Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
https://doi.org/10.36930/40330316
Article received 27.04.2023
Article accepted 25.05.2023
UDC 004.422
Correspondence author: O. S. Khil oleksandr.s.khil@lpnu.ua
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

О. С. Хіль, В. С. Яковина
Національний університет "Львівська політехніка", м. Львів, Україна

АНАЛІЗ ПРОБЛЕМИ ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ОЦІНЮВАННЯ ТА ПРОГНОЗУВАННЯ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Здійснено оцінювання та виконано аналіз літературних джерел, в яких досліджено методи машинного навчання для прогнозування дефектів програмного забезпечення. Визначено основні характеристики дефектів програмного забезпечення, такі як показники складності, ключові слова, зміни, розмір програмного коду та структурні залежності. Охарактеризовано основні методи та засоби прогнозування дефектів програмного забезпечення на основі метрик методами машинного навчання. Описано загальну схему прогнозування дефектів програмного забезпечення, яка дає змогу проводити експерименти та визначати наявність чи відсутність дефекту в програмному модулі. Продуктивність моделі передбачення дефектів програмного забезпечення істотно залежить від вибору набору даних, що є першим кроком проведення дослідження. Встановлено, що попередні дослідження здебільшого базуються на наборах даних з відкритим кодом, а програмні показники, які використовують для створення моделей, переважно є метриками продукту. Набір даних PROMISE (обіцянки) використовується в дослідженнях найчастіше, хоча дані проектів у наборі є застарілими та датуються 2004, 2005 та 2006 роками. Під час виконання цієї роботи проаналізовано сучасні наукові дослідження у галузі. Виявлено методи класифікації, що використовують під час прогнозування дефектів програмного забезпечення. Встановлено, що логістична регресія (англ. *Logistic Regression*), за якою слідує наївний Баєс (англ. *Naive Bayes*) та випадковий ліс (англ. *Random Forest*), є найбільш застосовуваними методами класифікації в таких моделях. Важливим етапом для розуміння ефективності моделі є її оцінювання. Виявлено показники оцінювання ефективності моделі прогнозування дефектів програмного забезпечення, що найчастіше використовують дослідженнях. З'ясовано, що f-measure, за якою слідує recall та AUC, є найпоширенішим показником, який використовується для оцінювання ефективності моделей передбачення дефектів програмного забезпечення. Виявлено, що за останні роки зріс інтерес до використання моделей дефектів програмного забезпечення та класифікації програмних дефектів на основі метрик коду та характеристик проекту. Обґрунтовано актуальність оцінювання та прогнозування дефектів програмного забезпечення методами машинного навчання. Встановлено деякі аспекти, які потребують додаткового дослідження. Визначено напрями майбутніх досліджень, а саме: методи вибору ознак, методи вибору класифікаторів, методи попереднього оброблення даних, побудова моделей прогнозування дефектів, розроблення методів і засобів прогнозування дефектів програмного забезпечення.

Ключові слова: метрики коду; надійність; класифікація; прогнозування дефектів між проектами (CPDP).

Вступ / Introduction

Програмне забезпечення (ПЗ), що використовується у критичних системах, потребує від розробників багато часу та зусиль на підтримку ПЗ через постійну появу дефектів. Серед виявлених дефектів можуть трапитись такі, що негативно впливають на ПЗ. Щоб скоротити час та зусилля розробника, у літературі запропоновано багато моделей машинного навчання, які використовують звіти про задокументовані дефекти для автоматичного прогнозування критичності дефектних програмних модулів [40].

Методи прогнозування дефектів програмного забезпечення можуть скоротити час виявлення дефектів та змістити цей процес на ранні етапи розроблення ПЗ, що знижує вартість самого процесу розроблення [42].

Під час тестування програмного забезпечення зазвичай виділяють три способи прогнозування дефектів програмного забезпечення: прогнозування дефектів у межах проекту (англ. *Within-Project Defect Prediction*, WPDP), прогнозування дефектів між проектами (англ. *Cross-Project Defect Prediction*, CPDP) та прогнозування дефектів гетерогенного перехресного проекту (англ. *Heterogeneous Cross Project Defect Prediction*, HCPDP) [18].

Прогнозування дефектів – це техніка, запроваджена для оптимізації етапу тестування в циклі розроблення ПЗ способом прогнозування наявності дефектів у його компонентах. Відповідна методологія навчає класифікатор із даними щодо набору ознак, виміряних на кожному компоненті цільового проекту ПЗ, щоб передбачити, чи може компонент бути дефектним чи ні. Однак припустимо, що інформація щодо наявності дефекту в компонентах ПЗ недоступна в навчальному наборі. У

Інформація про авторів:
Хіль Олександр Сергійович, аспірант, кафедра систем штучного інтелекту.
Email: oleksandr.s.khil@lpnu.ua; https://orcid.org/0009-0005-8451-3288
Яковина Віталій Степанович, д-р техн. наук, професор, кафедра систем штучного інтелекту.
Email: vitaliy.s.yakovyna@lpnu.ua; https://orcid.org/0000-0003-0133-8591

Цитування за ДСТУ: Хіль О. С., Яковина В. С. Аналіз проблеми застосування методів машинного навчання для оцінювання та прогнозування дефектів програмного забезпечення. Науковий вісник НЛТУ України. 2023, т. 33, № 3. С. 110–116.
Citation APA: Khil, O. S., & Yakovyna, V. S. (2023). Analysis of the problems of applying machine learning methods for evaluating and predicting software defects. *Scientific Bulletin of UNFU*, 33(3), 110–116. https://doi.org/10.36930/40330316

цьому разі нам потрібно покладатися на альтернативний підхід, який використовує навчальний набір зовнішніх проектів для навчання класифікатора. Цей підхід називають CPDP – прогнозуванням дефектів програмного забезпечення між проектами [38].

CPDP стосується розпізнавання дефектних модулів програмного забезпечення в одному проекті (тобто цільовому) з використанням історичних даних, зібраних з інших проектів (тобто джерела), що може допомогти розробникам знайти дефекти та визначити пріоритетність своїх зусиль тестування [27].

Новостворені проекти не мають достатнього обсягу навчальних даних, тому CPDP можна використовувати для створення предикторів дефектів за допомогою інших проектів. Однак CPDP не враховує попередніх знань про цільові елементи та дисбаланс класів у даних вихідного елемента [30].

Об'єкт дослідження – прогнозування дефектів програмного забезпечення.

Предмет дослідження – методи і засоби машинного навчання, які дають змогу оцінити дефектність програмних модулів, а також побудувати відповідні моделі прогнозування дефектів програмного забезпечення.

Мета роботи – проаналізувати наявні проблеми застосування методів машинного навчання для оцінювання та прогнозування дефектів програмного забезпечення.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*: описати та охарактеризувати особливості методології прогнозування дефектів програмного забезпечення; виявити перспективні напрями майбутніх досліджень прогнозування дефектів програмного забезпечення.

Аналіз останніх досліджень та публікацій. Якість програмного забезпечення відіграє важливу роль у життєвому циклі розроблення програмного забезпечення з різних поглядів, які можна брати до уваги. Наприклад, багато підходів до тестування зосереджені на якості, що тлумачиться як відповідність продукту вимогам; якщо є невідповідності, вони вважаються дефектами. Процес тестування спрямований на виявлення та вимірювання дефектів ПЗ: раннє виявлення та усунення дефектів може покращити якість кінцевого продукту. Існує низка різних підходів, які можна використовувати, враховуючи ті, що базуються на аналізі діяльності розробників [10, 14], підходи до оцінювання [32], інструменти [24, 25]. Моделі прогнозування дефектів програмного забезпечення покращують процес тестування ПЗ, допомагаючи ідентифікувати модулі, які, швидше за все, вийдуть з ладу. На етапі тестування таким модулям можна приділяти більше уваги, щоб зменшити ймовірність помилки. Рання ідентифікація таких модулів, які вимагають особливої обережності у тестуванні, розглядається як схильні до дефектів модулі в моделях прогнозування. Тестування великих програмних систем вимагає багато часу та витрат. Через часові та бюджетні обмеження доцільно виявити потенційно схильні до дефектів модулі на ранніх етапах розроблення, щоб пріоритезувати ці модулі на етапі тестування.

Моделі CPDP привернули значну увагу за останні кілька років. Великі програмні системи містять численні функціональні можливості, які взаємодіють одна з одною складними способами, тому ручне тестування цих функцій вимагає відповідних зусиль. Ідея передбачення дефектів базується на здатності побудувати модель із наявних даних за допомогою контрольованих або неконтрольованих підходів машинного навчання та застосувати таку модель до цільового проекту, щоб допомогти на етапах розроблення та тестування. Навчальні дані моделі можуть бути з того самого проекту або різних проектів. Цілями моделей прогнозування дефектів є ідентифікація несправних модулів та оцінка кількості дефектів, щоб оптимально розподілити доступні ресурси для покращення загальної якості. З розвитком штучного інтелекту, зокрема машинного навчання, стало можливим ідентифікувати потенційно схильні до дефектів модулі автоматично на основі емпіричних даних [17, 34]. Отже, модель прогнозування дефектів ПЗ може ідентифікувати дефектні модулі не тестуючи всю базу коду, що дає змогу зосередити зусилля ручного тестування. Подібні підходи можуть зменшити загальні зусилля на тестування та, водночас, підвищити їх ефективність.

Більшість проаналізованих досліджень зосереджено на моделях прогнозування дефектів ПЗ SDP (англ. *Session Description Protocol*, протокол опису сеансу зв'язку) у межах проекту. Такі моделі розробляються з використанням навчальних даних того ж проекту. Тому вихідні та цільові дані моделі однорідні. Однак не завжди можливо знайти дані з того самого проекту, оскільки або вони не існують, або вони не були зібрані та збережені належно для подальшого використання. Через відсутність емпіричних даних з того самого проекту, дослідники з нетерпінням чекають подібних проектів для отримання даних для навчання моделі.

Навіть якщо за останні три десятиліття існує чимало досліджень моделей прогнозування дефектів ПЗ, це не робить їх корисними для галузі через складність збирання та організації даних про дефекти ПЗ [39]. Моделі CPDP пропонують альтернативний варіант, який не потребує великих зусиль для збирання, зберігання та організації даних. Окрім цього, оскільки більшість проектів розроблення програмного забезпечення зараз розробляються за допомогою гнучких підходів, нелегко використовувати інформацію, отриману з попередніх підходів до розроблення та наявні моделі можуть не працювати належно [22]. За таких обставин CPDP є зручною моделлю для впровадження прогнозування дефектів на практиці, оскільки вона не покладається на попередні дані того самого проекту. Можна використовувати схожі та відповідні дані з відкритим кодом для навчання міжпроектних моделей.

Останні публікації [5, 16, 35] свідчать на те, що дедалі більше уваги приділяють прогнозуванню дефектів ПЗ та створенню моделей і методів CPDP [39]. Використання штучних нейронних мереж може виявитися перспективним вирішенням цієї проблеми.

Результати дослідження та їх обговорення / Research results and their discussion

Вибір набору даних є першим кроком для проведення дослідження щодо передбачення дефектів ПЗ. Продуктивність моделі істотно залежить від вибору набору даних. Тому, щоб подолати це, багато досліджень вибирають дані з різних джерел, щоб оцінити стабільність продуктивності в різних наборах даних.

Передусім варто зосередитись на різних проектах ПЗ, які використовують в різних статтях для прогнозу-

вання дефектів. Програмні проекти використовують ітераційний підхід, що робить їх гнучкими за своєю природою. Окрім цього, емпіричні дані проектів розроблення програмного забезпечення, які використовують в різних статтях, взяті з відкритих наборів даних, які є часто не актуальні. Наприклад, дані проекту розроблення програмного забезпечення в репозиторії PROMISE датуються 2004, 2005 та 2006 роками. Дослідники часто використовують ці старі набори даних для проведення своїх експериментів, що зменшує гнучкість впливу проектів розроблення програмного забезпечення на міжпроектні моделі в різних наукових статтях.

Остання тенденція експериментів прогнозування дефектів ПЗ спирається на загальнодоступні набори даних з відкритим кодом. PROMISE – це дуже популярна база даних розроблення програмного забезпечення, призначена для дослідницьких цілей. Понад це, було знайдено вісім наборів даних із відкритим кодом, які об'єднують різну кількість проектів, а саме набір даних NASA MDP (PROMISE), набір даних AEEEM, набір даних NetGene, набір даних SeaCraft, набір даних MORPH, набір даних ReLink, набір даних SOFTLAB і набір даних JURECZKO, які використовують у дослідженнях, пов'язаних з передбаченням дефектів ПЗ.

Загальний підхід для прогнозування дефектів програмного забезпечення на основі машинного навчання складається із таких етапів (рисунок) [3]:

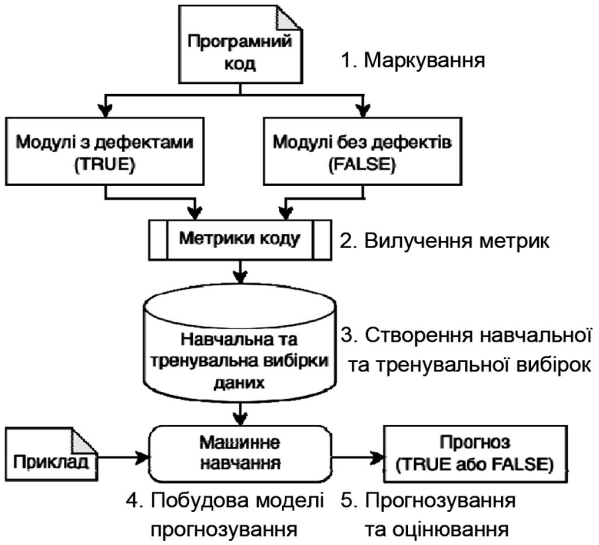

Рисунок. Загальний процес прогнозування дефектів / General defect prediction process

Маркування: дані про дефекти необхідно зібрати для навчання моделі прогнозування. У цьому процесі зазвичай виконується ідентифікація модулів системи на основі тестування на дефектні і без дефектів (маркування TRUE або FALSE).

Вилучення метрик і створення навчальних наборів: цей крок містить вилучення ознак для передбачення міток екземплярів. Загальними характеристиками для передбачення дефектів є показники складності, ключові слова, зміни, розмір програмного коду та структурні залежності. Комбінуючи мітки та ознаки екземплярів, можемо створити навчальний набір, який буде використовуватися під час машинного навчання для побудови моделі передбачення.

Побудова моделі прогнозування: на цьому етапі відбувається вибір моделі (моделей) машинного навчання, які можна використовувати для передбачення за допомогою навчального набору.

Оцінювання: для оцінювання моделі передбачення потрібен набір даних для тестування, окрім навчального набору.

Одним із перших кроків процесу прогнозування дефектів є вилучення ознак (метрик) коду. Метрика коду – міра, що дає змогу отримати чисельне значення деякої якості ПЗ або його специфікацій, до метрик коду належать: кількість рядків коду; цикломатична складність; кількість операторів і операндів, зв'язність коду; ступінь покриття коду тестами тощо. І важливим питанням у процесі прогнозування дефектів, яке є відкритим і актуальним сьогодні, є – які метрики коду найбільше впливають на його якість і надійність.

Ознаки в моделях прогнозування дефектів ПЗ можна розділити на два типи: семантичні ознаки та ознаки, створені вручну. Створені вручну ознаки розроблені та виділені з тексту вихідного коду (наприклад, McCabe [29], CK [8]). Деякі дослідники стверджують, що створеним вручну ознакам не вистачає семантичної інформації вихідного коду [37]. Тому деякі дослідники звертаються до вилучення семантичних ознак, які базуються на абстрактному синтаксичному дереві AST (англ. *Abstract Syntax Tree*). Семантична ознака виділяється на основі перетворення вихідного коду у вектор маркерів за допомогою AST [37]. Застосування методів виділення семантичних ознак у моделі CPDP обмежене (приблизно 6 %) порівняно з техніками, створеними вручну (приблизно 94 %).

Вилучення даних із вихідного коду у вигляді програмних метрик є першим етапом прогнозування дефектів. Дослідники зазвичай використовують такі три методи, щоб вибрати найбільш ефективну метрику програмного забезпечення серед усіх доступних метрик програмного забезпечення в наборі даних:

1. Без вибору: дослідники використовували всі доступні метрики в наборі даних для аналізу [11].
2. Ручний вибір: дослідники вибирають найбільш ефективні метрики вручну на основі своєї інтуїції та досвіду [2].
3. Автоматичний вибір: дослідники застосували автоматичний підхід до вибору метрик, щоб вибрати найбільш прийнятні серед усіх доступних у наборі даних [41].

Виявлено, що більшість досліджень використовували для аналізу комбінацію метрик програмного продукту та процесу. У табл. 1 зображено метрики продукту, що найчастіше використовують у дослідженнях.

Табл. 1. Метрики програмного продукту / Software product metrics

№	Метрика	Опис
1	2	3
1	Chidamber & Kemerer metrics [8]	Набір метрик для об'єктно-орієнтованого проектування, запропонований Чідамбером і Кемерером (1994)
2	Henderson Sellers Metrics [15]	Набір метрик для складності програмних компонентів, запропонований Брайаном Хендерсоном-Селером (1995)
3	Martin Metrics [28]	Набір метрик для якості об'єктно-орієнтованого проектування, запропонований Робертом Мартіним (1994)
4	QMOOD Metrics [5]	Набір метрик для якості об'єктно-орієнтованого проектування, запропонований Карлом Девісом та Джагдишем Бансія (2002)
5	McCabe Metrics [29]	Набір метрик для складності програмних компонентів, запропонований Томасом Маккабе (1976)

1	2	3
6	Size Metrics [7]	Набір метрик для розміру програмних компонентів
7	Halstead Metrics [4]	Набір метрик для складності програмних компонент який базується на кількості унікальних операторів та операндів, запропонований Maurice Howard Halstead (1977)
8	Object-Oriented (OO) Metrics [12]	Набір метрик для об'єктно-орієнтованого проектування

Метрики процесу розроблення ПЗ також розглядають як метрики змін [9]. Тому також класифіковано [19] найбільш використовувані метрики процесу (табл. 2).

Табл. 2. Метрики процесу / Process metrics

№	Метрика	Опис
1	Number of Revisions [31]	Метрика програмного забезпечення, яка вказує на кількість змін програмного артефакту, наприклад файлу, класу чи методу, під час його розроблення чи підтримки. Використовуються для оцінювання стабільності, надійності та супроводжуваності програмних систем, а також для виявлення дефектних або часто змінюваних компонентів.
2	Commits [33]	Метрика програмного забезпечення, яка вказує кількість разів, коли артефакт програмного забезпечення було зареєстровано в системі контролю версій. Використовується для позначення частоти та інтенсивності діяльності з розроблення програмного забезпечення, а також внеску та співпраці розробників.
3	Modified Code [23]	Метрика програмного забезпечення, яка вказує кількість змінених стрічок коду програмного артефакту. Зміною вважається додавання, видалення або редагування коду програмної компоненти.

Широко використовують різноманітні методи машинного для вибору ознак, перед процесом прогнозування дефектів. Виділення або відбір ознак – це процес вибору найважливіших ознак вибірки даних для зменшення кількості вхідних змінних способом усунення зайвих або невідповідних ознак і звуження набору ознак до тих, які найбільше стосуються моделі машинного навчання [21]. Використання методів вибору ознак є хорошим вирішенням проблеми високої розмірності вибірки даних і ці методи активно використовують в контексті прогнозування дефектів програмного забезпечення. Існує чимало досліджень на тему вибору ознак у прогнозуванні дефектності, головною метою яких є вибрати найточнішу комбінацію метрик коду для прогнозування дефектів.

Прогнозування дефектів ПЗ – це проблема бінарної класифікації [13], мета якої класифікувати компоненти ПЗ як дефектний чи недефектний. У моделі прогнозування дефектів ПЗ класифікація є останнім кроком. Упродовж багатьох років дослідники намагалися покращити продуктивність моделі передбачення дефектів ПЗ, також над покращенням продуктивності класифікаторів. Набір класифікаторів, які застосовують для передбачення дефектності програмної компоненти, зображено в табл. 3.

Такі класифікатори, як логістична регресія (LR), наївний Байєс (англ. *Naive Bayes,* NB) та випадковий ліс (RF) найчастіше трапляються в різних роботах з передбачення дефектів ПЗ. Класифікатор є важливим компонентом моделі передбачення дефектів, але не всі дослідження намагалися вдосконалити його для досягнення кращих загальних прогнозів.

Табл. 3. Класифікатори / Classifiers

№	Класифікатор	Позначення
1	Ada Boost	AB
2	Alternating Decision Tree	ADTree
3	Artificial Neural Network	ANN
4	Bayesian Networks	BN
5	Boost	BT
6	Boosting-Support Vector Machine	B-SVM
7	Classification and Regression Tree	CART
8	Coordinate Ascent	CA
9	Correlation Metric Selection based Correlation Slignment	CMSCA
10	Cost-Sensitive Kernelized Semi-Supervised Dictionary Learning	CKSDL
11	Decision Table	DTa
12	Decision Tree J48	DT
13	Deep Adaptation Networks	DAN
14	Diffused Bayes Classifier	DBC
15	Ensemble Learning	EL
16	Ensemble learning classifier Gradient Boosting	ELGB
17	Extra Tree	ET
18	Genetic Algorithm with Ensemble Learning	GAEL
19	Gradient Boost	GB
20	Heterogeneous Ensemble	HE
21	Kernelized Semi-Supervised Dictionary Learning	KSDL
22	K-Nearest Neighbors	KNN
23	Kstar (K*)	KS
24	LambdaMART	LM
25	ListNet	LN
26	Logistic Model Tree	LMT
27	Logistic Regression	LR
28	Logistic Regression using Genetic Algorithm	LRGA
29	Multi-Objective Decision Tree	MODT
30	Multi-Objective Logistic Regression	MOLR
31	Multi-Objective Naive Bayes	MONB
32	Multi-Objective Naive Bayes with Nearest Neighbors	MONNB
33	Naive Bayes	NB
34	oneR	OR
35	Radial Basis Function Network	RBF
36	Random Forest	RF
37	RankNet	RN
38	Ridge	RE
39	Semi-Supervised Structured Dictionary Learning	SSDL
40	Semi-Supervised Dictionary Learning	SDL
41	Spectral Clustering	SC
42	Spotted Hyena Optimizer	SHO
43	Support Vector Machine	SVM
44	Support Vector Machine with RBF kernel	RBF-SVM
45	Token Based Classifier	TBC
46	Transfer Naive Bayes	TNB
47	Value Aware Boosting with Support Vector Machine	VAB-SVM

Оцінювання моделі прогнозування дефектів ПЗ є важливим етапом для розуміння її ефективності. Було визначено показники, які використовувались у дослідженнях для оцінювання моделі прогнозування: precision, recall, accuracy, probability of false positive, true negative rate, balance, ROC-AUC, F-Measure, G-Measure, H-Measure, G-Mean, Win/Draw/Loss, false negative rate, Matthews correlation coefficient.

Recall, ROC-AUC та F-measure є найчастіше використовуваними показниками для оцінювання ефективності моделі передбачення дефектів ПЗ. Незважаючи на те, що дослідники обирають різні показники оцінюван-

ня для вимірювання ефективності моделі, майже жодна зі статей не пояснює мети вибору конкретного показника оцінки.

Існує велика кількість робіт, де розглянуто особливості прогнозування дефектів ПЗ. Проте все ще залишаються проблеми, які необхідно взяти до уваги. Виявлено, що більшість досліджень CPDP використовують контрольоване навчання. Напівконтрольоване та неконтрольоване навчання не були ґрунтовно вивчені. Немає чітких доказів того, який підхід є найкращим і за яких умов.

Рання ідентифікація відповідних проектів для навчання моделі може кардинально змінити загальну продуктивність. Існує дуже обмежена кількість статей, які стосуються цього, а також вони не забезпечують належних вказівок для відбору проектів.

Незрозуміло, які показники працюють краще в кроспроектних налаштуваннях. Більшість моделей CPDP побудовано з припущенням, що вибрані показники підходять для CPDP. У деяких дослідженнях порівнювали показники процесу та продукту. Однак потрібно виконати детальний аналіз на рівні метрики.

Концепція інформаційного вмісту може бути корисною для вивчення прихованого зв'язку між проектами та ознаками. Є кілька робіт, заснованих на концепції інформаційного вмісту, але вони не досліджували прихований зв'язок між проектами та між ознаками.

Зрозуміло, що майже всі дослідження намагалися покращити продуктивність CPDP шляхом удосконалення методів відбору проектів, методів автоматичного вибору ознак і відповідних класифікаторів. Однак вони не змогли вийти за межі 70 % продуктивності. Цей рівень потрібно покращити, щоб запровадити моделі CPDP на практиці.

Застосування глибокого навчання дає цікаві результати в багатьох різних областях. Однак було знайдено тільки кілька останніх досліджень, заснованих на глибокому навчанні.

Модель програмного обчислення на основі правил (наприклад, Fuzzy Logic) може бути корисною в CPDP. Прийняття міжпроектних рішень є дуже складним процесом, і прийняття рішень на основі правил може дати цікаві результати.

Жодне з досліджень не інтегрувало причину несправностей (аналіз першопричини) у своїх моделях. Ідентифікація причини несправності може дати користь у практичному використанні таких моделей.

Обговорення результатів дослідження. Область прогнозування дефектів набула величезної популярності у світі досліджень і розробок. Це виявилося важливою сферою дослідження, оскільки вдалося зменшити кількість зусиль і ресурсів, необхідних для тестування програмного забезпечення. На сьогодні виконують різноманітну дослідницьку роботу для створення та використання моделей машинного навчання для прогнозування дефектів програмних систем.

У роботі [36] порівняли шість алгоритмів фільтрування вибору ознак та дійшли висновку, що результати підвищують точність прогнозування на малих наборах даних. Згодом провели порівняльний аналіз техніки вибору ознак фільтра із застосуванням різних алгоритмів класифікації. Результати цього дослідження доводять, що точність передбачення підвищується за наявності найменшої кількості ознак. Очікувані результати ілюструють скорочення часу обчислення та витрат на конструкцію як на етапах навчання, так і на етапі класифікації моделі продуктивності.

Дослідження [6] застосовує методи вибору ранжованих ознак, Data Sampling та ітераційний фільтр розділення для зменшення високої розмірності, дисбалансу класів і шумових ознак. Автори використовували методи класифікації для прогнозування дефектних модулів у програмному забезпеченні та порівнювали його продуктивність. З результатів автори дійшли висновку, що алгоритм класифікації Random Forest працює краще порівняно з іншими алгоритмами класифікації.

У роботі [43] поєднано кореляційний аналіз та ReliefF Feature Selection для вибору точних ознак. ANOVA (дисперсійний аналіз) показує, що новий метод під назвою ReliefF-LC (лінійний кореляційний аналіз) може підвищити ефективність прогнозування дефектів.

У дослідженні [1] було використано алгоритми машинного навчання для прогнозування дефектів програмного забезпечення. Показники точності класифікації, f-measure і ROC-AUC використовують для оцінювання показників ефективності різних алгоритмів машинного навчання. Перевибірка SMOTE використовується для пом'якшення проблеми нерівності набору даних. Нарешті результати довели, що алгоритми Random Forest, AdaBoost із випадковим лісом і пакетування з деревом рішень працюють добре.

У роботі [16] проаналізовано продуктивність CPDP. Це дослідження складається з 24 наукових статей з 2008 по 2015 роки. Було виявлено, що CPDP на основі класифікатора дерева рішень працює краще порівняно з іншими класифікаторами. Встановлено, що жоден підхід CPDP не досягає бажаної продуктивності принаймні 75 % точності. Також досліджено вплив підходів фільтрації даних на продуктивність. Порівняно набори даних JURECZKO та FILTERJURECZKO, щоб зрозуміти, чи це вплинуло на продуктивність моделі. Набір даних JURECZKO [26] містить 48 продуктів з відкритим кодом, 27 пропрієтарних продуктів і 17 академічних продуктів із загальною кількістю 92 продукти. Виявлено, що немає істотної різниці між двома наборами даних щодо AUC, f-measure, g-measure, MCC та AUCEC. Однак, якщо відфільтровані набори даних малі порівняно з вихідними, це може збільшити продуктивність до 5 %. Унаслідок зроблено висновок, що моделі CPDP не досягли необхідної продуктивності для їх практичного застосування.

У дослідженні [5] проаналізували 34 статті, пов'язані з CPDP з 2008 по 2019 роки. Досліджено типи наборів даних, методи моделювання, типи програмних показників, параметри оцінювання та проведено статистичні тести. Вони виявили, що дослідження використовували пропрієтарні набори даних (6 %), набори даних з відкритим кодом (26 %) і змішані набори даних (68 %). Показано, що логістична регресія та Naive Bayes є найбільш використовуваними методами класифікації. Метрики продукту використовувалися в 62 % досліджень, а метрики процесу – у 38 %. Окрім цього, f-measure є кращим параметром оцінки. Було з'ясовано, що 21 % досліджень не містять жодного статистичного тесту, тоді як найбільш використовуваним є тест Wilcoxon Signed-Rank. Унаслідок цього автори дійшли висновку, що існує великий простір для покращення продуктивності моделей CPDP.

На відміну від наявних робіт, які проводять аналіз проблеми передбачення дефектів методами машинного

навчання враховуючи результати досліджень до 2015 р., у нашому дослідженні розглянуто актуальні роботи в період з 2018 до 2023 роки.

Отже, за результатами виконаної роботи можна сформулювати наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – набуло подальшого розвитку питання вирішення проблеми застосування методів машинного навчання для оцінювання та прогнозування дефектів програмного забезпечення.

Практична значущість результатів дослідження – отримані результати можуть бути використані для покращення наявних моделей і методів машинного навчання у системах діагностики програмних засобів; результати дослідження можна використати в навчальному процесі під час викладання дисципліни "Машинне навчання".

Висновок / Conclusions

Якість програмної системи залежить від того, що ПЗ повинно бути доставлено вчасно, без помилок і система повинна відповідати всім вимогам кінцевого користувача. Щоб створювати ПЗ без дефектів, програмні помилки потрібно ідентифікувати та передбачити на ранній фазі життєвого циклу системи. Упродовж останніх трьох десятиліть дослідники активно працювали у сфері прогнозування дефектів програмного забезпечення. Ця робота містить аналіз сучасних наукових досліджень у сфері прогнозування дефектів програмного забезпечення, внаслідок якого:

1. Визначено та описано основні характеристики дефектів, які можуть виникати у програмному забезпеченні.
2. Охарактеризовано основні методи та засоби прогнозування дефектів програмного забезпечення на основі метрик методами машинного навчання.
3. Представлено загальну модель прогнозування дефектів програмного забезпечення, яка дає змогу експериментувати та встановлювати належність програмного модуля до дефектного чи бездефектного класу.
4. Встановлено, що більшість попередніх досліджень використовували набори даних з відкритим кодом та метрики продукту для побудови моделей.
5. Визначено методи класифікації, які застосовуються для прогнозування дефектів програмного забезпечення.
6. Встановлено найпоширеніші показники оцінювання ефективності моделі прогнозування дефектів програмного забезпечення, які застосовують у дослідженнях.

Це дослідження може допомогти дослідникам і практикам визначити належні підходи для застосування в їхньому контексті та покращити сучасні підходи. Окрім цього, було визначено деякі відкриті питання та області, які потребують додаткових досліджень для покращення продуктивності моделей CPDP.

References

1. Abdullah, A., & Khan, M. Z. (2019). Software defect prediction using supervised machine learning and ensemble techniques: a comparative study. *Journal of Software Engineering and Applications*, *12*(5), 85–100. https://doi.org/10.4236/jsea.2019.125007
2. Agrawal, A., & Malhotra, R. (2022). Cross project defect prediction for open source software. *International Journal of Information Technology*, *14*(1), 587–601. https://doi.org/10.1007/s41870-019-00299-6
3. Akif, H. M., Reddy, R. V., Nagella, K., & Vidya, S. (2021). Software Defect Estimation Using Machine Learning Algorithms. *International Journal of Recent Technology and Engineering*, *10*(1), 204–8. https://doi.org/10.35940/ijrte.a5898.0510121
4. Bailey, C. T., & Dingee, W. L. (1981). A software study using Halstead metrics. *ACM SIGMETRICS Performance Evaluation Review*, *10*(1), 189–197. https://doi.org/10.1145/1010627.807928
5. Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, *28*(1), 4–17. https://doi.org/10.1109/32.979986
6. Bashir, K., Li, T., Yohannese, C. W., & Mahama, Y. (2017). Enhancing software defect prediction using supervised-learning based framework. *International Conference on Intelligent Systems and Knowledge Engineering* (*ISKE*), *12*(*1*), 1–6. https://doi.org/10.1109/ISKE.2017.8258790
7. Basili, V. R., & Reiter, R. W. (1979). Evaluating automatable measures of software development. *Workshop on Quantitative Software Models for Reliability, Complexity and Cost*, 107–116.
8. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, *20*(6), 476–493. https://doi.org/10.1109/32.295895
9. Choudhary, G. R., Kumar, S., Kumar, K., Mishra, A., & Catal, C. (2018). Empirical analysis of change metrics for software fault prediction. *Computers and Electrical Engineering*, *67*(1), 15–24. https://doi.org/10.1016/j.compeleceng.2018.02.043
10. Coman, I. D., & Sillitti, A. (2007). An empirical exporatory study on inferring developpers activities from low-level data. *International Conference on Software Engineering & Knowledge Engineering* (*SEKE*), *19*(1), 15–18.
11. Cui, C., Liu, B., & Wang, S. (2019). Isolation forest filter to simplify training data for cross-project defect prediction. *Prognostics and System Health Management Conference* (*PHM-Qingdao*), 1–6. https://doi.org/10.1109/PHM-Qingdao46334.2019.8942919
12. DAmbros, M., Lanza, M., & Robbes, R. (2012). Evaluating defect prediction approaches: A benchmark and an extensive comparison. *Empirical Software Engineering*, *17*(4), 531–577. https://doi.org/10.1007/s10664-011-9173-9
13. Felix, E. A., & Lee, S. P. (2020). Predicting the number of defects in a new software version. *PLoS ONE*, *15*(3). https://doi.org/10.1371/journal.pone.0229131
14. Fronza, I., Janes, A., Sillitti, A., Succi, G., & Trebeschi, S. (2013). Cooperation wordle using pre-attentive processing techniques. *International Workshop on Cooperative and Human Aspects of Software Engineering* (*CHASE*), *6*(1), 57–64. https://doi.org/10.1109/CHASE.2013.6614732
15. Henderson-Sellers, B. (1995). Object-Oriented Metrics: Measures of Complexity.
16. Herbold, S., Trautsch, A., & Grabowski, J. (2018). A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, *44*(9), 811–833. https://doi.org/10.1109/TSE.2017.2724538
17. Jayanthi, R., & Florence, L. (2019). Software defect prediction techniques using metrics based on neural network classifier. *Cluster Computing*, *22*(1), 77–88. https://doi.org/10.1007/s10586-018-1730-1
18. Jindal, R., Ahmad, A., & Aditya, A. (2021). Ensemble Based-Cross Project Defect Prediction. *Smart Innovation, Systems and Technologies*, *243*(1), 611–620. https://doi.org/10.1007/978-981-16-3675-2_47
19. Jureczko, M., & Madeyski, L. (2011). A review of process metrics in defect prediction studies. *Metody Informatyki Stosowanej*, 5, 133–145.
20. Khatri, Y., & Singh, S. K. (2022). Cross project defect prediction: A comprehensive survey with its SWOT analysis. *Innovations in Systems and Software Engineering*, *18*(2), 263–281. https://doi.org/10.1007/s11334-020-00380-5
21. Kim, S., Zhang, H., Wu, R., & Gong, L. (2011). Dealing with Noise in Defect Prediction. *International Conference on Software Engineering*, 33(1), 481–490. https://doi.org/10.1145/1985793.1985859
22. Kitchenham, B. A., Mendes, E., & Travassos, G. H. (2007). Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, *33*(5), 316–329. https://doi.org/10.1109/TSE.2007.1001

23. Layman, L., Kudrjavets, G., & Nagappan, N. (2008). Iterative identification of fault-prone binaries using in-process metrics. *ACM-IEEE International symposium on Empirical software engineering and measurement*, *2*(1), 206–212. https://doi.org/10.1145/1414004.1414038
24. Lenarduzzi, V., Sillitti, A., & Taibi, D. (2017). Analyzing forty years of software maintenance models. *IEEE/ACM 39th International Conference on Software Engineering Companion* (*ICSE-C*), *39*(1), 146–148. https://doi.org/10.1109/ICSE-S.2017.122
25. Lenarduzzi, V., Sillitti, A., & Taibi, D. (2018). A survey on code analysis tools for software maintenance prediction. *International Conference in Software Engineering for Defence Applications* (*SEDA*), *6*(1), 165–175. https://doi.org/10.1007/978-3-030-14687-0_15
26. Li, Y., Huang, Z., Wang, Y., & Fang, B. (2017). Evaluating data filter on crossproject defect prediction: Comparison and improvements. *Access*, *10*(1), 25646–25656. https://doi.org/10.1109/ACCESS.2017.2771460
27. Li, Z., Zhang, H., Jing, X., Xie, J., Guo, M., & Ren, J. (2022). DSSDPP: Data Selection and Sampling Based Domain Programming Predictor for Cross-Project Defect Prediction. *IEEE Transactions on Software Engineering*, *49*(4), 1941–1963. https://doi.org/10.1109/TSE.2022.3204589
28. Martin, R. (1994). OO design quality metrics: A*n Analysis of Dependencies*, 12(1), 151–170.
29. McCabe, T. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, *2*(4), 308–320. https://doi.org/10.1109/TSE.1976.233837
30. Mi, W., Li, Y., Wen, M., & Chen, Y. (2022). Using active learning selection approach for crossproject software defect prediction. *Connection Science*, *34*(1), 1482–1499. https://doi.org/10.1080/09540091.2022.2077913
31. Moser, R., Pedrycz, W., & Succi, G. (2008). A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. *ACM/IEEE International Conference on Software Engineering*, *30*(1), 181–190. https://doi.org/10.1145/1368088.1368114
32. Moser, R., Pedrycz, W., Sillitti, A., & Succi, G. (2008). A model to identify refactoring effort during maintenance by mining source code repositories. *International Conference on Product Focused Software Process Improvement* (*PROFES*), *9*(1), 360–370. https://doi.org/10.1007/978-3-540-69566-0_29
33. Muthukumaran, K., Choudhary, A., & Murthy, N. L. B. (2015). Mining GitHub for novel change metrics to predict buggy files in software systems. *International Conference on Computational Intelligence and Networks*, 15–20. https://doi.org/10.1109/CINE.2015.13
34. Pal, S. (2021). Generative adversarial network-based cross-project fault prediction. arXiv:2105.07207. https://doi.org/10.48550/arXiv.2105.07207
35. Pal, S., & Sillitti, A. (2022). Cross-Project Defect Prediction: A Literature Review. *Access*, 10(1), 118697–118717. https://doi.org/10.1109/ACCESS.2022.3221184
36. Ramaswami, M., & Bhaskaran, R. (2009). A study on feature selection techniques in educational data mining. *Journal of Computing*, *1*(1), 7–11. https://doi.org/10.48550/arXiv.0912.3924
37. Sheng, L., Lu, L., & Lin, J. (2020). An adversarial discriminative convolutional neural network for cross-project defect prediction. *Access*, *8*(1), 55241–55253. https://doi.org/10.1109/ACCESS.2020.2981869
38. Sotto-Mayor, B., & Kalech, M. (2021). Cross-project smell-based defect prediction. *Soft Computing*, *25*(1), 14171–14181. https://doi.org/10.1007/s00500-021-06254-7
39. Turhan, B., Menzies, T., Bener, A. B., & Di Stefano, J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, *14*(5), 540–578. https://doi.org/10.1007/s10664-008-9103-7
40. Umamaheswara Sharma, B., & Ravichandra, Sadam. (2022). Towards Developing and Analysing Metric-Based Software Defect Severity Prediction Model. https://doi.org/10.48550/arXiv.2210.04665
41. Vashisht, R., & Rizvi, S. A. M. (2020). Feature extraction to heterogeneous cross project defect prediction. International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 8(1), 1221–1225. https://doi.org/10.1109/ICRITO48877.2020.9197799
42. Wen, W., Shen, C., Lu, X., Li, Z., Wang, H., Zhang, R., & Zhu, N. (2022). Cross-Project Software Defect Prediction Based on Class Code Similarity. *Access*, 10(1), 105485–105495. https://doi.org/10.1109/ACCESS.2022.3211401
43. Xia, Y., Yan, G., Jiang, X., & Yang, Y. (2014). A new metrics selection method for software defect prediction. *International Conference on Progress in Informatics and Computing,* 433–436. https://doi.org/10.1109/PIC.2014.6972372

O. S. Khil, V. S. Yakovyna

Lviv Polytechnic National University, Lviv, Ukraine

ANALYSIS OF THE PROBLEMS OF APPLYING MACHINE LEARNING METHODS FOR EVALUATING AND PREDICTING SOFTWARE DEFECTS

An evaluation and analysis of literature sources, in which the methods of machine learning for predicting software defects, were investigated. The main characteristics of software defects are identified, such as complexity indicators, keywords, changes, software code size, and structural dependencies. The main methods and means of predicting software defects based on metrics by machine learning methods are characterized. A general scheme for predicting software defects is described, which makes it possible to conduct experiments and determine the presence or absence of a defect in a software module. The performance of a software defect prediction model strongly depends on the choice of data set, which is the first step in conducting research. Previous studies have been found to be mostly based on open-source datasets, and the software metrics used to build the models are mostly product metrics. The PROMISE data set is the most common in research, although the project data in the data set is outdated and dates back to 2004, 2005 and 2006. While performing this work, modern scientific research in the field was analyzed. The classification methods used in the prediction of software defects have been revealed. It has been established that Logistic Regression, followed by Naive Bayes and Random Forest, are the most widely used classification methods in such models. An important stage for understanding the effectiveness of the model is its evaluation. Indicators of the evaluation of the effectiveness of the software defect forecasting model, which are most often used in research, have been revealed. It is found that f-measure, followed by recall and AUC, is the most common metric used to evaluate the performance of software defect prediction models. It has been found that in recent years there has been an increased interest in the use of software defect models and the classification of software defects based on code metrics and project characteristics. The relevance of evaluating and predicting software defects using machine learning methods is substantiated. Some aspects that require additional research have been identified. The directions of future research are determined, namely: feature selection methods, classifier selection methods, data preprocessing methods, construction of defect prediction models, development of software defect prediction methods and tools.

Keywords: code metrics; reliability; classification; Cross-project software defect prediction, CPDP.