

О. А. Горносталь, В. В. Челак

Національний технічний університет «Харківський політехнічний інститут», Харків, Україна

## ДИВЕРСИФІКОВАНІ АНСАМБЛІ КЛАСИФІКАТОРІВ ДЛЯ ВИЯВЛЕННЯ ДЕФЕКТІВ ПРОГРАМНОГО КОДУ ЗА СТАТИЧНИМИ МЕТРИКАМИ ЯКОСТІ

**Анотація.** Актуальність дослідження полягає у необхідності вдосконалення існуючих підходів до класифікації дефектів вихідного коду з метою своєчасного попередження можливих проблем з його виконанням та потенційних вторгнень. **Об'єктом дослідження** є процес автоматизованого виявлення дефектів програмного коду. **Предметом дослідження** є класифікаційні методи ансамблевого машинного навчання. **Метою дослідження** є підвищення якості ансамблевих класифікаторів в задачах класифікації дефектів програмного коду. **Завдання дослідження:** проаналізувати характеристики набору даних, побудувати послідовність ансамблевих моделей від одиночних класифікаторів до диверсифікованого ансамблю з метою класифікації потенційних дефектів програмних модулів, підтвердити статистичну значущість приросту якості на кожному етапі. **Методи дослідження:** методи попередньої обробки даних, методи відбору ознак, базові методи машинного навчання, ансамблеві методи, диверсифікаційні техніки, статистичні методи оцінювання. **Результати дослідження.** Експериментально підтверджено монотонне зростання якості класифікації шляхом ускладнення ансамблевої архітектури – від найкращого базового класифікатора Gradient Boosting (CV F1 = 0,798) до запропонованого методу розширеного диверсифікованого стекингу (CV F1 = 0,824), який перевершив інші моделі за контрольованими метриками при найнижчій дисперсії та найкращому калібруванні ймовірностей. **Висновки.** За результатами дослідження запропоновано метод класифікації дефектів програмного коду, що поєднує диверсифікований стекинг на підпросторах ознак із додатковими декорельованими гілками на основі Extra Trees та адаптивним мета-класифікатором логістичної регресії. Загальний приріст якості наблизив досягнутої точності до теоретичної верхньої межі, зумовленої наявністю шуму в розмітці набору даних.

**Ключові слова:** дефекти програмного коду; статичні метрики якості; машинне навчання; класифікація; ансамблеві методи; диверсифікація ознак; виявлення вразливостей; мета-класифікатор.

### 1. Вступ

#### 1.1 Актуальність дослідження

Розробка програмного забезпечення сучасних інформаційних систем ведеться в рамках безперервного циклу інтеграції та доставки (CI/CD). За такого підходу кожна зміна вихідного коду дуже швидко проходить шлях від фрагменту програми до виконуваного файлу, а тому повинна пройти автоматизовану перевірку якості перед потраплянням у живе середовище, щоб гарантувати її надійність та безпечність. У цих умовах дефекти програмного коду залишаються головним джерелом вразливостей безпеки. Згідно зі звітом Національного інституту стандартів і технологій США кількість поданих записів до Національної бази даних вразливостей зросла на 263% упродовж 2020-2025 років, сформувавши стійке відставання в їх опрацюванні [1]. Крім того, за даними IBM середні світові втрати організації від одного інциденту порушення захисту даних у 2025 році склали 4,44 млн доларів за середньої тривалості життєвого циклу інциденту 241 день [2].

Провідні позиції переліку в рейтингу CWE Top 25 стабільно посідають саме дефекти рівня вихідного коду [3], тобто категорії, які можна потенційно виявити на етапі статичного аналізу коду перед його збиранням для подальшого виконання на цільових пристроях. Виявлення потенційних дефектів на цьому кроці дозволить знизити вартість усунення наслідків та зберегти репутацію.

Сукупність цих факторів зумовлює актуальність розробки автоматизованих класифікаційних систем, здатних з прийнятною точністю розрізняти дефектні та недефектні модулі на основі обчислюваних характеристик вихідного коду, а саме класичних

статичних метрик якості Чідамбера-Кемерера для об'єктно-орієнтованого коду (зокрема, складність класу WMC, глибину успадкування DIT, зчеплення CBO, відгук класу RFC, зв'язність LCOM, тощо) [4].

З урахуванням зазначених факторів стає зрозумілим, що сучасні автоматизовані системи виявлення вторгнень в комп'ютерні системи через програмне забезпечення потребують високих показників якості розпізнавання, стабільності результатів між етапами життєвого циклу ПЗ та швидкодію, яка не буде сповільнювати робочі бізнес процеси. Як показують попередні дослідження найкраще з цими задачами впораються методи машинного навчання, а саме ансамблеві класифікатори. Вони структурно краще за одиночні моделі і дозволяють поєднати прогнози кількох базових класифікаторів та знизити дисперсію помилки, підвищити здатність до узагальнення та отримати більш надійні оцінки ймовірностей [5-7].

За таких умов особливо актуальним стає відкрите питання побудови ансамблів для задач класифікації дефектів на основі статичних метрик, а також процес вибору базових моделей і процесу агрегації їх прогнозів з урахуванням складних нелінійних залежностей між статичними метриками і потенційною вразливістю. Найпоширеніший підхід, просте голосування по кількох базових моделях, не гарантує приросту якості над найкращою одиночною моделлю, оскільки структурно близькі базові класифікатори (наприклад, дерево-подібні методи) допускають корельовані помилки. Стекинг із мета-класифікатором дає кращі результати, проте типова практика обмежується невеликою кількістю гомогенних базових моделей на повному наборі ознак, що не реалізує потенціал диверсифікації [8-9].

Виходячи з наведених фактів, постає нагальна потреба у розробці методу ансамблевої класифікації дефектів програмного коду, який цілеспрямовано керує різноманітністю базових класифікаторів як на рівні набору даних та підпростору ознак, так і на рівні алгоритмічних родин базових моделей, поєднуючи це з адаптивним мета-навчанням для гарантованого монотонного підвищення якості на кожному етапі ускладнення архітектури.

### 1.2 Аналіз літературних джерел і визначення проблеми

Виявлення дефектів програмного коду на основі статичних метрик якості є усталеним напрямом досліджень у програмній інженерії. Класичні роботи спираються на набір метрик Чідамбера-Кемерера та похідні від них характеристики, які кількісно описують складність, зчеплення та зв'язність об'єктно-орієнтованого коду.

Перші підходи до класифікації базувались на одиночних статистичних та машинно-навчальних моделях – логістичній регресії, наївному баєсівському класифікаторі, методі опорних векторів. Такі моделі є інтерпретованими, проте демонструють обмежену точність на даних з мультиколінеарними ознаками та нелінійними залежностями, характерними для метрик складності коду. Подальший розвиток напряму пов'язаний із застосуванням методів машинного навчання, зокрема дерево-подібних методів та їх ансамблевих варіацій [10]. Сюди відносять метод випадкового лісу та методи градієнтного бустингу. Ці методи природним чином враховують нелінійні залежності та надлишковість ознак, однак, як показують емпіричні дослідження, прогнози дерево-подібних моделей у межах одного ансамблю схильні до корельованих помилок, що обмежує ефект від подальшого об'єднання таких моделей простим голосуванням.

Окремий напрям становлять гетерогенні ансамблі на основі стекінгу [11, 12], у яких прогнози кількох структурно різних базових моделей передаються мета-класифікатору, що навчається оптимально комбінувати їх. Стекінг, на відміну від голосування, здатний автоматично занижувати вагу слабких або корельованих гілок, що робить його більш стійким до різноманітності якості базових моделей. Водночас типова практика побудови стекінгу обмежується невеликою кількістю базових моделей на повному наборі ознак, не використовуючи диверсифікацію на рівні підпросторів ознак.

Питання керованої диверсифікації ансамблю активно досліджується у суміжних задачах класифікації (виявлення шахрайства, медична діагностика, виявлення мережових вторгнень [5]), де застосовуються метрики попарної розбіжності прогнозів, Q-статистика та Jaccard-відстань множин помилок для кількісної оцінки незалежності помилок базових класифікаторів. Показано, що цілеспрямоване підвищення різноманітності пулу класифікаторів – за рахунок різних підпросторів ознак, різних алгоритмічних родин або різних підвибірок навчальних даних – здатне покращити

якість ансамблю понад рівень, досяжний простим збільшенням кількості гомогенних базових моделей [13, 14]. Незважаючи на наявні результати, відкритим залишається питання систематичного поєднання трьох рівнів диверсифікації – підпросторів ознак [15–17], алгоритмічних родин та адаптивного мета-навчання [18, 19] – в єдиній архітектурі, яка б гарантувала монотонне підвищення якості класифікації дефектів програмного коду на кожному етапі ускладнення. Це й визначає напрям поточного дослідження.

### 1.3 Мета і завдання дослідження

Основна мета поточного дослідження полягає в розробці методу ансамблевої класифікації дефектів програмного коду на основі статичних метрик якості, який забезпечує контрольоване та монотонне підвищення якості розпізнавання при переході від одиночних класифікаторів до ансамблів зростаючої складності. Для досягнення поставленої мети необхідно дослідити наступні напрямки:

1. Аналіз характеристик набору даних – баланс класів, мультиколінеарність ознак, наявність шуму в розмітці – та визначення теоретичної стелі досяжної точності.

2. Дослідження ефективності класичних одиночних класифікаторів машинного навчання як орієнтиру для подальших ансамблевих побудов.

3. Перевірка здатності найпростіших ансамблевих методів з голосуванням перевищити якість найкращого одиночного класифікатора.

4. Розробка диверсифікованого стекінгу на основі кількох незалежних методів відбору ознак (кореляція Пірсона, взаємна інформація, LASSO, RFE) з архітектурною гарантією не гіршого результату, ніж у наївного стекінгу.

5. Розробка та обґрунтування розширеного диверсифікованого ансамблевого алгоритму на базі стекінгу, що доповнює диверсифікований стекінг алгоритмічно відібраними декорельованими гілками.

6. Експериментальна перевірка статистичної значущості приросту якості на кожному етапі за критерієм Вілкоксона та формування зведеного монотонного ланцюжка покрокового вдосконалення.

Передбачається, що виконання поставлених кроків дозволить отримати метод класифікації дефектів програмного коду, ефективність якого послідовно та статистично обґрунтовано зростає на кожному рівні ускладнення ансамблевої архітектури, а також сформулювати рекомендації щодо застосування диверсифікованих ансамблів у задачах статичного аналізу якості коду.

## 2. Матеріали і методи дослідження

Об'єктом дослідження є процес автоматизованого виявлення дефектів програмного коду. Предметом дослідження є класифікаційні методи ансамблевого машинного навчання.

Нехай  $f(x)$  – модель класифікації, що приймає вхідний вектор статичних метрик якості  $x \in \mathbb{R}^N$ , де  $N$  – кількість обраних метрик, і повертає оцінку ймовірності належності фрагменту програмного

коду до класу «дефектний». В такому випадку основною задачею стає формування вирішального правила бінарної класифікації:

$$\hat{y} = 1[P(\text{defect} | x) > t], \quad (1)$$

де  $\hat{y}$  – передбачена мітка класу;  $1[\cdot]$  – індикаторна функція;  $P(\text{defect} | x)$  – апостеріорна ймовірність дефекту;  $t$  – порогове значення (за замовчуванням  $t = 0,5$ ). Ідея використання ансамблевих класифікаторів передбачає поєднання прогнозів декількох базових моделей за допомогою процедури агрегації:

$$F(x) = g(f_1(x), f_2(x), \dots, f_M(x); \theta), \quad (2)$$

де  $f_1(x), \dots, f_M(x)$  – прогнози  $M$  базових класифікаторів для вхідного вектора  $x$ ;  $g(f; \theta)$  – функція агрегації (мета-алгоритм) із параметрами  $\theta$ , що поєднує прогнози базових моделей у підсумкове рішення  $F(x)$ ; конкретний вигляд  $g$  і природа  $\theta$  визначаються обраною схемою ансамблювання.

При цьому відбувається декомпозиція очікуваної помилки узагальнення на зміщення, дисперсію та незвідний шум:

$$E[(\hat{y} - y)^2] = \text{Bias}^2(\hat{y}) + \text{Var}(\hat{y}) + \sigma^2, \quad (3)$$

де математичне сподівання береться за реалізаціями навчальної вибірки  $D$ ,  $\hat{y} = \hat{y}(x; D)$  – прогноз моделі, побудованої на цій вибірці;  $\sigma^2$  – незвідна помилка, зумовлена шумом у даних.

Агрегація прогнозів декількох слабо корельованих класифікаторів зменшує доданок дисперсії, не збільшуючи зміщення. Саме тому ключовим фактором ефективності ансамблю є взаємна незалежність помилок його членів. Для оцінки якості класифікації використовується низка класичних метрик [20]. Основною з них обрано F1-міру як збалансований показник точності за умови рівної важливості обох класів:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (4)$$

Для оцінки якості ранжування незалежно від обраного порогу використовується площа під ROC-кривою, що визначається наступним чином з урахуванням частки істинно позитивних та хибно позитивних рішень:

$$\text{AUC}_{\text{ROC}} = \int_0^1 \text{TPR}(t) d\text{FPR}(t), \quad (5)$$

Крім того, в якості популярного збалансованого показника якості класифікації також прийнято брати коефіцієнт кореляції Метьюза, який є стійким до оптимістичних зміщень при незбалансованих помилках та розраховується наступним чином:

$$\begin{aligned} MCC &= \\ &= \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \end{aligned} \quad (6)$$

З метою визначення якості калібрування ймовірнісних оцінок, критичної для встановлення операційних порогів блокування у CI/CD-системах, використовується оцінка за допомогою метрики Brier-score, для якої менше значення свідчить про кращі результати:

$$\text{Brier} = \frac{1}{N} \sum_{i=1}^N (p_i - y_i)^2, \quad (7)$$

де  $p_i$  – передбачена ймовірність дефекту для  $i$ -го зразка,  $y_i$  – істинна мітка. Стабільність моделі між релізами оцінюється через стандартне відхилення F1-міри по групах крос-валідації.

Довірчі інтервали 95 % для метрик на ізольованій тестовій вибірці оцінено методом бутстрепа за 1000 повторними вибірками з поверненням. Набір розглянутих метрик дозволяє комплексно оцінити ефективність роботи класифікаційних моделей в процесі їх навчання та налаштування.

### 3. Результати розробки і дослідження методу виявлення дефектів програмного коду

#### 3.1 Математична модель процесу формування диверсифікаційно-орієнтованого ансамблю

В рамках дослідження для вирішення сформульованої проблеми пропонується побудувати ансамблевий класифікатор з урахуванням диверсифікації базових моделей за рахунок незалежного відбору підпросторів ознак та декорельованих алгоритмічних гілок, що забезпечує зниження дисперсії помилки узагальнення без збільшення зміщення. Процес формування такого класифікатора формалізується як двоетапна задача оптимізації, яка включає вибір архітектури ансамблю та навчання параметрів агрегуювальної функції.

Нехай  $P = \{f_1, f_2, \dots, f_N\}$  – пул кандидатів базових класифікаторів, отриманих навчанням різних алгоритмів на повному наборі ознак та на підпросторах ознак, відібраних незалежними методами. Задача першого етапу полягає у виборі підмножини  $S \subseteq P$  потужності  $K$ , яка одночасно максимізує середню якість і середню взаємну різноманітність гілок:

$$S^* = \underset{S \subseteq P, |S|=K}{\operatorname{argmax}} \left[ \frac{1}{|S|} \sum_{i \in S} Q(i) + \lambda \cdot D(S) \right], \quad (8)$$

де  $Q(i)$  – нормалізована оцінка якості кандидата  $i$  (наприклад,  $F1_{\text{norm}(i)}$ );  $D(S)$  – середня попарна розбіжність прогнозів усередині обраної підмножини  $S$  (означена нижче виразом (9));  $\lambda \geq 0$  – коефіцієнт балансу між якістю та різноманітністю на рівні підмножини (не слід плутати з коефіцієнтом  $\alpha$  жадібного алгоритму, який діє на рівні окремого кандидата).

$$D(S) = \frac{2}{|S|(|S| - 1)} \cdot \sum_{i < j, i, j \in S} \text{dis}(i, j), \quad (9)$$

Пряме розв'язання задачі (8) вимагає перебору  $C(N, K)$  підмножин  $i \in \text{NP}$ -складним навіть для помірних  $N$ . Це обґрунтовує застосування жадібного наближеного алгоритму, який послідовно нарощує  $S$ , додаючи на кожному кроці кандидата з найвищим

приростом комбінованого критерію – тобто виконує локально-оптимальну апроксимацію задачі (8).

Другий етап – навчання параметрів  $\theta$  агрегуювальної функції  $g$  на вже зафіксованій архітектурі  $S^*$ :

$$\theta^* = \operatorname{argmin}_{\theta} N^{-1} \times \sum_{n=1}^N L(y_n, g(f_1(x_n), \dots, f_K(x_n); \theta)) + \Omega(\theta), \quad (10)$$

де  $L(\cdot; \cdot)$  – функція втрат (логістична втрата для бінарної класифікації);  $\Omega(\theta)$  – регуляризаційний доданок, що обмежує складність мета-класифікатора;  $N$  – обсяг навчальної вибірки.

Підсумкова модель ансамблю визначається композицією розв'язків обох етапів:

$$F(x) = g(f_1(x), \dots, f_K(x); \theta^*), \{f_1, \dots, f_K\} = S^*. \quad (11)$$

Такий поділ на архітектурну (8) і параметричну (10) підзадачі формально відображає ключову відмінність запропонованого методу від наївних ансамблів: наївний стекинг розв'язує лише другу підзадачу на фіксованому  $S = P$ , тоді як диверсифікаційно-орієнтований ансамбль розв'язує обидві послідовно, що й забезпечує архітектурну гарантію не гіршого результату. Таким чином, фінальну схему запропонованого диверсифікаційно-орієнтованого ансамблю представлено на рис. 1.

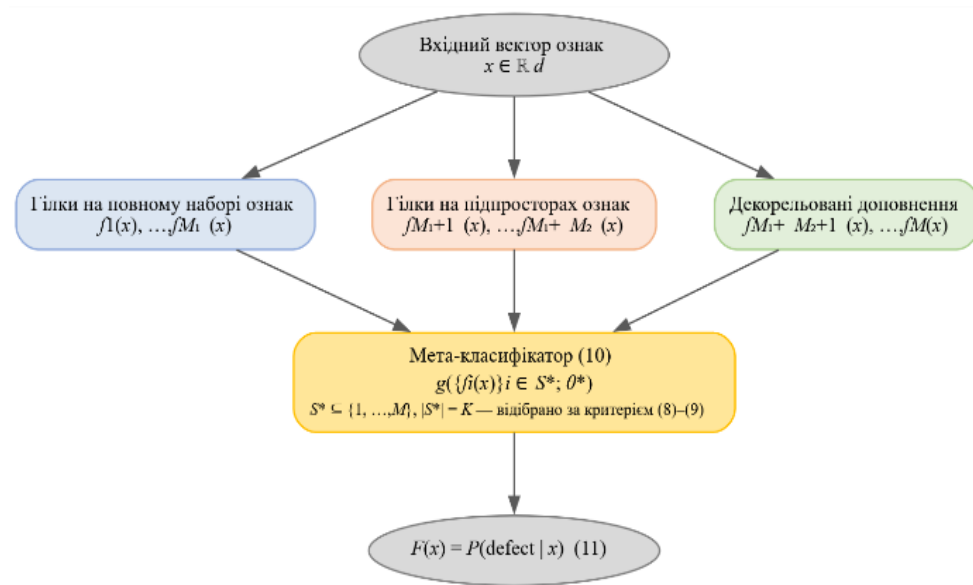


Рис. 1. Запропонована архітектура диверсифікаційно-орієнтованого ансамблю

### 3.2 Аналіз набору даних

Для проведення експериментального дослідження використано набір даних code\_defects\_identification [21], що містить 6052 спостереження та 21 статичну метрику якості коду, представлені у табл. 1, а також бінарну цільову мітку.

Важливою характеристикою набору даних є наявність шуму в розмітці: серед 6052 спостережень виявлено 286 дублікатів за вектором ознак, які мають різні мітки класу. Вони утворюють 150 конфліктних груп, що визначає теоретичну верхню межу точності, досягну будь-яким алгоритмом, незалежно від його складності [22]:

$$Accuracy_{max} = 1 - N_{conf}/N \approx 0,975, \quad (12)$$

де  $N_{conf}$  – кількість конфліктних спостережень, а  $N = 6052$  – загальний обсяг вибірки.

Подальший аналіз ознак виявив суттєву мультиколінеарність: 3 пари ознак мають коефіцієнт кореляції  $|r| > 0,9$ , а середнє позадіагональне значення  $|r|$  по всій матриці кореляцій складає 0,47, що представлено на рис. 2. Висока мультиколінеарність змусила застосувати методи відбору підпросторів ознак для зменшення надлишковості інформації, що використовується базовими класифікаторами.

Попередня обробка даних включала базові кроки з аналізу пропусків та викидів, логарифмічну трансформацію усіх 21 числових ознак для зменшення правосторонньої асиметрії розподілів з подальшою стандартизацією.

Таблиця 1 – Огляд метрик якості коду

| Група                    | Метрики                                                                                                              | Призначення                         |
|--------------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| Складність               | wmc, loc, totalMethods, totalFields, maxNestedBlocks, loopQty, comparisonsQty, tryCatchQty, returnQty                | Обсяг і структурна складність класу |
| Зчеплення та спадковість | cbo, rfc, nosi, dit                                                                                                  | Залежність від інших класів         |
| Зв'язність               | lcom                                                                                                                 | Узгодженість методів                |
| Лексика та стиль коду    | variablesQty, assignmentsQty, mathOperationsQty, numbersQty, stringLiteralsQty, parenthesizedExpsQty, uniqueWordsQty | Насиченість і читабельність         |

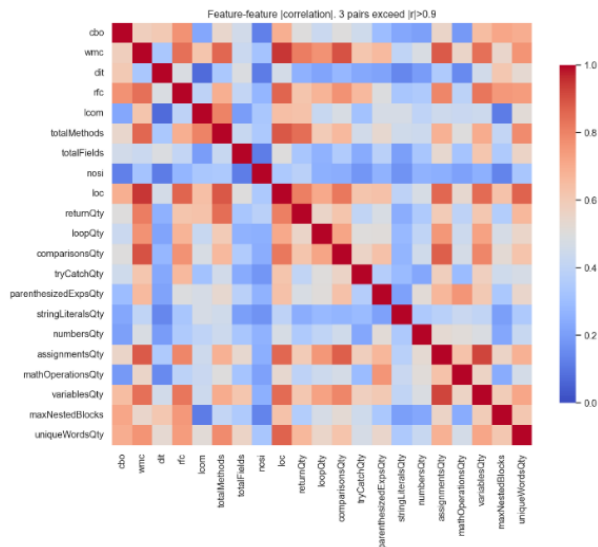


Рис. 2. Теплова карта кореляцій 21 ознаки

Після цього вибірку було розбито на навчальну (4841 зразок) та тестову (1211 зразків) частини у співвідношенні 80/20 зі збереженням пропорцій класів. Для побудови програмної реалізації та навчання моделей машинного навчання була використана мова програмування Python та бібліотека scikit-learn [23]. Аналіз результатів попередньої обробки даних показав, що найбільший приріст якості отримали моделі, які були чутливі до масштабу ознак і залежать від відстаней або градієнтів. Так F1 міра у

KNN зросла з 0,614 до 0,732, а у SVM – з 0,724 до 0,795. При цьому логістична регресія продемонструвала помірніший приріст з 0,759 до 0,785, тоді як дерево-подібний Gradient Boosting, як і очікувалося з огляду на його інваріантність до монотонних перетворень ознак, майже не відреагував, показавши приріст лише з 0,795 до 0,798.

### 3.3 Експериментальне дослідження ефективності пропонуваного алгоритму

Експериментальне частина дослідження складалась з п'яти послідовних етапів, які передбачали побудову різних моделей машинного навчання: від простих одиночних, до ансамблевих, в тому числі гетерогенних, структура яких була ускладнена за рахунок запропонованого методу з метою підвищення метрик якості.

Метою **першого етапу** є встановлення базового рівня якості для семи одиночних класифікаторів та визначення найкращої одиночної моделі як орієнтиру для побудови ансамблів в подальшому. Оцінювання виконувалося за допомогою 5-кратної стратифікованої крос-валідації на навчальній вибірці, у той час як тестова вибірка (20 % даних) залишалася ізольованою до фінального етапу порівняння. Найкращою одиночною моделлю виявився градієнтний бустинг (CV F1 = 0,798 ± 0,008). На другому місці була модель SVM зі значенням F1 (0,795), проте вона суттєво довше працювала в режимі тестування. Результати першого етапу наведені на рис. 3.

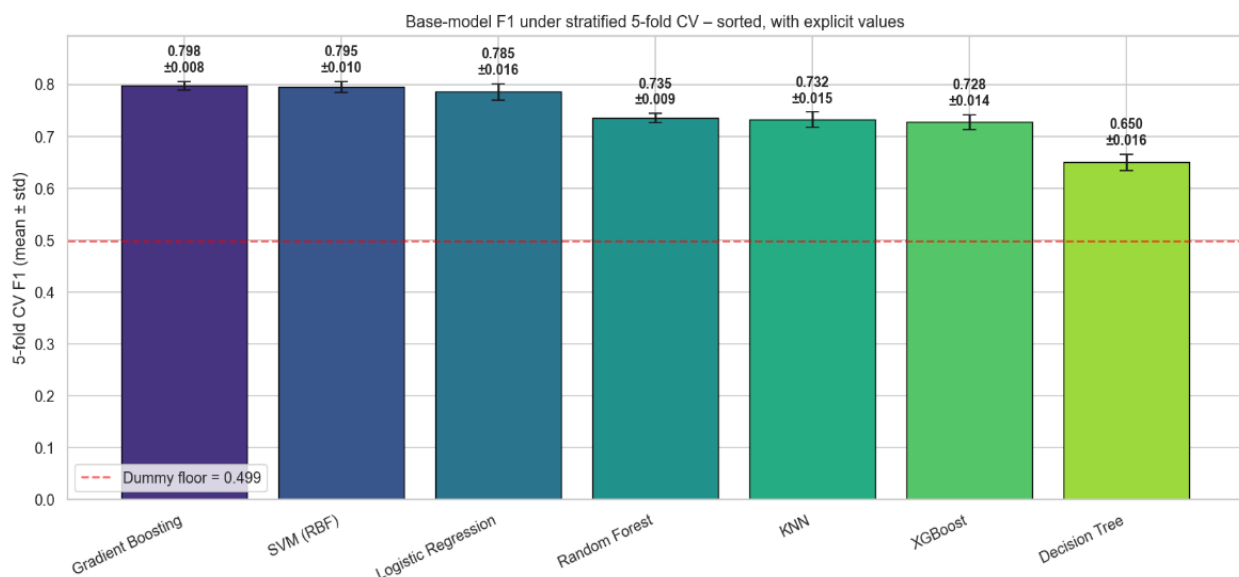


Рис. 3. Гістограма з результатами першого етапу дослідження

Метою **другого етапу** була перевірка того, чи дає проста агрегація гетерогенних базових класифікаторів приріст якості над найкращою одиночною моделлю. Для побудови наївних ансамблів обрано 6 базових класифікаторів (Logistic Regression, Random Forest, Gradient Boosting, XGBoost, SVM, KNN) на повному наборі ознак, поєднаних трьома способами: голосуванням за більшістю, усередненням ймовірностей та стекінгом із мета-класифікатором на основі логістичної регресії. За критерієм Вілкоксона

приріст Stacking-LR над Gradient Boosting є межовим за статистичною значущістю: різниця середніх значень F1 = +0,0136,  $p = 0,0625$  (мінімально досяжне значення  $p$  при  $n = 5$  розбиттях перехресної перевірки для одностороннього тесту). Діаграму розмаху F1 міри представлено на рис. 4.

Голосування (як просте, так і з усередненням ймовірностей) не перевищує якості найкращої базової моделі (F1 = 0,783–0,786 проти 0,798 у GB) – просте об'єднання прогнозів без мета-навчання не

гарантує покращення. Натомість стекинг із мета-класифікатором LR дає статистично межовий, але практично значущий приріст: CV F1 зростає з 0,798

до 0,811, Test F1 – з 0,780 до 0,797, ROC-AUC – з 0,811 до 0,859, а Brier score знижується з 0,170 до 0,151.

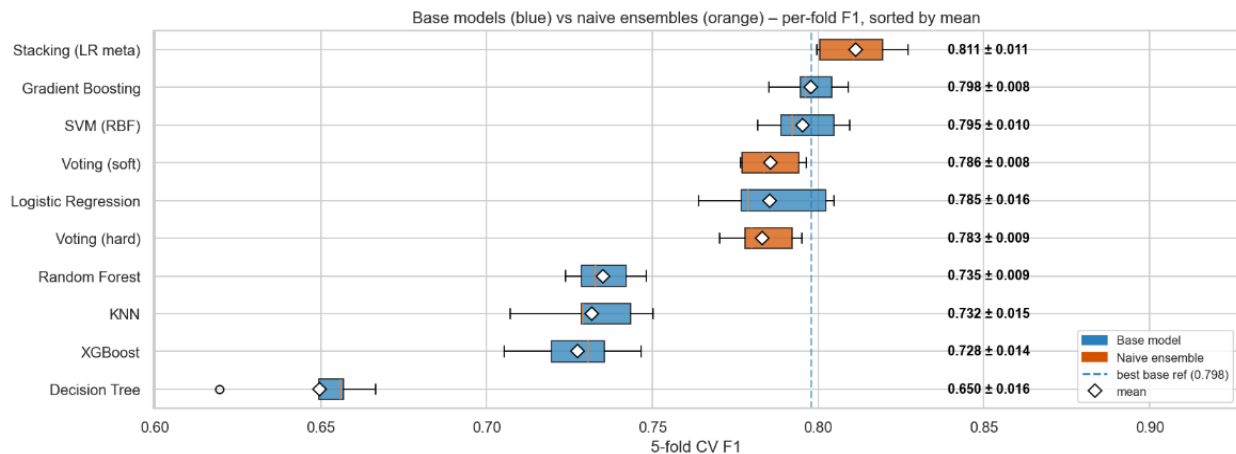


Рис. 4. Діаграма розмаху F1 міри: базові класифікатори (ліворуч) та наївні ансамблі (праворуч)

Обмежена ефективність голосування пояснюється через аналіз різноманітності базових класифікаторів: дерево-подібні методи (випадковий ліс, градієнтний бустинг, екстремальний градієнтний бустинг) кластеризуються за схожістю прогнозів, тобто допускають корельовані помилки. Це знижує ефективний приріст від простого об'єднання їхніх прогнозів. Таким чином, використання стекингу з мета-класифікатором продемонструвало перспективний приріст якості над найкращою одиночною моделлю, тоді як просте голосування такого приросту не забезпечило, що обґрунтовує доцільність подальшого розвитку саме стекингової архітектури на наступних етапах дослідження.

Метою **третього етапу** є підвищення різноманітності ансамблю шляхом поєднання кількох незалежних методів відбору ознак та побудова диверсифікованого стекингу з логістичною регресією в якості мета-класифікатора, здатного перевершити найкращий наївний метод. Для відбору інформативних підмножин ознак застосовано чотири незалежні методи: кореляцію Пірсона ( $|r| \geq \text{порогу}$ ), взаємну інформацію, L1-регуляризацію (LASSO) та рекурсивне виключення ознак (на основі градієнтного бустингу). Для кожного методу побудовано криву CV F1 від кількості ознак  $k$ . Лінійна кореляція Пірсона та взаємна інформація ранжують ознаки по-різному, що ілюструє рис. 5.

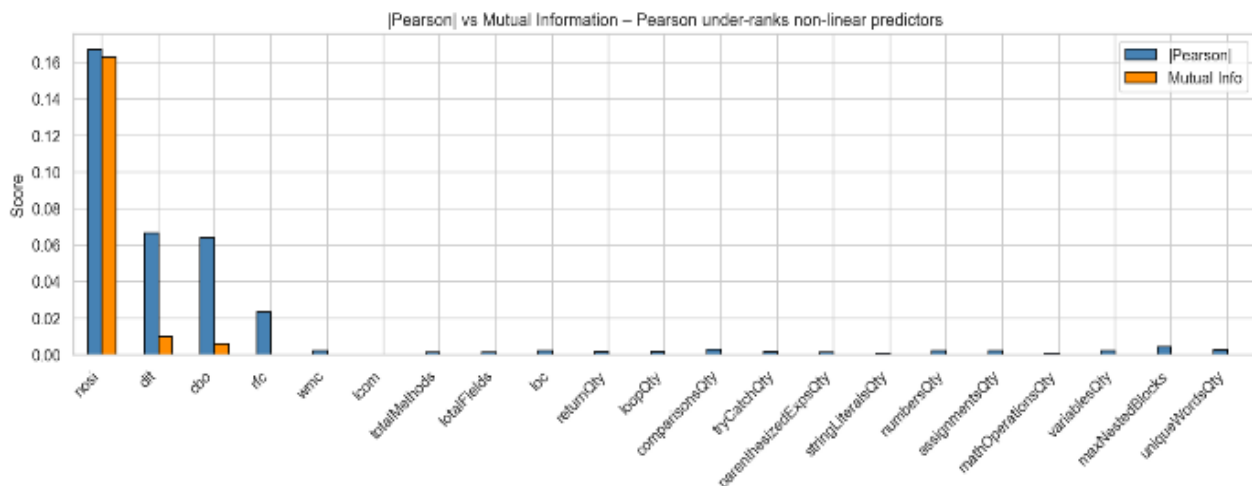


Рис. 5. Порівняння модуля коефіцієнта кореляції Пірсона  $|r|$  та взаємної інформації для 21 ознаки

За взаємною інформацією домінує ознака *posi*, тоді як за коефіцієнтом Пірсона порівнянню вагу мають також *dit*, *cbo* та *rfc*. Це підтверджує доцільність використання кількох методів відбору ознак одночасно.

Усі чотири методи конвергують до схожого набору основних ознак (*posi*, *dit*, *cbo*, *rfc*), що

підтверджує їхню справжню інформативність незалежно від конкретного методу оцінки значущості.

Таким чином, було прийнято рішення на **четвертому етапі** побудувати диверсифікований стекинг. Його архітектура складається з 9 гілок: 6 моделей (Logistic Regression, Random Forest, Gradient

Boosting, XGBoost, SVM, KNN) на повному наборі ознак, що забезпечують мінімальний рівень якості не нижче Stacking-LR, та 3 моделі на підпросторах ознак – Logistic Regression на LASSO-ознаках, Random Forest на MI-ознаках, XGBoost на RFE-ознаках. Мета-класифікатором було обрано логістичну регресію. В результаті отримано значення  $CV F1 = 0,813$ .

Важливим методологічним спостереженням є те, що диверсифіковане голосування без мета-навчання поступається найвчому стекінгу. Це підтверджує ключову гіпотезу дослідження: диверсифікація підпросторів ознак дає приріст лише у поєднанні з мета-класифікатором, здатним оптимально зважувати різні гілки, а не сама по собі.

Перед формуванням фінальної архітектури проведено допоміжне абляційне дослідження. Воно дозволило перевірити, чи здатний алгоритмічний відбір найкомплементарніших класифікаторів із великого пулу моделей з подальшою оптимальною агрегацією голосуванням перевершити диверсифікований стекінг без залучення мета-навчання.

Для цього абляційного дослідження було сформовано пул з 8 родин класифікаторів, які в тому числі розглядалися на попередніх кроках (Logistic Regression, LDA, Naive Bayes, Decision Tree, KNN, Gradient Boosting Trees, Random Forest, XGBoost), по 10 екземплярів кожної родини за допомогою беггінг-ансамблю з різними підвбірками. З цього підготовленого пулу жадібним алгоритмом відбирається  $K = 10$  моделей за критерієм, що розраховується наступним чином:

$$Q_{sel}(i) = \alpha \cdot F1_{norm}(i) + (1 - \alpha) \cdot \min_{j \in S} dis(i, j), \quad (13)$$

де  $F1_{norm}(i)$  – нормалізована оцінка якості кандидата  $i$ ;  $S$  – множина вже відібраних моделей;  $\alpha \in [0, 1]$  – коефіцієнт балансу між якістю та різноманітністю на рівні окремого кандидата.

Порівняння двох значень коефіцієнта  $\alpha$  показало мінімальну різницю: відбір з урахуванням різноманітності ( $\alpha = 0,5$ , м'яке голосування) дає  $CV F1 = 0,806 \pm 0,010$ , тоді як відбір лише за якістю ( $\alpha = 1,0$ ) дає  $CV F1 = 0,807 \pm 0,009$  – різниця 0,001. Водночас диверсифікація змінює склад відібраного набору моделей (наприклад, з {GBT:8, LR:1, NB:1} на {GBT:7, LR:2, NB:1}) – невеликий, але вимірюваний ефект на структуру ансамблю. Обмеженням підходу Custom Bagging Voter є те, що пул складається з відносно слабких моделей (найкраща окрема модель пулу:  $F1 \approx 0,800 < 0,813$  у Div. Stacking), а проста агрегація голосуванням не здатна суттєво перевершити найсильнішого члена пулу.

На останньому п'ятому етапі дослідження було затверджено фінальну архітектуру диверсифікованого ансамблю класифікаторів, яка складалася з 12 гілок: 9 гілок від диверсифікованого стекінгу (гарантія мінімального рівня 0,813) та 3 нові, алгоритмічно відібрані, декорельовані доповнення. Вони включали в себе моделі Extra Trees (300 дерев) на повному наборі ознак, Extra Trees (200 дерев) на підпросторі взаємної інформації та логістична

регресія зі слабшою регуляризацією. Мета-класифікатором обрано логістичну регресію з адаптивним підбором сили регуляризації.

Обґрунтування вибору Extra Trees полягає в тому, що, на відміну від Gradient Boosting та Random Forest, цей алгоритм обирає пороги розбиття вузлів дерев цілком випадково, а не за інформаційним критерієм, що генерує суттєво відмінні поверхні рішення та зменшує кореляцію помилок з іншими дерево-подібними методами в ансамблі.

Статистична значущість приросту: критерій Вілкоксона Ext. Div. Stacking проти Div. Stacking:  $\Delta mean F1 = +0,0115$ ,  $p = 0,0625$ .

Ext. Div. Stack перевершує Div. Stack за усіма контрольованими метриками одночасно. Найбільший приріст спостерігається за ROC-AUC (з 0,855 до 0,879) та за зниженням дисперсії між розбиттями перехресної перевірки на 40 %. Значення ст. відхилення = 0,006 є найнижчим серед усіх протестованих у дослідженні моделей, що робить Ext. Div. Stacking найстабільнішим класифікатором. Значення Brier = 0,140 – найкраще калібрування серед усіх моделей, тобто ймовірнісні оцінки Ext. Div. Stacking найбільш придатні для встановлення операційних порогів у CI/CD-системах.

Важливим результатом дослідження є монотонне зростання якості класифікації на кожному етапі ускладнення ансамблевої архітектури – від найкращого одиночного класифікатора до запропонованого методу Ext. Div. Stacking (табл. 2).

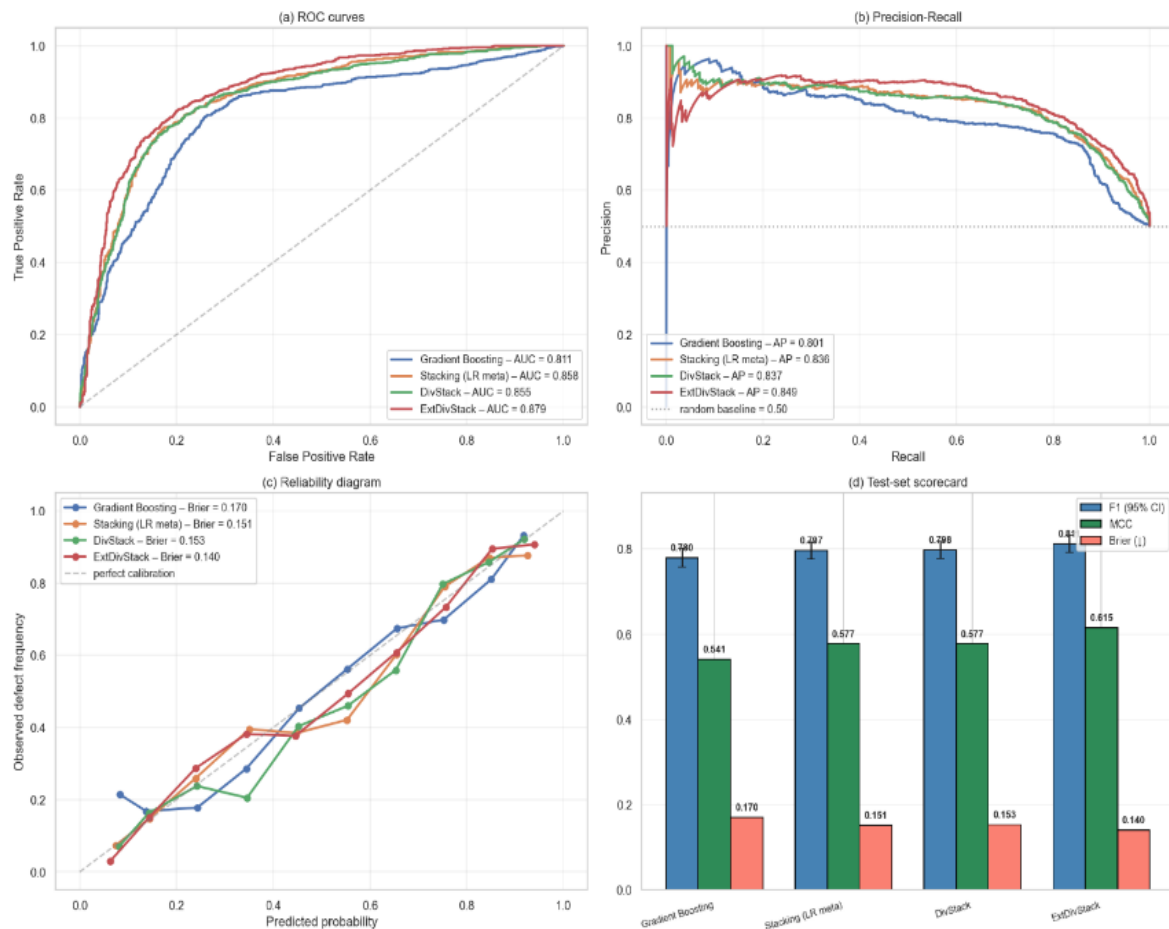
Таблиця 2 – Порівняння приросту якості на різних етапах дослідження

| Найкраща модель           | CV F1                | Test F1      | p-value       |
|---------------------------|----------------------|--------------|---------------|
| Gradient Boosting         | 0,798 ± 0,008        | 0,780        | –             |
| Stacking-LR               | 0,811 ± 0,011        | 0,797        | 0,0625        |
| Div. Stacking             | 0,813 ± 0,010        | 0,798        | 0,3125        |
| <b>Ext. Div. Stacking</b> | <b>0,824 ± 0,006</b> | <b>0,811</b> | <b>0,0625</b> |

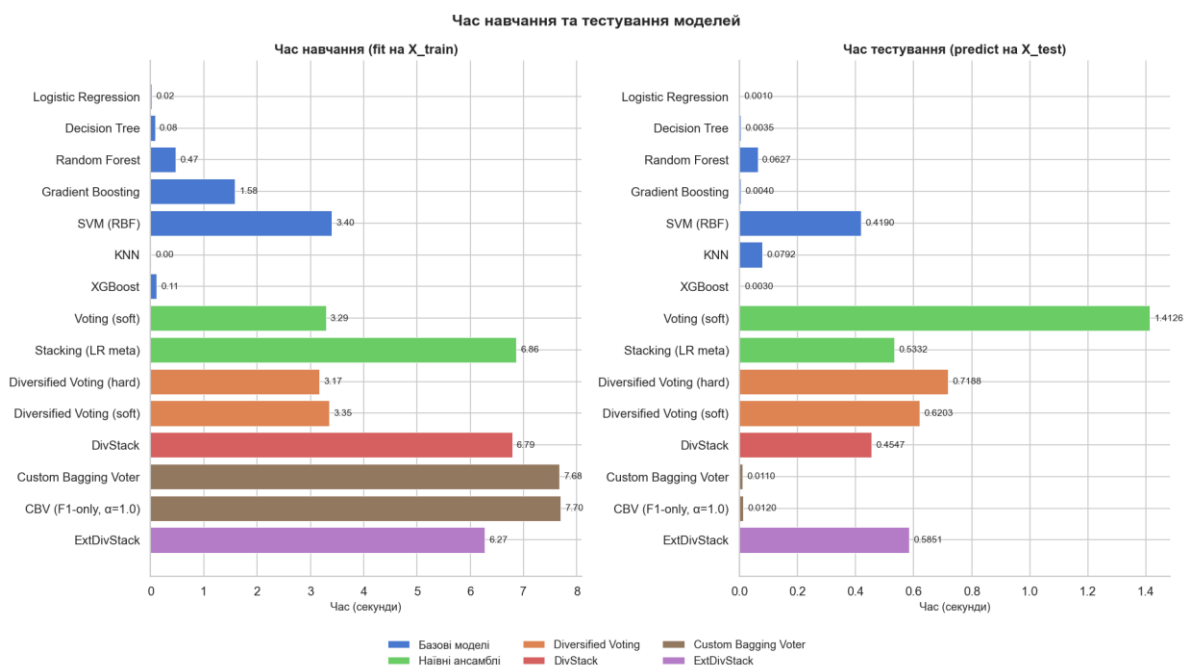
Загальний приріст на шляху від Gradient Boosting до Ext. Div. Stacking складає 0,026 за CV F1 (з 0,798 до 0,824) та 0,031 за Test F1 (з 0,780 до 0,811). Важливо, що етапи дослідження продемонстрували покращення метрик якості, що візуалізовано на рис. 6. Для остаточного підтвердження результатів чотири відібрані методи були порівняні на ізольованій тестовій вибірці. Результати цієї перевірки представлені у табл. 3. Фінальна модель розширеного диверсифікованого ансамблю продемонструвала найвищі значення F1, ROC-AUC, PR-AUC та MCC, а також найнижчі (найкращі) значення Brier score та ст. відхилення серед усіх чотирьох фіналістів, що підтверджує його переваги не лише на крос-валідації, а й на повністю незалежній тестовій вибірці. Окрім якості класифікації, для практичного застосування у CI/CD-системах важливою є обчислювальна вартість моделі, яка

характеризується часом навчання та часом тестування, що графічно представлені на рис. 7. Фінальна модель розширеного диверсифікованого ансамблю приблизно у 6 разів повільніша за

навчанням, ніж Gradient Boosting (5,45 с проти 0,90 с), проте, як вже зазначалося раніше, має кращі показники якості, а саме забезпечує зростання Test F1 з 0,780 до 0,811.



**Рис. 6.** Порівняння метрик якості найкращих моделей кожного з етапів:  
(а) ROC-криві з AUC; (б) PR-криві; (с) крива калібровки; (д) гістограми F1, MCC та Brier



**Рис. 7.** Час навчання (ліворуч) та час тестування (праворуч)  
для всіх досліджених моделей з кольоровим кодуванням за етапами

Таблиця 3 – Порівняння відібраних моделей на ізольованій тестовій вибірці

| Метрика        | GB          | Stack LR    | Div. Stack  | Ext. Div. Stack    |
|----------------|-------------|-------------|-------------|--------------------|
| F1             | 0,780       | 0,797       | 0,798       | <b>0,811</b>       |
| 95% ДІ         | 0,758-0,801 | 0,777-0,816 | 0,778-0,818 | <b>0,792-0,831</b> |
| ROC-AUC        | 0,811       | 0,858       | 0,855       | <b>0,879</b>       |
| PR-AUC         | 0,801       | 0,836       | 0,837       | <b>0,849</b>       |
| MCC            | 0,541       | 0,577       | 0,577       | <b>0,615</b>       |
| Brier ↓        | 0,170       | 0,151       | 0,153       | <b>0,140</b>       |
| Ст. відхилення | 0,008       | 0,011       | 0,010       | <b>0,006</b>       |

Час тестування такої моделі (0,522 с) є прийнятним для асинхронної безпекової перевірки в режимі реального часу у CI/CD-конвеєрі та не блокує основний процес збірки, а тому задовольняє вимогам.

#### 4. Обговорення результатів дослідження методу виявлення дефектів програмного коду

Отримані результати підтверджують, що цілеспрямоване керування різноманітністю ансамблю є дієвим і статистично обґрунтованим шляхом підвищення якості класифікації дефектів програмного коду.

Ключовим досягненням є демонстрація монотонного зростання якості на кожному етапі ускладнення ансамблевої архітектури: від найкращого одиночного класифікатора Gradient Boosting (CV F1 = 0,798) до запропонованого розширеного диверсифікованого ансамблю (CV F1 = 0,824).

В роботі продемонстровано, що приріст досягнуто не екстенсивним нарощуванням кількості моделей, а якісною зміною складу ансамблю за рахунок його диверсифікації. Додавання трьох декорельованих гілок на основі Extra Trees дало приріст ROC-AUC з 0,855 до 0,879 і водночас знизило дисперсію між окремими розбиттями перехресної перевірки на 40%, тобто підвищило не лише точність, а й стабільність моделі, що є важливою вимогою для практичного застосування в CI/CD-конвеєрах. Найкраще серед усіх протестованих моделей значення Brier score (0,140 проти 0,170 у Gradient Boosting) підтверджує також практичну придатність методу для встановлення операційних порогів автоматичного блокування збірки.

При аналізі отриманих результатів варто враховувати, що досяжна якість класифікації природно обмежена наявним у наборі даних шумом розмітки: 150 конфліктних груп спостережень формують верхню межу частки правильних рішень (Accuracy)  $\approx 0,975$ , до якої запропонована модель уже наближається, і подальше зростання точності на цьому наборі даних значною мірою залежатиме від

якості самої розмітки, а не лише від архітектури класифікатора. Крім того, приріст якості закономірно супроводжується дещо вищою обчислювальною вартістю навчання порівняно з базовим Gradient Boosting, що є звичним і очікуваним компромісом для ансамблевих методів і легко враховується при виборі конфігурації під конкретні ресурсні умови.

Перспективними напрямками подальших досліджень є перевірка узагальнюваності запропонованого методу на інших мовах програмування, типах проєктів і категоріях дефектів, зокрема вразливостях, а не лише структурних помилках, а також валідація на сучасних датасетах із суворішим контролем якості розмітки, що дозволило б точніше відокремити ефект архітектурних удосконалень від впливу шуму в даних. Не менш цікавим напрямом є перевірка стійкості диверсифікованого ансамблю до навмисних атак на класифікатор, оскільки декорельовані гілки за своєю природою ускладнюють перенесення змагальних збурень між моделями, що відкриває шлях до застосування подібної архітектури не лише для класифікації дефектів, а й для виявлення шкідливого коду, згенерованого великими мовними моделями, та розпізнавання спроб обходу автоматизованих систем контролю якості. Розвиток у цьому напрямі дозволить би поєднати задачі забезпечення якості коду та безпеки програмних систем у межах єдиної диверсифікованої ансамблевої архітектури.

#### 5. Висновки

У рамках дослідження розроблено та експериментально обґрунтовано вдосконалений метод класифікації дефектів програмного коду на основі диверсифікованих ансамблів класифікаторів зі статичними метриками якості. Дослідження виконано на наборі даних з 6052 спостережень та 21 статичної метрики зі збалансованим розподілом класів (50/50) та теоретичною верхньою межею точності, зумовленою шумом у розмітці (Accuracy  $\approx 0,975$ ).

Експериментально підтверджено, що проста агрегація гетерогенних базових класифікаторів (голосування) не дає приросту якості над найкращою одиночною моделлю (Gradient Boosting, CV F1 = 0,798), оскільки дерево-подібні базові класифікатори допускають корельовані помилки. Натомість стекінг із мета-класифікатором забезпечує статистично межовий, але практично значущий приріст якості.

Подальша диверсифікація стекінгу на основі підпросторів ознак, відібраних чотирма незалежними методами (кореляція Пірсона, взаємна інформація, LASSO, RFE), дозволила побудувати архітектуру з гарантованим мінімальним рівнем якості (не гірше наївного стекінгу) та незначним, проте стабільним додатковим приростом.

Найбільший приріст якості забезпечив запропонований метод розширеного диверсифікованого стекінгу, що доповнює диверсифікований стекінг алгоритмічно відібраними декорельованими гілками на основі Extra Trees.

Запропонований метод перевершує звичайний диверсифікований стекінг за усіма контрольованими метриками одночасно: CV F1 зріс з 0,813 до 0,824, Test F1 – з 0,798 до 0,811, ROC-AUC – з 0,855 до 0,879, а дисперсія знизилась на 40 %, забезпечуючи найвищу стабільність серед усіх моделей, розглянутих у дослідженні.

Загальний приріст якості на шляху від найкращого базового класифікатора до запропонованого методу Ext. Div. Stacking склав 0,026 за CV F1 та 0,031 за Test F1. Це підтверджує центральну гіпотезу дослідження: цілеспрямоване керування різноманітністю ансамблю на рівні підпросторів ознак та алгоритмічних родин, поєднане з адаптивним мета-навчанням, забезпечує монотонне підвищення якості класифікації дефектів програмного коду.

Практичне значення отриманих результатів дослідження полягає у можливості використання запропонованого метода в автоматизованих системах контролю якості програмного коду в рамках безперервного конвеєра доставки як асинхронний етап оцінки ризиків дефектності певного модуля. При цьому враховано як метрики якості, так і часові характеристики.

Наукова новизна дослідження полягає у тому, що вперше запропоновано архітектуру розширеного диверсифікованого стекінг-ансамблю, у якому диверсифікація здійснюється одночасно за рахунок

трьох компонентів, включаючи підпростір ознак, вибір алгоритмічної родини та адаптивне мета-навчання.

Крім того, набув подальшого розвитку процес застосування критеріїв різноманітності до формування складу ансамблю в задачах виявлення дефектів програмного коду на основі статистичних метрик.

### Конфлікт інтересів

Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження та його результати, представлені в даній статті.

### Фінансування

Дослідження проводилося без фінансової підтримки.

### Доступність даних

Рукопис має пов'язані дані у сховищі даних. Набір даних, використаний у дослідженні, є загальнодоступним і вказаний у тексті рукопису з посиланням на першоджерело

### Використання засобів штучного інтелекту

Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

### СПИСОК ЛІТЕРАТУРИ

1. National Institute of Standards and Technology (2026), "NIST Updates NVD Operations to Address Record CVE Growth", *NVD General Announcement*, available at: <https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth> (accessed 01.08.2026).
2. IBM Security, and Ponemon Institute (2025), "Cost of a Data Breach Report 2025", *IBM Security*, available at: <https://www.ibm.com/reports/data-breach> (accessed 01.08.2026).
3. MITRE Corporation, and Cybersecurity and Infrastructure Security Agency (2024), "2024 CWE Top 25 Most Dangerous Software Weaknesses", available at: [https://cwe.mitre.org/top25/archive/2024/2024\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html) (accessed 01.08.2026).
4. Chidamber, S.R., and Kemerer, C.F. (1994), "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, doi: <https://doi.org/10.1109/32.295895>
5. Hornostal, O., and Chelak, V. (2025), "Classification of network attacks using machine learning methods in conditions of training data imbalance", *Control, Navigation and Communication Systems*, vol. 3, no. 81, pp. 64–71, doi: <https://doi.org/10.26906/SUNZ.2025.3.064>
6. Gordieiev, O., Gordieieva, D., and Rainer, A. (2023), "Software Quality Assessment: Defect Life Cycle, Software Defect Profile, Its Types and Misalignments", *International Conference on Software Quality*, pp. 109–120, doi: [https://doi.org/10.1007/978-3-031-31488-9\\_6](https://doi.org/10.1007/978-3-031-31488-9_6)
7. Setia, S., Ravulakollu, K.K., Verma, K., Garg, S., Mishra, S.K. and Sharan, B. (2024), "Software Defect Prediction using Machine Learning", *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, pp. 560–566, doi: <https://doi.org/10.23919/INDIACom61295.2024.10498707>
8. Alazba, A., and Aljamaan, H. (2022), "Software defect prediction using stacking generalization of optimized tree-based ensembles", *Applied Sciences*, vol. 12, no. 9, art. 4577, doi: <https://doi.org/10.3390/app12094577>
9. Meng, F., Huang, R., and Wang, J. (2023), "A survey of software defects research based on deep learning", *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 1–5, doi: <https://doi.org/10.1109/ISCON57294.2023.10112194>
10. Olivares-Galindo, J. A., Sánchez-García, Á. J., Barrientos-Martínez, R. E., and Ocharán-Hernández, J. O. (2023), "Ensemble classifiers in software defect prediction: a systematic literature review", *2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pp. 1–8, doi: <https://doi.org/10.1109/CONISOFT58849.2023.00011>
11. Wolpert, D.H. (1992), "Stacked Generalization", *Neural Networks*, vol. 5, no. 2, pp. 241–259, doi: [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1)
12. Mehta, A., Kaur, A., and Kaur, N. (2024), "Optimizing software fault prediction using voting ensembles in class imbalance scenarios", *International Journal of Performability Engineering*, vol. 20, no. 11, pp. 676–687, doi: <https://doi.org/10.23940/ijpe.24.11.p4.676687>
13. Isong, B., and Igo, E. (2025), "Ensemble learning for software defect prediction: performance, practicality and future directions", *Journal of Information Systems and Informatics*, vol. 7, no. 3, pp. 2245–2291, doi: <https://doi.org/10.51519/journalisi.v7i3.1171>

14. Ali, M., Mazhar, T., Al-Rasheed, A., Shahzad, T., Ghadi, Y.Y., and Khan, M.A. (2024), "Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning", *PeerJ Computer Science*, vol. 10, art. e1860, doi: <https://doi.org/10.7717/peerj-cs.1860>
15. Ho, T.K. (1998), "The Random Subspace Method for Constructing Decision Forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, doi: <https://doi.org/10.1109/34.709601>
16. Tang, Y., Dai, Q., Yang, M., Du, T., and Chen, L. (2023), "Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm", *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 6, pp. 1967–1987, doi: <https://doi.org/10.1007/s13042-022-01740-2>
17. Gao, Y., and Yang, C. (2016), "Software defect prediction based on manifold learning in subspace selection", *Proceedings of the 1st International Conference on Intelligent Information Processing (ICIIP)*, ACM, New York, pp. 1–6, doi: <https://doi.org/10.1145/3028842.3028859>
18. Gavrylenko, S., and Hornostal, O. (2023), "Study of methods for improving the meta-algorithm of the bagging classifier", *Proceedings of the IEEE 4th KhPI Week on Advanced Technology*, pp. 1–6, doi: <https://doi.org/10.1109/KhPIWeek61412.2023.10312977>
19. Osei, S., Masegosa, A.R., and Masegosa, A.D. (2025), "Understanding the role of diversity in ensemble-based AutoML methods for classification tasks", *IEEE Access*, vol. 13, pp. 63566–63586, doi: <https://doi.org/10.1109/ACCESS.2025.3554093>
20. Chelak, V., Hornostal, O., Chelak, Y., and Gavrylenko, S. (2025), "Advanced methods for classification quality assessment leveraging ROC analysis and multidimensional confusion matrix", *Advanced Information Systems*, vol. 9, no. 1, pp. 24–34, doi: <https://doi.org/10.20998/2522-9052.2025.1.03>
21. Xu, J., Wang, F., and Ai, J. (2021), "Defect prediction with semantics and context features of codes based on graph representation learning", *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 613–625, doi: <https://doi.org/10.1109/TR.2020.3040191>
22. Croft, R., Babar, M.A., and Kholoosi, M.M. (2023), "Data quality for software vulnerability datasets", *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 121–133, doi: <https://doi.org/10.1109/ICSE48619.2023.00022>
23. Pedregosa, F., Varoquaux, G., Gramfort, A. et al. (2011), "Scikit-learn: machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, available at: <https://www.jmlr.org/papers/v12/pedregosa11a.html>

Received (Надійшла) 15.06.2026

Accepted for publication (Прийнята до друку) 26.08.2026

Publication date (Дата публікації) 25.09.2026

## ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

**Горносталь Олексій Андрійович** – PhD, доцент, доцент кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;

**Oleksii Hornostal** – PhD, Associate Professor, Associate Professor of Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;

e-mail: [oleksii.hornostal@khpi.edu.ua](mailto:oleksii.hornostal@khpi.edu.ua); ORCID Author ID: <http://orcid.org/0000-0001-5820-9999>;

Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=57216331944>.

**Челак Віктор Володимирович** – PhD, доцент, доцент кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;

**Viktor Chelak** – PhD, Associate Professor, Associate Professor of Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;

e-mail: [viktor.chelak@khpi.edu.ua](mailto:viktor.chelak@khpi.edu.ua); ORCID Author ID: <https://orcid.org/0000-0001-8810-3394>

Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=57189040595>.

### Diversified Classifier Ensembles for Software Defect Prediction Based on Static Quality Metrics

Oleksii Hornostal, Viktor Chelak

**Abstract. Background.** The relevance of the study lies in the need to improve existing approaches to source code defect classification in order to provide timely warning of potential execution problems and possible intrusions. **The object of the study** is the process of automated detection of software code defects. **The subject of the study** is ensemble machine learning classification methods. **The objective of the study** is to improve the quality of ensemble classifiers in software code defect classification tasks. **The research tasks are:** to analyze the characteristics of the dataset; to construct a sequence of ensemble models, from single classifiers to a diversified ensemble, for the classification of potential defects in software modules; and to confirm the statistical significance of the quality gain achieved at each stage. **The methods used** include data preprocessing techniques, feature selection methods, base machine learning methods, ensemble methods, diversification techniques, and statistical evaluation methods. **Results.** The study experimentally confirmed a monotonic increase in classification quality as the ensemble architecture was progressively complexified – from the best-performing base classifier, Gradient Boosting (CV F1 = 0.798), to the proposed extended diversified stacking method (CV F1 = 0.824), which outperformed the other models across all monitored metrics while achieving the lowest variance and the best probability calibration. **Conclusions.** Based on the results obtained, a method for software code defect classification is proposed that combines diversified stacking over feature subspaces with additional decorrelated branches based on Extra Trees and an adaptive logistic regression meta-classifier. The overall quality gain brought the achievable accuracy closer to the theoretical ceiling imposed by label noise in the dataset.

**Keywords:** software code defects; static quality metrics; machine learning; classification; ensemble methods; feature diversification; vulnerability detection; meta-classifier.