак, так і на етапі класифікації. На першому етапі вони розробили три алгоритми вибору ознак, чутливі до витрат, використовуючи різні витрати на неправильну класифікацію. На другому етапі вони застосували алгоритм класифікації на основі зворотного поширення нейронної мережі, чутливий до

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

витрат, зі стратегією переміщення порогу.

Таблиця 3.3. Порівняння досліджень з прогнозування дефектів, зосереджених на використанні даних

Категорія	Тема	Метод / Техніка	Набір даних (Датасет)	Рік
local versus global models (локальні моделі проти глобальних)	порівняння здатності підгонки (відповідності) локальних і глобальних моделей (fit ability comparison of local and global models)	кластеризація, MARS s, лінійна регресія (clustering, MARS s, linear regression)	шість проєктів із відкритим вихідним кодом (six open source projects)	2015
	порівняння прогностичної здатності локальних і глобальних моделей у контексті крос-проєктного аналізу (predictive ability comparison of local and global models in a cross-project context)	кластеризація, метод опорних векторів (clustering, support vector machine)	NASA, AEEEM, PROMISE	2017
	порівняння прогностичної здатності локальних і глобальних моделей (predictive ability comparison of local and global models)	кластеризація k-медоїдів, спектральна кластеризація (k-medoids clustering, spectral clustering)	AEEEM, PROMISE	2016
heterogeneous defect prediction models (гетерогенні моделі прогнозування дефектів)	метод вирішення проблеми гетерогенних метрик (a method to solve heterogeneous metric problem)	канонічний кореляційний аналіз, трансферне навчання (canonical correlation analysis, transfer learning)	NASA, SOFTLAB, ReLink, AEEEM	2015
	метод вирішення проблеми гетерогенних метрик (a method to solve heterogeneous metric problem)	відбір метрик, зіставлення метрик, трансферне навчання (metric selection, metric matching, transfer learning)	NASA, SOFTLAB, ReLink, AEEEM, PROMISE	2015
	доцільність використання різних метрик (feasibility using different metrics)	статистика, логістична регресія (statistics, logistic regression)	ReLink, AEEEM, PROMISE	2014

	покращений метод вирішення проблеми гетерогенних метрик (an improved method to solve heterogeneous metric problem)	канонічний кореляційний аналіз, трансферне навчання, метод опорних векторів (canonical correlation analysis, transfer learning, support vector machine)	NASA, SOFTLAB, ReLink, AEEEM	2016
--	--	---	------------------------------	------

Обробили незбалансовані дані, порівнюючи та оцінюючи різні типи алгоритмів для прогнозування дефектів програмного забезпечення. Вони окремо використовували вибірккові, економічно чутливі ансамблі та їх гібридні форми для побудови моделей прогнозування. Результати показують, що ці методи можуть покращити правильність класифікації класу дефектів, а попередня обробка даних впливає на покращення моделі прогнозування.

Обробка шумів: Інформація про дефекти відіграє важливу роль у підтримці програмного забезпечення, такій як вимірювання якості та прогнозування дефектів. Оскільки поточні методи збору інформації про дефекти базуються на необов'язкових ключових словах для виправлення помилок або посиланнях на звіти про помилки в журналах змін, зібрані дані про дефекти з журналів змін та звітів про помилки можуть містити шуми [14]. Ці упереджені дані про дефекти впливатимуть на ефективність прогнозування дефектів.

Щоб дослідити характеристики неправильного маркування, вплив реалістичного неправильного маркування на ефективність прогнозування та інтерпретацію моделей дефектів, провели поглиблене дослідження 3931 вручну курованого звіту про проблеми з двох великих систем з відкритим кодом для прогнозування дефектів. Вони виявили, що неправильне маркування звітів про проблеми не є випадковим і рідко впливає на точність моделей прогнозування дефектів, а також не має значного впливу на найвпливовіші показники.

Досліджено вплив змін у коді, пов'язаних зі сплутаними елементами, на моделі прогнозування дефектів. Вони виявили, що до 20% усіх змін, пов'язаних з виправленням помилок, склалися з кількох сплутаних елементів, що суттєво впливало на підрахунок помилок та моделі прогнозування помилок. Через

					БР.ІП – 08.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

сплутані зміни щонайменше 16,6% усіх вихідних файлів неправильно пов'язані зі звітами про помилки для прогнозування файлів, схильних до помилок. Експериментальні результати на наборах даних про сплутані помилки показують, що розплутування змін у коді, пов'язаних зі сплутаними елементами, може призвести до створення точніших моделей. Вони рекомендували, щоб у майбутніх тематичних дослідженнях досліджувалася краща організація змін для зменшення впливу сплутаних змін.

Обмін даними із збереженням конфіденційності: Через проблеми з конфіденційністю більшість компаній не бажають ділитися своїми даними. З цих причин багато дослідників сумніваються в практичності обміну даними для дослідницьких цілей. Нещодавно питання збереження конфіденційності досліджувалося в деяких програмах програмної інженерії, наприклад, прогнозування дефектів програмного забезпечення, оцінка зусиль розробки програмного забезпечення [15].

Щоб вирішити проблему збереження конфіденційності в багатосторонньому сценарії для прогнозування дефектів між проектами, представили LACE2 (середовище забезпечення конфіденційності великомасштабного масштабування). LACE2 поєднує алгоритми CLIFF, LeaF та MORPH. За допомогою LACE2 власники даних можуть поступово додавати дані до приватного кешу та вносити «цікаві» дані, які не схожі на поточний вміст приватного кешу. Результати показують, що LACE2 може забезпечити вищу конфіденційність та кращу продуктивність прогнозування дефектів.

Короткий зміст: Для кращого пояснення та порівняння відмінностей між цими дослідженнями прогнозування дефектів ми наводимо короткий зміст. Вищезгадані дослідження прогнозування дефектів в основному зосереджуються на якості набору даних, включаючи дисбаланс класів, шум даних та проблеми обміну даними, що зберігають конфіденційність. Ці дослідження досліджують різні методи моделювання та надають деякі глибокі наслідки. Як згадувалося вище в Розділі 3, деякі методи на основі машинного навчання також розглядаються як проблема дисбалансу класів з використанням різних типів

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

методів, які включають економічно чутливе навчання, ансамблеве навчання та алгоритми вибірки. Можна побачити, що вибірка даних, ансамблеве навчання, економічно чутливе навчання та їхні гібридні методи зазвичай використовуються для вирішення проблеми дисбалансу класів у галузі прогнозування дефектів. Крім того, більшість досліджень прогнозування дефектів використовують методи вибірки, можливо тому, що вони прості, ефективні та легкі в реалізації. Для обробки шумів, набір даних без шуму дуже важливий для побудови ефективних моделей прогнозування дефектів. Дані містять шуми, що має великий вплив на ефективність прогнозування дефектів. Через проблеми збереження конфіденційності більшість власників даних не бажають ділитися своїми даними, що обмежує нас у використанні цих даних для прогнозування дефектів між проектами, особливо для нових проектів або проектів без достатніх історичних даних про дефекти. Отже, дуже необхідно ретельно дослідити якість набору даних для майбутніх досліджень прогнозування дефектів.

3.2 Реалізація методів вибору ознак для аналізу та прогнозування дефектів програмного забезпечення

Методи вибору ознак використовуються в машинному навчанні та аналізі даних для вибору підмножини відповідних ознак або змінних з більшого набору доступних ознак. Ці методи спрямовані на покращення продуктивності моделі, зменшення перенавчання та скорочення часу навчання шляхом вибору найбільш інформативних та важливих ознак [33]. Дослідники запропонували різні методи ФС для покращення продуктивності розглянутого методу [26]. У наступних - підрозділах наведено вичерпний огляд методів фільтрації, обгортки, гібридних та вбудованих ФС, що висвітлює, як кожна з цих категорій використовувалася в первинних дослідженнях у галузі прогнозування помилок програмного забезпечення.

Представлено метод FS, що складається з трьох алгоритмів. У першому

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

алгоритмі певна підмножина атрибутів була обрана з усього пулу атрибутів з усіма можливими попарними комбінаціями. Після комбінації атрибути були переупорядковані на основі їх частоти, причому атрибути з вищою частотою вважалися більш важливими. Другий алгоритм допоміг вибрати атрибути-кандидати. Нарешті, третій алгоритм визначив найкращу підмножину атрибутів, тестуючи всі можливі попарні комбінації та вибираючи найкращу. Експериментальні результати показали, що FS, що базується на даних, працює краще, навіть з простим класифікатором. Запропонували нову структуру FS, засновану на кластеризації ознак та ранжуванні ознак, яка називається FECAR. Структура складалася з двох фаз: кластеризація ознак та вибір ознак. На етапі кластеризації ознак для розділення ознак на кластери використовувався алгоритм кластеризації k-medoids, заснований на мірі кореляції ознак (FF). На етапі вибору ознак відповідні ознаки з кожного кластера були обрані на основі мір FCR для створення остаточної підмножини ознак. У дослідженні симетрична невизначеність використовувалася як міра кореляції FF, а коефіцієнт приросту інформації, χ^2 -квадрат та ReliefF розглядалися як міри релевантності FC. Результати показали, що фреймворк FECAR ефективно знизив коефіцієнт надмірності та покращив роботу предикторів помилок.

Спробували покращити точність SDP на основі метрик, впровадивши алгоритм ReliefF-LC, який поєднував вибір ознак ReliefF та кореляційний аналіз. Цей інноваційний підхід ефективно вирішив проблему надлишкових метрик та їх негативного впливу на точність прогнозування. Експерименти проводилися на двох наборах даних Національного управління з аеронавтики та дослідження космічного простору (NASA), і запропонований алгоритм порівнювали з двома іншими методами FS. Експериментальні результати показали чудовий потенціал запропонованого алгоритму для значного покращення ефективності прогнозування несправностей. Було представлено мережу атрибутів програмного забезпечення, що демонструє наявність взаємної інформації між ознаками. Було запропоновано метод вибору атрибутів на основі кластеризації. Дослідники використовували коефіцієнт максимальної інформації (MIC) для

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

визначення кореляцій атрибутів у мережі. Це лягло в основу вибору підмножин ознак за допомогою таких методів, як спектральна кластеризація, ієрархічна кластеризація, кластеризація k-середніх та вибір ознак на основі F-оцінки. Спектральна кластеризація на основі вибору ознак показала кращу продуктивність у SDP.

Запропоновано оптимізовану модель для прогнозування несправностей з використанням статистичного контролю процесів та методу FS. У методі FS релевантність різних атрибутів аналізувалася за допомогою таких методів, як кореляція та дисперсійний аналіз (ANOVA). Кореляція оцінювала залежність між атрибутами та їх близькість, тоді як ANOVA оцінював дисперсію атрибутів у межах класів та між ними. Застосовуючи ці методи, були виявлені та відфільтровані нерелевантні або надлишкові атрибути, що призвело до оптимізованого набору даних для SDP. Запропонували новий метод FS, що використовує кластери гібридних даних для прогнозування відмов між проектами (CPDP). Підхід включав дві фази: фазу кластеризації ознак з використанням методу кластеризації на основі щільності, який називається DPC, та фазу вибору ознак з використанням трьох стратегій ранжування: локальна щільність ознак (LDF), подібність розподілів ознак (SFD) та релевантність класу ознак (FCR). Експериментальні дослідження реальних програмних проектів показали, що запропонований метод працює краще, ніж базові методи, і залишається стабільним у різних класифікаторах. Автори [39] запропонували метод FS, заснований на мірі подібності (SM). Метою було вилучення релевантних ознак з урахуванням відмінностей у розподілі вибірок у різних класах (дефектних чи недефектних). Підхід включав оновлення ваг ознак на основі подібності вибірок, специфічної для класу, ранжування ознак, а потім послідовну оцінку підмножин ознак за допомогою моделі k-найближчих сусідів (KNN) з метрикою площі під кривою (AUC) для ефективності класифікації. Запропонований метод порівняли з іншими підходами FS на одинадцяти наборах даних NASA, продемонструвавши його ефективність, особливо на великих та незбалансованих наборах даних.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

Запропоновано новий метод FS, який використовує кластери гібридних даних для прогнозування невдач між проектами (CPDP). Підхід включав дві фази: фазу кластеризації ознак з використанням методу кластеризації на основі щільності, який називається DPC, та фазу вибору ознак з використанням трьох стратегій ранжування: локальної щільності ознак (LDF), подібності розподілу ознак (SFD) та релевантності класу ознак (FCR). Експериментальні дослідження реальних програмних проектів показали, що запропонований метод перевершує базові методи та залишається стабільним для різних класифікаторів. Досліджено різні методи FS та екстракції. Методи використовували вибір ознак на основі кореляції (CFS), аналіз головних компонентів (PCA) та аналіз головних компонентів ядра (KPCA). Результати показали, що SVM з методом вибору ознак CFS забезпечив найкращу точність прогнозування порівняно з PCA + SVM та KPCA + SVM. Запропоновано метод, який поєднує генетичний алгоритм (GA) для оптимізації ознак з глибокою нейронною мережею (DNN) для класифікації. Географічний алгаритм (ГА) був покращений завдяки новому дизайну хромосом та новій методиці розрахунку функції придатності. Експериментальні результати показали, що запропонований метод перевершує існуючі методи та досяг значної точності класифікації. Досліджено підходи до вибору підмножини ознак та ранжування ознак з метою покращення продуктивності CPD (неперервного професійного розвитку). Були досліджені два методи FS: перший, вибір підмножини ознак, в якому підмножина найважливіших ознак була обрана за допомогою таких методів, як вибір ознак на основі кореляції. Другий метод - це ранжування ознак, в якому ознаки ранжуються за їхньою важливістю за допомогою таких методів, як SM, кореляція та коефіцієнт посилення. Ці методи FS були застосовані до початкових проектів для забезпечення узгодженості наборів ознак. Дослідження показало, що обидва підходи FS покращили точність CPDP, що свідчить про те, що вибір відповідних ознак або їх ранжування може покращити прогнозування в різних проектах.

Запропоновано метод FS з використанням асоціативного аналізу правил (ARM). Процес включав аналіз історичних даних програмного забезпечення, які

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

включали метрики, пов'язані зі складністю програмного забезпечення, розміром коду та іншими відповідними атрибутами. Метод ARM потім був використаний для виявлення закономірностей або правил, які вказували на кореляції між певними метриками та виникненням помилок. Результати емпіричних оцінок показали, що запропонований метод, що поєднує NB з вибором ознак на основі ARM, забезпечує вищі показники точності, повноти, F-міри та точності порівняно з традиційними методами. Запропоновано метод FS, заснований на випадковому виборі. Метою було визначити компактний набір ознак, який дозволив би ефективно прогнозувати дефекти та таким чином зменшити зусилля та вартість збору даних для моделей прогнозування дефектів.

Алгоритм вибору підмножини ознак (RSFS) перевершив інші алгоритми за продуктивністю. Представлено фреймворк, який дозволяє методам контролю фільтрації ознак включати інформацію про набір ознак із зовнішніх джерел знань. Фреймворк був застосований для покращення алгоритму мінімальної надлишковості та максимальної релевантності (MRMR), що призвело до створення алгоритму групування MRM R, який був розроблений для досягнення вищої точності класифікації шляхом сприяння вибору ознак з різних груп ознак. Запропонований фреймворк показав значне покращення точності порівняно з традиційним алгоритмом MRMR та іншими поширеними методами фільтрації. Досліджували взаємодію між методами FS та методами вибірки та аналізували вплив на точність моделей прогнозування для SDP. Вони використовували три специфічні методи: χ^2 -квадрат, інформаційний коефіцієнт посилення та рельєф. Ці методи оцінювали релевантність кожної ознаки в наборі даних. Дослідження показало, що при використанні χ^2 -квадрат та інформаційного коефіцієнта посилення для FS, вигідно спочатку виконати вибірку для методів AUC та повного відтворення. При використанні рельєфу разом з NB, LR та SVM, вигідно спочатку виконати FS для покращення AUC та повного відтворення. Послідовність цих трьох методів визначення ознак та методів вибірки також призвела до різних результатів з точки зору точності, прецизійності та F1.

Представлено концепцію багатофільтрового FS на основі рангової

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

агрегації. Вони використали три різні методи FS, включаючи χ^2 -квадрат, ReliefF та інформаційний приріст, для підвищення точності моделей SDP. У цьому підході окремі ранжовані списки були створені з використанням різних методів фільтрації, а потім агреговані за допомогою функції агрегації для середнього рангу. Отриманий агрегований список був додатково уточнений за допомогою функції середнього геометричного для вибору оптимальних ознак. Експериментальні результати показали, що запропонований метод покращив продуктивність. Застосували метод FS на основі F-значення ANOVA. Цей метод був використаний для визначення найкращих дискримінативних ознак для прогнозування несправностей у програмних модулях. Запропонований метод порівнювали з сучасними методами, що використовують точність, AUC та повноту на 11 наборах даних. Застосування цього методу допомогло подолати проблеми роботи з даними з високою асиметрією та досягти більш точного та ефективного автоматичного прогнозування несправностей програмного забезпечення. Була запропонована вдосконалена модель CNN під назвою CNN на основі метрик (MB-CNN). Зокрема, був реалізований метод χ^2 -квадрат FS. Метод χ^2 -квадрат оцінює статистичну залежність між кожною метрикою та цільовою змінною, якою в цьому випадку була кількість помилок у програмному модулі. Метрики, які показали вищий ступінь залежності, вважалися більш релевантними та були обрані для навчання моделі MB-CNN. Результати показують, що запропонована модель MB-CNN є перспективною для прогнозування несправностей.

Представлено метод трансформації та FS під назвою TFSCPDP. На етапі трансформації ознаки вихідного проекту використовувалися для трансформації цільового проекту. На етапі вибору ознак відповідні ознаки вибиралися на основі відстаней до трансформованого джерела. Результати показали, що TFSCPDP ефективно зменшив відмінності в розподілі та проблеми з високовимірними ознаками в CPDP та досяг середнього F-оцінки 0,8, найвищого показника серед існуючих підходів. Запропоновано модель на основі LASSO-SVM. На етапі попередньої обробки було реалізовано метод FS, стиснення та вибір

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

мінімального абсолютного значення, для зменшення розмірності вихідного набору даних та виключення даних, які не були релевантними для SDP. Цей метод сприяв більш цілеспрямованому та точному аналізу помилок програмного забезпечення, що зрештою призвело до більш ефективного та результативного процесу прогнозування. Застосували алгоритм методу найменших квадратів опорних векторів (LSSVM) з вибором ознак ReliefF та методом SMOTE для вирішення проблем із незбалансованими класами в наборах даних SDP. Метод ReliefF FS був реалізований для вирішення проблеми визначення відповідних атрибутів. Основним принципом ReliefF було оцінити, чи є ознака значущою, шляхом аналізу того, як її значення відрізняються в межах класів та між ними. Якщо значення подібні в межах одного класу, але відрізняються між класами, атрибут вважався цінним. Дослідження показало позитивний вплив методу ReliefF на ефективність класифікації.

Досліджували, як FS та ансамблеве навчання впливають на точність прогнозування помилок. Вони запровадили метод ансамблю середньої ймовірності (APE) для класифікації помилок. На етапі попередньої обробки ознаки були відібрані за допомогою жадібного прямого відбору (GFS) та FS на основі кореляції (CFS). Для створення моделі APE було використано сім базових класифікаторів: випадковий ліс (RF), градієнтне підвищення (GB), стохастичний градієнтний спуск (SGD), зважена машина опорних векторів, логістична регресія (LR), багатоміальний наївний баєсівський метод (MNB) та наївний баєсівський метод Бернуллі (BNB). Модель була протестована на шести наборах даних, і результати показали, що GFS працює краще, ніж CFS. Проаналізували продуктивність моделей SDP за допомогою методів FS, зосередившись на тому, чи підмножина ознак покращує точність прогнозування. Вони провели комплексне дослідження шістнадцяти проектів з відкритим кодом з десятима популярними алгоритмами FS, включаючи алгоритм прямого жадібного пошуку (Forward Greedy Algorithm), алгоритм зворотного жадібного пошуку (Backward Greedy Algorithm), індекс Джині (Index Gini), інформаційний приріст (Information Gain), алгоритм на основі рельєфу (Relief-based Algorithm), CFS (Common Field

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Regression-based Feature Selection - BLogReg), алгоритм ранжування (Ranker Algorithm), алгоритм OneR (OneR Algorithm) та алгоритм підмножини-обгортки (Wrapper Subset Algorithm). Експериментальні результати показали, що методи FS, особливо прямий та зворотний жадібні алгоритми, послідовно забезпечували найкращі результати в усіх проектах.

Представили комплексний процес вибору атрибутів, що складається з п'яти послідовних кроків. Спочатку вони обчислили баланс кожного атрибута за допомогою базового класифікатора, а потім ранжували атрибути у списку на основі їхніх відповідних значень балансу. Потім були сформовані попарні комбінації всіх атрибутів, і для кожної комбінації був обчислений баланс. Ці комбінації були точно відфільтровані, вибираючи лише комбінації з високим балансом атрибутів для створення каталогу потенційних комбінацій після вибору атрибутів-кандидатів. Останній крок включав визначення найкращого набору атрибутів. Це включало визначення набору атрибутів, які забезпечили найсприятливіший результат балансу для моделі SDP. Запропонований підхід показав покращення продуктивності до 54% у прогнозуванні помилок програмного забезпечення. Запропонували новий метод, що використовує процес перетворення простору ознак у поєднанні з методом нормалізації. Метод вибору ознак включав перетворення просторів ознак, а потім класифікацію екземплярів за допомогою методу опорних векторів (SVM) з ядром перетину гістограм (HI). Запропонований багатоетапний процес починався з обчислення балансу кожного атрибута за допомогою базового класифікатора та відповідного сортування. Потім були згенеровані потенційні пари атрибутів, розрахований їх баланс та відібрані ті, що мають високий баланс. Відповідні атрибути потім були відібрані на основі їх частоти та вагової складової, а найкращий набір атрибутів був визначений шляхом ітеративної оптимізації балансу. За допомогою цього підходу найбільш релевантні атрибути були систематично ідентифіковані та збережені, що підвищило точність прогнозування несправностей. Запропонували нову структуру FS під назвою MICHAS (Максимальний інформаційний коефіцієнт з ієрархічною агломеративною кластеризацією), яка

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

складається з двох основних кроків: використання МІС для ранжування та виключення нерелевантних ознак та застосування ієрархічної агломеративної кластеризації (НАС) для групування та видалення надлишкових ознак. Експерименти на наборах даних NASA та AEEEM показали ефективність МІСНАС у покращенні прогнозування дефектів за різними класифікаторами та показниками продуктивності.

Розроблено фреймворк з використанням FS на основі п'яти класифікаторів, включаючи IBk, KStar, локально зважене навчання (LWL), випадкове дерево (RT) та RF. Метод FS використовував оцінку атрибутів χ^2 -квадрат та оцінку атрибутів кореляції разом з методами Best First, Greedy Stepwise та НАС для оптимізації підмножин атрибутів та усунення надлишковості. Нове додавання атрибутів, що оцінюються за МІС, призвело до підвищення точності системи. Запропонували метод, який поєднує алгоритм пошуку на основі Bat (BA) для вибору ознак та алгоритм RF для прогнозування. FS поверх BA мав на меті усунути некорельовані ознаки та зберегти значущі ознаки. Алгоритм CFS був інтегрований у процес вибору ознак. CFS оцінює підмножини ознак на основі їх кореляції з мітками класів та їх взаємкореляції одна з одною. Вибрана підмножина ознак була результатом комбінації BA та CFS. Запропонований підхід призвів до більш точної ідентифікації несправних програмних модулів. Поєднали метод інтегрованої вибірки з FS, що ефективно вирішує проблеми дисбалансу класів та релевантності атрибутів у SDP. Були використані три алгоритми вибору атрибутів: χ^2 -квадрат (CS), інформаційний приріст (IG) та рельєф (RLF). CS оцінював релевантність атрибутів; IG оцінював важливість атрибутів на основі інформаційного приросту, тоді як RLF визначав релевантні атрибути шляхом порівняння найближчих сусідів одного та різних класів. Інтеграцію методу синтетичної передискретизації меншин (SMOTE) з методом рельєфу було застосовано до наївного байєсівського класифікатора (NB). Результати дослідження показали, що інтегрований підхід перевершив інші методи та досяг покращеної точності прогнозування на 82%.

Розроблено паралельний гібридний фреймворк для ефективної обробки

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

великих наборів даних метрик програмного забезпечення в хмарі. Використаний метод FS був гібридним підходом, який поєднував методи фільтрації та обгортки. Фільтр Фішера та фільтр максимальної релевантності (MR) використовувалися для призначення ваг метрикам програмного забезпечення на основі їх релевантності для прогнозування несправностей. Ці фільтри оцінювали дискримінаційну силу та кореляцію кожної метрики з мітками класів. Метрики, вибрані з цих методів фільтрації, потім інтегрувалися в обгортку на основі штучної нейронної мережі (ANN), яка оцінювала різні підмножини метрик шляхом навчання та тестування моделі ANN для прогнозування несправностей. Цей гібридний підхід покращив точність прогнозування несправностей шляхом вибору важливих метрик програмного забезпечення. Представили гібридний метод вибору важливих функцій програмного забезпечення для покращення автоматизованого прогнозування несправностей. Запропонований підхід поєднував методи фільтрації та обгортки. Метод фільтрації оцінював релевантність функцій без оцінки продуктивності, тоді як метод обгортки включав алгоритми індукції для оцінки продуктивності вибраних підмножин функцій. Гібридний фреймворк використовував два підходи обгортки ANNIGMA та SVM з фільтрами MR. Це об'єднання спростило FS та навчання, а також зменшило зусилля на вимірювання.

Представили новий алгоритм FS, який працює у два кроки в рамках гібридної структури обгортки та фільтра. На першому кроці ознаки групувалися на основі надмірності за допомогою швидкого алгоритму групування фільтрів на основі кореляції. На другому кроці з кожної групи вибиралася підмножина ознак за допомогою алгоритму оптимізації рою частинок (PSO), щоб гарантувати, що принаймні одна ознака була обрана з кожної групи. Експерименти, проведені з різними класифікаторами на наборах даних NASA та PROMISE, показали, що запропонований метод досяг до 90% кращих результатів порівняно з використанням усіх ознак. Представили кластерний гібридний вибір ознак (CHIFS), де вибираються релевантні та ненадлишкові ознаки. Вони визначили спектральний кластерний коефіцієнт якості ознак (FQ) для оцінки релевантності

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

та надмірності ознак. Метод ітеративно вибирав кінцеву підмножину ознак з оцінених ознак на основі FQ. Експериментальні результати показали, що CHIFS перевершував інші методи за точністю та ефективністю, і він був особливо ефективним, коли базувався на коефіцієнті кореляції Пірсона. Використовували комбінацію методів FS для покращення процесу прогнозування програмних помилок. У дослідженні було використано метод FS простого більшостю голосів, який включав результати трьох різних методів FS: рекурсивного виключення ознак з перехресною перевіркою (RFECV), CFS та Select-k-Best FS. Остаточна підмножина ознак визначалася більшістю голосів між трьома методами. Запропонований підхід значно вніс свій вклад у прогнозування програмних помилок і таким чином покращив продуктивність моделей прогнозування. Запропоновано новий метод, RAHMFWS (Rang Aggregation-Based Hybrid Multi-Filter Wrapper Feature Selection), для подолання проблем, пов'язаних з високою розмірністю програмних ознак у моделях SDP. Метод RAHMFWS був розділений на два етапи. Перший етап включав вибір ознак на основі рангової агрегації (RMFFS), який агрегував окремі рейтинги з різних методів фільтрації за допомогою унікального механізму агрегації рангів. На другому етапі було використано метод розширеного вибору обгорткових ознак (EWFS), який керувався стратегією динамічного переупорядкування на основі агрегованого ранжованого списку. Запропонований метод RAHMFWS було оцінено за допомогою класифікаторів NB та дерева рішень (DT) на наборах даних для бенчмаркінгу на наявність дефектів. Результати показали ефективність запропонованого підходу у вирішенні проблем, пов'язаних з вибором рангу фільтра та локальною стагнацією в рамках гібридного вибору ознак, що призвело до покращення продуктивності моделей SDP.

Запропоновано алгоритм FS, відомий як кластеризація на основі ReliefF (RFC). Алгоритм RFC включав кілька кроків, включаючи початкове ранжування релевантності, кластеризацію на основі симетричної невизначеності та репрезентативну FS. Запропонований алгоритм порівнювали з класичними методами FS на наборах даних NASA SDP, використовуючи такі метрики, як

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

AUC та F-оцінка. Результати показали, що RFC покращує продуктивність SDP, ефективно обробляючи нерелевантні ознаки.

Інтегровано метрики коду та метрики процесу як індикатори. Вони досліджували процес SDP, застосовуючи алгоритми RF, нейронних мереж (NN) та SVM, використовуючи середовище Eclipse для бінарної класифікації. Вибір ознак здійснювався за допомогою загальної лінійної регресії моделі (GLM) для оцінки важливості змінних (VI) для кожної ознаки. З 61 ознаки 20 найважливіших ознак було фіксовано на основі VI, що виділяє найважливіші ознаки для прогнозування помилок та призводить до підвищення точності. Запропоновано новий підхід під назвою MOFES (багатоцільовий вибір ознак), заснований на принципах багатоцільової оптимізації, щоб досягти балансу між зменшенням кількості вибраних ознак та оптимізацією продуктивності моделі прогнозування дефектів. Дослідники дійшли висновку, що MOFES ефективно вибирає менше ознак, досягаючи при цьому кращої продуктивності прогнозування порівняно з базовими методами. Запропоновано підхід до вибору ознак для покращення продуктивності шаруватої рекурентної нейронної мережі (L-RNN) для SDP. У запропонованому методі використовувалися три алгоритми-обгортки FS, включаючи бінарний генетичний алгоритм (BGA), бінарну оптимізацію рою частинок (BPSO) та бінарну оптимізацію колонії мурах (BACO). Підхід був оцінений на 19 реальних програмних проектах та порівняний з іншими методами, такими як NB, ANN, LR, використовуючи AUC як міру продуктивності. Запропонований підхід показав кращу продуктивність порівняно з існуючими методами. Запропонований підхід показав кращу продуктивність порівняно з поточними методами. Запропоновано модель під назвою GFsSDAEsTSE, яка спеціально має справу з надлишковістю ознак та дисбалансом класів. Метод інтегрує FS, глибоке навчання за допомогою багатошарового автоенкодера з шумозаглушенням (SDAE) та ансамблеве навчання. Метод JRC використовувався для ітеративного FS, де ознаки додавались до підмножини. Алгоритм зупинявся, коли додавання більшої кількості ознак не покращувало продуктивність суттєво або коли досягався поріг

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

толерантності до мінливості, що призводило до покращення продуктивності SDP.

Представлено новий метод під назвою MOFES (багатоцільовий вибір ознак), який вибирає відповідні ознаки. Було розглянуто дві цілі оптимізації: мінімізація кількості вибраних ознак (з точки зору вартості) та максимізація продуктивності моделей прогнозування (з точки зору корисності). Для вирішення цієї проблеми було використано алгоритми на основі Парето для багатокритеріальної оптимізації. У дослідженні проаналізовано різні алгоритми та порівняно MOFES з існуючими методами FS, продемонструвавши ефективність запропонованого методу з використанням реальних наборів даних. Запропоновано метод FS, заснований на алгоритмі світлячків (FA), для підвищення точності SDP. Природна поведінка миготіння світлячків надихнула на цей метод оптимізації, в якому світлячки спілкуються за допомогою сигналів біолоюмінесценції для ефективного пошуку в просторі функцій вартості. Цей підхід розглядає як точність класифікації, так і зменшення розміру як цілі придатності. Експерименти показали, що потенціал FA для FS у прогнозуванні помилок програмного забезпечення призвів до покращення точності прогнозування та, отже, якості. Досліджували два основні типи методів FS, включаючи вибір ознак на основі фільтрів (FFS) та вибір обгорткових ознак (WFS). Були розглянуті різні методи пошуку, включаючи пошук за найкращим варіантом (BFS) та жадібний покроковий пошук (GSS).

Конкретним методом FS, що використовувався в дослідженні, був WFS, в якому класифікатор використовувався для оцінки продуктивності кожної ознаки шляхом вимірювання її взаємодії з базовим класифікатором. Цей підхід порівнювали з методами FFS, в яких ознаки, отримані з ознак у наборі даних, оцінювалися та ранжувалися без безпосередньої участі класифікатора. Експериментальні результати показали, що метаевристичні методи в WFS перевершили традиційні методи пошуку BFS та GSS. Досліджували комбінацію SMOTE для вирівнювання даних та PSO для FS. Процес FS включав представлення кожної потенційної підмножини ознак як «частинки» у

					БР.ІП – 08.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

багатовимірному просторі пошуку. Положення кожної частинки було пов'язане з певною комбінацією ознак; її рух базувався на її власній історичній ситуації. найкраща позиція та глобальна найкраща позиція, знайдена всім роєм. Це дослідження сприяло вдосконаленню моделей SDP, забезпечуючи ефективну обробку незбалансованих даних та вибір оптимальних ознак. Досліджували методи FS, такі як рекурсивне виключення ознак (RFE), вибір ознак на основі кореляції, Lasso, Ridge, ElasticNet та Boruta. Запропонована модель поєднувала регресію частково найменших квадратів (PLS) та RFE для FS, яка додатково поєднувалася зі SMOTE через незбалансовану природу використовуваних наборів даних. Запропонований підхід досяг значних покращень у прогнозуванні помилок. Представлено розширений метаевристичний алгоритм FS на основі пошуку під назвою EMWS. Цей алгоритм використовує алгоритм оптимізації кита (WOA) та імітований відпал (SA) для ефективного вибору підмножини відповідних ознак. Було розроблено уніфікований алгоритм прогнозування дефектів під назвою WSHCKE, який поєднує гібридну глибоку нейронну мережу (DNN), що складається зі згорткової нейронної мережі (CNN) та машини екстремального навчання ядра (KELM), яка інтегрувала вибрані ознаки в глибокі семантичні ознаки за допомогою CNN та покращила прогнозування за допомогою класифікаційної здатності KELM. Запропонований алгоритм був оцінений за допомогою масштабних експериментів у 20 програмних проектах з різними показниками оцінки. Результати підтвердили ефективність EMWS та WSHCKE порівняно з існуючими методами.

Запропоновано новий фреймворк, який інтегрує вкладене стекування та гетерогенне FS для досягнення точнішого прогнозування дефектів програмного забезпечення. У гетерогенному підході FS різні базові моделі, включаючи AdaBoost та RF, використовувалися для незалежного виконання FS. Кожна базова модель визначала найважливіші ознаки для свого прогнозування. Ознаки, вибрані цими моделями, потім розглядалися для стекування в класифікаторі вложеного стекування. Результати експериментів показали, що запропонована система вложеного стекування перевершила базові моделі. Запропоновано

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

фреймворк, заснований на алгоритмі оптимізації бінарного кита (BWOA), який називається SBEWOA. BWOA використовувався як метод FS, що використовував передавальні функції для перетворення вихідного безперервного WOA у бінарну версію, придатну для FS. Крім того, інші алгоритми оптимізації, зокрема Gray Wolf Optimizer (GWO) та Harris Hawks Optimization (HHO), були інтегровані в фреймворк BWOA для підвищення його ефективності навігації простором ознак. Цей удосконалений метод, відомий як «SBEWOA», показав багатообіцяючі результати для SDP завдяки ефективному вибору інформативних ознак та мінімізації нерелевантних ознак.

Запропоновано структуру, яка досліджувала спільний вплив FS та повторної вибірки даних на незбалансовану двокласову класифікацію. У дослідженні порівнювалися два контрастні підходи, тобто один, у якому FS застосовувалося перед повторною вибіркою даних (FS+DS), та інший, у якому FS застосовувалося після повторної вибірки даних (DS+FS). Методи FS були окремо обрані з фільтрувальних, обгорткових та вбудованих методів. У дослідженні було проведено комплексний емпіричний аналіз п'ятдесяти двох еталонних наборів даних з незбалансованими розподілами з використанням різних методів FS. Дослідження дійшло висновку, що FS є обов'язковим кроком для покращення класифікації незбалансованих двокласових наборів даних і має бути інтегрований або до, або після повторної вибірки даних.

Розподіл кожного методу FS по роках зображено на рисунку 3.1, тоді як на рисунку 3.2 показано метод FS та відповідну кількість первинних досліджень за роками. Підсумовує методи вибору ознак, розглянуті в обраних первинних дослідженнях.

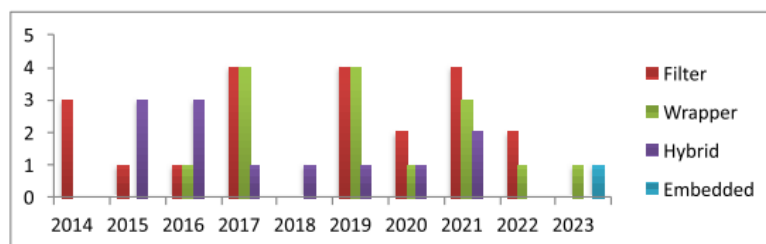


Рисунок 3.1 - Розподіл методів FS за роками.

Вибір методу відбору ознак залежить від кількох факторів, таких як розмір набору даних, доступні обчислювальні ресурси та бажаний рівень точності прогнозування. У цьому контексті було проведено аналіз, щоб виділити сильні та обмежені сторони різних методів відбору ознак на основі їх поширеності у вибраних первинних дослідженнях. Таблиця У розділі 9 розглядаються сильні та обмежені властивості фільтруючих, гібридних та обгорткових методів, а також надається вичерпний огляд їхньої корисності для прогнозування програмних помилок.

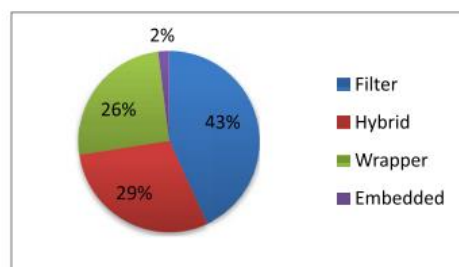


Рисунок 3.2 - Розподіл методів FS у первинних дослідженнях.

З 2014 по 2023 рік було запропоновано численні методи вибору ознак. Дослідники запропонували кілька фреймворків, включаючи FECAR, MICNAS, FeSCH, MOFES, EMWS, CNN та TFSCPDP, використовуючи методи вибору ознак на основі фільтрів, обгортки, гібридні та вбудовані, що призвело до покращення ефективності класифікації. У первинних дослідженнях найчастіше використовувалися методи фільтрів та гібридні методи FS, тоді як методи обгортки та вбудовані методи використовувалися рідше.

3.3 Результати аналізу дефектів програмного забезпечення та їх обговорення

Експерименти з класифікації машинного навчання (ML), описані в цьому документі, були виконані на ноутбучі з Windows, процесором Intel(R) Core(TM)

i5-2410 M, 6 ГБ основної пам'яті та 1 ТБ додаткової. Модель ML була реалізована за допомогою мови програмування Python. Python зазвичай використовується в проєктах прогнозової аналітики та науки про дані, що включають як якісні, так і кількісні дані. Для створення предиктора ML були використані пакети Python: pandas, numpy, seaborn, matplotlib, sklearn та pyswarm.

Результати експериментів на наборі даних CM1 з використанням різних методів машинного навчання представлені у вигляді графіків та таблиць. Ці графіки пропонують порівняння кількох методів. Ми оцінимо всі результати, порівнюючи всі стратегії з оптимізацією та без неї, щоб зробити поняття очевидним та зрозумілим. Таблиці 3.4 та На рисунку 3.3 показано метрики оцінки для всіх досліджених підходів машинного навчання без оптимізації та з оптимізацією відповідно. Усі аналізовані моделі мають найкращі результати, але SVM та оптимізований SVM мають кращу продуктивність в обох випадках, ніж інші. Тут модель SVM точно передбачила вихідний клас 1 як 1, тому точність становить 100%, тоді як один вихідний клас 0 прогнозується як 1, тому повнота результатів менша. Більше того, NB працює погано, оскільки він добре працює на багатовимірному наборі даних. RF та ансамбль мають майже однаковий результат, оскільки Rf також є ансамблевим підходом, який поєднує велику колекцію дерев. Модель RF досягає точності 100% та точності 98,7%. Ця модель також зі 100% точністю прогнозує вихідний клас 1 та пропускає два значення в класі 0.

Таблиця 3.4

Метрики оцінювання класифікаторів без оптимізації.

Набір даних (Dataset)	Метрики оцінювання (Evaluation Measures)	SVM	NB	RF	Ансамбль (Ensemble)
CM1	Точність (Accuracy)	99	93.90	98.70	98.80
	Прецизійність / Влучність (Precision)	100	100	100	100
	Повнота (Recall)	90.90	50	82.20	82.30
	F-міра (F-measure)	95.20	66.70	90.90	90.90

Метрика оцінювання класифікаторів з оптимізацією.

Набір даних (Dataset)	Метрики оцінювання (Evaluation Measures)	SVM	NB	RF	Ансамбль (Ensemble)
CM1	Точність (Accuracy)	99.80	93.80	99.50	97.60
	Прецизійність / Влучність (Precision)	99.70	100	100	99
	Повнота (Recall)	100	92.90	99.50	97.50
	F-міра (F-measure)	96	67.30	91.10	84.80

Результати всіх моделей машинного навчання, а також ансамблевого підходу без оптимізації показано на рисунку 3.3. Точність усіх алгоритмів є найбільшою серед вимірювань, як видно на графіку. Це пояснюється тим, що всі моделі передбачили один клас як 1. Модель NB погано працює у випадку повноти та f-міри, оскільки вона працює з великою кількістю даних або високовимірним набором даних. Модель RF та ансамблева модель мають майже однакові результати. Однак модель SVM перевершує її за всіма оціночними показниками.

Рисунок 3.4 це сумарний результат усіх моделей прогнозування з PSO. Загалом, усі моделі разом з ансамблевим методом досягають хороших оціночних показників; однак кожна модель була перевершена без оптимізації. Оптимізаційні рішення найкращі, коли розмір набору даних збільшено. Однак у цьому випадку значення повноти та f-міри покращуються. Однак тут повнота та F-міра покращилися порівняно з іншими.

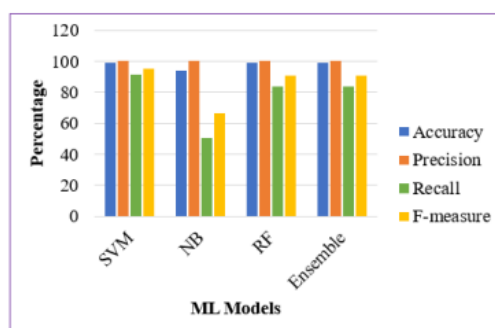


Рисунок 3.3 - Метрики оцінювання всіх моделей без оптимізації.

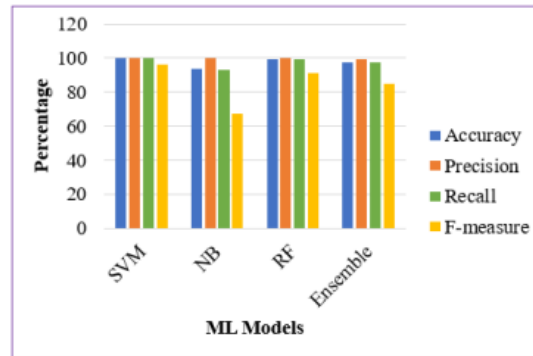


Рисунок 3.4 - Метрики оцінювання всіх моделей з оптимізацією.

підвищення точності класифікаторів, щоб результати можна було передбачити якомога точніше. Аналіз очікуваної моделі порівнюється та розрізняється з визнаними контрольними показниками, такими як середня абсолютна відсоткова похибка (MAPE), середньоквадратична похибка (RMSE) та середня абсолютна похибка (MAE), щоб переконатися та підтвердити, що моделі є адаптивними. Наведені нижче таблиці 3.6 та На рисунку 3.5 показано значення похибки продуктивності моделей. Щодо оцінки похибки продуктивності, як показують значення в таблиці, усі проаналізовані моделі мають нижчий коефіцієнт помилок.

Таблиця 3.6

Метрики помилок продуктивності класифікаторів без оптимізації.

Набір даних (Dataset)	Метрики похибки ефективності (Performance Error Metrics)	SVM	NB	RF	Ансамбль (Ensemble)
CM1	MAE	0.006	0.061	0.012	0.012
	RMSE	0.078	0.25	0.11	0.11
	MAPE	0.65	6.45	1.29	1.29

Таблиця 3.7 та на рисунку 3.6 показано міри помилок усіх моделей без оптимізації та з оптимізацією. Міри помилок, такі як MAE, RMSE, MAPE тощо, показують, скільки помилок було під час навчання та тестування моделі машинного навчання.

Метрики помилок продуктивності класифікаторів з оптимізацією.

Набір даних (Dataset)	Метрики похибки ефективності (Performance Error Metrics)	SVM	NB	RF	Ансамбль (Ensemble)
CM1	MAE	0.157	0.072	0.008	0.079
	RMSE	0.398	0.269	0.089	0.252
	MAPE	9.26	6.02	0.602	7.36

Чим менші значення помилок, тим менша помилка в моделі, а чим більші значення помилок, тим більша помилка в моделі машинного навчання. Усі моделі мають нижчі коефіцієнти помилок, але SVM та оптимізований RF мають нижчі значення помилок як в сценаріях без будь-якої оптимізації, так і з оптимізацією за допомогою інших методів.

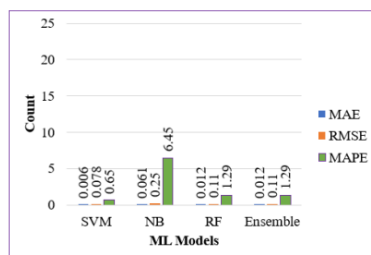


Рисунок 3.5 - Метрики оцінки помилок усіх моделей без оптимізації

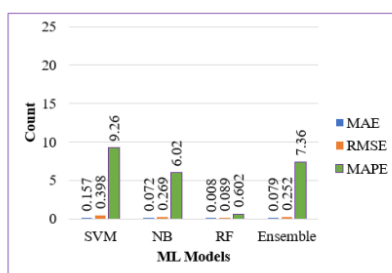


Рисунок 3.6 - Метрики оцінки помилок усіх моделей з оптимізацією

Статистичний аналіз – це процес використання кількісних інструментів для пошуку закономірностей, тенденцій та взаємозв'язків у даних. Використання статистичного аналізу може допомогти в прогнозуванні майбутніх аспектів

дослідження. Можна збирати дані, використовуючи найбільш відповідні підходи, проводити адекватний аналіз та успішно повідомляти результати, якщо є розуміння статистики. Статистичний аналіз, як показано в таблицях 3.8 та 3.9, також використовується в дослідженні для кращого розуміння результатів та прогнозування майбутніх потреб. У цій роботі виконано аналіз хі-квадрат та дисперсійний аналіз. Результат показує, що наші проаналізовані моделі добре функціонували у всіх випадках. Значення p – це метрика для оцінки значущості тесту в статистиці.

Таблиця 3.8

Статистичний аналіз класифікаторів без оптимізації.

Критерій / Тест	Статистичний показник	SVM	NB	RF	Ансамбль (Ensemble)
Chi-Squared (Хі-квадрат)	F-Stat	0.02964	32.736	3.549	3.45
	p-Value	5.29×10^{-6}	0.0856	0.0041	0.0329
Anova (Дисперсійний аналіз)	F-Stat	18.849	2.786	0.2987	0.2503
	p-Value	1.41×10^{-5}	1.51×10^{-8}	0.0085	1.61×10^{-9}

Як результат, низьке значення p вказує на те, що нульову гіпотезу слід відхилити, а замість неї прийняти альтернативну гіпотезу, коротше кажучи, значення p має бути якомога меншим, щоб відхилити нульову гіпотезу. Значення значущості в цьому дослідженні становить 0,10. Значення p менше 0,10 показує, що нульова гіпотеза відхилена і демонструє значущість тесту. Менше значення p , ніж значення значущості, показує значущість тесту.

Таблиця 3.7 та на рисунку 3.6 показано міри помилок усіх моделей без оптимізації та з оптимізацією. Міри помилок, такі як MAE, RMSE, MAPE тощо, показують, скільки помилок було під час навчання та тестування моделі машинного навчання.

Статистичний аналіз класифікаторів з оптимізацією.

Критерій / Тест	Статистичний показник	SVM	NB	RF	Ансамбль (Ensemble)
Chi-Squared (<i>Хі-квадрат</i>)	F-Stat	0.2299	31.437	5.721	12.408
	p-Value	6.78×10^{-5}	2.06E-8	1.51×10^{-5}	0.000245
Anova (<i>Дисперсійний аналіз</i>)	F-Stat	12.5958	5.503	3.432	0.87212
	p-Value	0.00041	0.0121	0.005112	6.88×10^{-17}

Порівняння з попередніми методами з точки зору точності показано порівняння численних підходів машинного навчання (ML), що використовувалися в попередніх дослідженнях. Ми порівняли численні підходи ML, які використовувалися як найсучасніші методи в попередніх дослідженнях, наприклад, В. SVM, NB, Boost- RF, Bag-RF, SVM-FS, NB-FS, RF-FS, SVM-AdaBoost, RF-AdaBoost, SVM-PCA тощо, на вибраному нами наборі даних CM1, який є методом ML з найнижчою точністю, і який ми хотіли покращити, порівнюючи різні стратегії ML за допомогою FS та кластеризації. В результаті порівняння наші оцінені методи працюють добре та демонструють покращення точності порівняно з попередніми сучасними методами.

З графіків видно, що наші моделі з FS та кластеризацією перевершили інші методи з літератури, і ми валідували нашу модель на великих даних, яка на 33% точніша, ніж інші дослідження. Рисунок. На рисунку 3.7 показано графік точності літературних методик, а на рисунку На рисунку 3.8 показано графік аналізованих нами методів з FS та кластеризацією. У цій роботі ми розглянули різні відомі методи машинного навчання (ML) та оптимізували їх з оптимізацією та без неї на вільно доступному наборі даних, щоб заощадити витрати на тестування та одночасно підвищити точність системи.

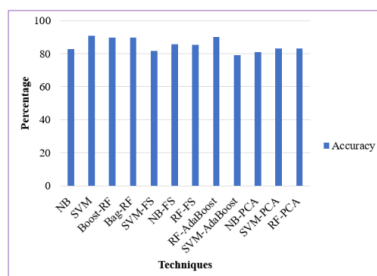


Рисунок 3.7 - Точність літературної моделі

Модель ML була реалізована за допомогою мови програмування Python. Python зазвичай використовується в проектах прогнозовної аналітики та аналізу даних, що включають як якісні, так і кількісні дані. Для створення предиктора ML були використані пакети Python pandas, numpy, seaborn, matplotlib, sklearn та pyswarm. Метою дослідження було покращити продуктивність моделі з точки зору точності та прецизійності набору даних порівняно з попередніми дослідженнями. Таким чином, компанії-розробники програмного забезпечення можуть використовувати це рішення для раннього прогнозування дефектів на етапах розробки та виправлення їх перед постачанням.

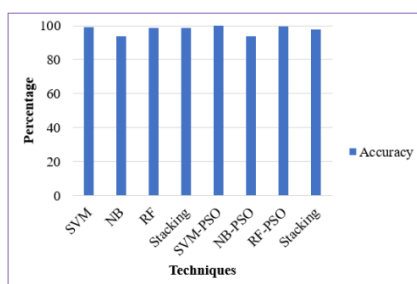


Рисунок 3.8 - Проаналізована точність моделі за допомогою FS та кластеризації.

Найважливішим етапом будь-якого програмного забезпечення, який вимагає інтенсивного тестування, є виявлення дефектів програмного забезпечення. Він також займає найважливіше місце в життєвому циклі розробки програмного забезпечення (ЖЦП). Команда розробників програмного забезпечення може виправити та усунути помилку до надання готового

продукту, якщо їм вдасться знайти проблемний компонент. Очікується, що виявлення та усунення дефектів коштуватимуть мільярди доларів щорічно. Метою ЖЦП є створення відмінного програмного забезпечення, ефективно використовуючи обмежені доступні ресурси. Основними цілями нашого дослідження були:

- Аналіз методів машинного навчання на основі метрик програмного забезпечення;
- Моделі прогнозування дефектів програмного забезпечення з високою точністю;
- Підвищити продуктивність з точки зору точності.

Для оцінки продуктивності моделей використовувалися такі показники, як точність, правильність, повнота, f-міра, міри помилок продуктивності та матриця плутанини. Результати показують, що всі моделі машинного навчання (ML) та оптимізовані моделі ML дають найкращі результати, але моделі SVM та оптимізовані SVM перевершили їх, показавши найвищу досягнуту точність відповідно 98% та 99,80%. Відповідні показники точності для підходів NB, оптимізованого NB, RF, оптимізованого RF та ансамблю становлять 93,90%, 93,80%, 98,70%, 99,50%, 98,80% та 97,60%. Таким чином, ми отримали найвищий ступінь точності порівняно з попередніми дослідженнями, що було нашою метою. Таким чином, ми можемо досягти високоякісного програмного забезпечення, зменшуючи витрати та час тестування. У майбутньому ми можемо розширити набір даних і спробувати проаналізувати різні типи ансамблевих підходів з методами балансування даних, щоб також покращити рівень помилок. Крім того, ми можемо використовувати деякі інші методи оптимізації, які також найкраще працюють на обмежених даних.

Методи машинного навчання (ML) використовуються разом з методами вибору ознак та кластеризації K-середніх для прогнозування дефектів програмного забезпечення. Ми дослідили різні відомі методи ML та оптимізовані методи ML на вільно доступному наборі даних, щоб покращити точність набору даних у порівнянні з попередніми дослідженнями. Результати також

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

аналізуються з використанням ML з методом PSO та ансамблевим підходом. Всі моделі ML навчені та протестовані мовою програмування Python з використанням блокнота Jupyter. Аналіз мав на меті покращити точність ML на наборі даних CM1. Результати показують, що всі моделі ML та оптимізовані моделі ML досягають максимальних результатів; однак моделі SVM та оптимізовані SVM перевершили їх з найвищою досягнутою точністю - 99% та 99,80% відповідно. Точність підходів NB, оптимізованого NB, RF, оптимізованого RF та ансамблю становить 93,90%, 93,80%, 98,70%, 99,50%, 98,80% та 97,60 відповідно.

У цій роботі розглянуто три моделі класифікації машинного навчання з оптимізацією та без неї виконується з використанням ансамблевого підходу для пошуку помилок у програмному забезпеченні. Ці моделі застосовуються до обмеженого набору даних. У майбутньому ми збільшимо розмір набору даних і спробуємо проаналізувати різні типи ансамблевих класифікаторів після застосування методів балансування даних, оскільки завдяки методу балансування ми можемо покращити вимірювання помилок, а також отримати максимальні результати. Ми також застосуємо інший підхід до оптимізації до великого набору даних.

3.4 Висновок по розділу

У третьому розділі виконано практичну реалізацію методів аналізу дефектів програмного забезпечення та оцінено їх ефективність. Проведено підготовку й обробку даних, необхідних для дослідження, а також реалізовано методи вибору ознак для підвищення якості прогнозування дефектів. На основі отриманих результатів здійснено аналіз ефективності застосованих підходів та визначено їх вплив на точність виявлення потенційних помилок. Отримані результати підтвердили доцільність використання сучасних методів аналізу даних для підвищення якості програмного забезпечення та зменшення кількості дефектів на етапах розробки й експлуатації.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання бакалаврської роботи було проведено комплексне дослідження методів аналізу дефектів програмного забезпечення, спрямоване на підвищення якості програмних продуктів та вдосконалення процесів їх розробки і тестування. У роботі розглянуто теоретичні основи аналізу дефектів, досліджено сучасні підходи до прогнозування програмних помилок, проаналізовано методи вибору ознак та оцінено ефективність використання різних алгоритмів для виявлення потенційно проблемних компонентів програмного забезпечення.

На початковому етапі було виконано аналіз предметної області та визначено основні поняття, пов'язані з дефектами програмного забезпечення, причинами їх виникнення та наслідками для якості програмних систем. Особливу увагу приділено сучасним підходам до забезпечення якості програмного забезпечення, а також дослідженню існуючих наукових праць у сфері прогнозування дефектів. Це дозволило сформулювати теоретичну основу для подальшого дослідження та визначити найбільш перспективні напрями аналізу.

У другому розділі було досліджено методологію аналізу дефектів програмного забезпечення, розглянуто матеріали та методи, що використовуються для побудови моделей прогнозування. Проведений аналіз показав, що якість прогнозування значною мірою залежить від коректності підготовки даних, вибору інформативних ознак та застосованих алгоритмів машинного навчання. Також було розглянуто основні категорії алгоритмів прогнозування дефектів, їх переваги, недоліки та сфери практичного використання.

Практична частина роботи була присвячена обробці та підготовці даних для подальшого аналізу. Було виконано маніпулювання даними, досліджено методи відбору ознак та реалізовано підходи, що дозволяють зменшити надлишковість даних і підвищити точність прогнозування. Отримані результати підтвердили, що використання ефективних методів вибору ознак позитивно

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

впливає на якість побудованих моделей і дозволяє більш точно визначати програмні модулі, схильні до виникнення дефектів.

Проведений аналіз результатів показав, що застосування сучасних методів аналізу даних та прогнозування дефектів сприяє підвищенню ефективності процесів тестування, скороченню витрат на виправлення помилок та покращенню загальної якості програмного забезпечення. Отримані висновки підтверджують доцільність використання методів прогнозування дефектів на ранніх етапах життєвого циклу програмного забезпечення, коли вартість усунення помилок є найменшою.

Загалом поставлена мета роботи була досягнута, а всі визначені завдання виконані. Результати дослідження можуть бути використані під час розробки та супроводу програмного забезпечення для підвищення ефективності процесів контролю якості. Перспективами подальших досліджень є вдосконалення моделей прогнозування дефектів шляхом застосування сучасних методів машинного навчання, глибокого навчання та інтелектуального аналізу великих обсягів даних, що дозволить підвищити точність виявлення потенційних дефектів і забезпечити створення більш надійних програмних систем.

					БР.ІП – 08.00.00.000 ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education. <https://www.mheducation.com>
2. Sommerville, I. (2020). *Software Engineering* (10th ed.). Pearson. <https://www.pearson.com>
3. Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). Wiley. <https://www.wiley.com>
4. Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*. Addison-Wesley. <https://www.informit.com>
5. Fenton, N., & Bieman, J. (2014). *Software Metrics: A Rigorous and Practical Approach*. CRC Press. <https://www.routledge.com>
6. IEEE Computer Society. (2024). *Software and Systems Engineering Standards Collection*. <https://www.computer.org>
7. ISO/IEC 25010:2023. *Systems and Software Quality Requirements and Evaluation (SQuaRE) – Product Quality Model*. <https://www.iso.org>
8. ISO/IEC/IEEE 12207:2017. *Systems and Software Engineering – Software Life Cycle Processes*. <https://www.iso.org>
9. ISO/IEC/IEEE 29119-1:2022. *Software Testing – Concepts and Definitions*. <https://www.iso.org>
10. IEEE Std 829-2008. *IEEE Standard for Software and System Test Documentation*. <https://standards.ieee.org>
11. JUnit 5 User Guide. (2025). junit.org. <https://junit.org/junit5/docs/current/user-guide>
12. Selenium Documentation. (2025). selenium.dev. <https://www.selenium.dev/documentation>
13. Cypress Documentation. (2025). docs.cypress.io. <https://docs.cypress.io>
14. SonarQube Documentation. (2025). sonarsource.com. <https://docs.sonarsource.com/sonarqube>
15. Apache JMeter Documentation. (2025). jmeter.apache.org.

					БР.ІІ – 08.00.00.000 ІІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://jmeter.apache.org>

16. Microsoft Testing Documentation. (2025). learn.microsoft.com.
<https://learn.microsoft.com/en-us/dotnet/core/testing>

17. GitHub Actions Documentation. (2025). docs.github.com.
<https://docs.github.com/actions>

18. Azure DevOps Test Plans Documentation. (2025). learn.microsoft.com.
<https://learn.microsoft.com/en-us/azure/devops/test>

19. Jenkins Documentation. (2025). jenkins.io. <https://www.jenkins.io/doc>

20. Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson. <https://www.pearson.com>

21. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley. <https://martinfowler.com/eaCatalog>

22. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson.
<https://www.pearson.com>

23. McConnell, S. (2004). *Code Complete* (2nd ed.). Microsoft Press.
<https://www.microsoftpressstore.com>

24. Osherove, R. (2019). *The Art of Unit Testing* (3rd ed.). Manning Publications. <https://www.manning.com/books/the-art-of-unit-testing-third-edition>

25. Kim, S., Whitehead, E., & Zhang, Y. (2008). Classifying Software Changes: Clean or Buggy? *IEEE Transactions on Software Engineering*, 34(2), 181–196. <https://doi.org/10.1109/TSE.2007.70773>

26. Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A Systematic Literature Review on Fault Prediction Performance. *IEEE Transactions on Software Engineering*, 38(6), 1276–1304. <https://doi.org/10.1109/TSE.2011.103>

27. Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction. *IEEE Transactions on Software Engineering*, 34(4), 485–496. <https://doi.org/10.1109/TSE.2008.35>

28. Menzies, T., Greenwald, J., & Frank, A. (2007). Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*,

					БР.ІІ – 08.00.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		81

33(1), 2–13. <https://doi.org/10.1109/TSE.2007.256941>

29. Khoshgoftaar, T. M., & Allen, E. B. (2003). Ordering Fault-Prone Software Modules. *Software Quality Journal*, 11(1), 19–37. <https://doi.org/10.1023/A:1023015828808>

30. Catal, C. (2011). Software Fault Prediction: A Literature Review and Current Trends. *Expert Systems with Applications*, 38(4), 4626–4636. <https://doi.org/10.1016/j.eswa.2010.10.024>

31. Malhotra, R. (2015). A Systematic Review of Machine Learning Techniques for Software Fault Prediction. *Applied Soft Computing*, 27, 504–518. <https://doi.org/10.1016/j.asoc.2014.11.023>

32. Rodriguez, D., Herraiz, I., Harrison, R., & Dolado, J. (2012). Preliminary Comparison of Techniques for Dealing with Imbalanced Software Defect Data. *Empirical Software Engineering*, 17(4–5), 403–434. <https://doi.org/10.1007/s10664-011-9174-6>

33. IEEE Software. (2024). Software Quality and Defect Management. <https://www.computer.org/software-magazine>

34. ACM Digital Library. (2025). Software Defect Prediction Research Collection. <https://dl.acm.org>

35. Oracle. (2025). Java Testing and Quality Assurance Documentation. <https://docs.oracle.com>

36. Microsoft Learn. (2025). Application Quality and Testing. <https://learn.microsoft.com/en-us/training/browse/?products=dotnet>

37. Mozilla Developer Network. (2025). Debugging and Testing Guide. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing

38. Google Testing Blog. (2025). Software Testing and Defect Analysis Practices. <https://testing.googleblog.com>

39. Red Hat. (2024). Software Quality Engineering and Defect Management. <https://www.redhat.com/en/topics/devops>.

40. Gartner. (2024). Trends in Software Quality Assurance and Defect Analytics. <https://www.gartner.com/en/information-technology/insights/software-engineering>

					БР.ІІІ – 08.00.00.000 ІІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: "Аналіз дефектів програмного забезпечення"

Обсяг пояснювальної записки: 82 аркуші

Дата закінчення дипломної роботи 10 червня 2026р.

Підпис студента _____