Юрій Кур'ян, Олександр Овсяк*, Наталія Гіссовська*
[1] НЛТУ України, Львів, Україна, ORCID: 0009-0007-3157-2289, yurkakuryan@gmail.com
[2] НЛТУ України, Львів, Україна. ORCID: 0000-0003-2620-1938, ovsjak@nltu.edu.ua
[3] НЛТУ України, Львів, Україна. ORCID: 0009-0006-1054-2411, gissovska.n@nltu.edu.ua

Адаптивна система для інтелектуального аналізу результатів автоматизованого тестування

Анотація. *Розроблено архітектурний підхід та систему для автоматизованого аналізу логів тестування з використанням методів машинного навчання. У зв'язку зі зростанням обсягів даних у сучасних CI/CD процесах, ручний аналіз стає неефективним. Запропонована система проводить кластеризацію для групування схожих помилок та класифікацію для виявлення аномалій і прогнозування результатів. Підхід дозволяє значно скоротити час на аналіз, підвищити швидкість реакції на збої та ефективність процесу тестування загалом.*

Ключові слова *– аналіз логів, машинне навчання, CI/CD, flaky tests, виявлення аномалій, кластеризація.*

Вступ. У сучасних процесах безперервної інтеграції та доставки (CI/CD) щоденно виконуються тисячі автоматизованих тестів, що генерують величезні масиви даних. Ефективний аналіз цих даних є ключовим для забезпечення якості програмного продукту, проте ручна обробка логів є часозатратною та неефективною. Основною проблемою є своєчасна ідентифікація повторюваних збоїв, виявлення нових, аномальних помилок та прогнозування нестабільності тестів, що вимагає автоматизованих інтелектуальних рішень.

Методологія та результати. Для вирішення цієї задачі запропоновано архітектуру адаптивної системи (рис. 1), яка реалізує повний цикл аналізу логів. Робота системи базується на публічному наборі даних[1], ключовою характеристикою цього датасету є сильний дисбаланс класів: лише 3.9 % записів відповідають збоям, тоді як 96.1 % успішним проходженням. Це створює реалістичні умови, що імітують справжні CI/CD процеси.Попередня обробка даних вклю-

* науковий керівник

чає кілька етапів. По-перше, було проведено очищення та нормалізацію текстових полів. По-друге, було згенеровано нові числові ознаки, такі як тривалість виконання тесту, текстові поля були об'єднані в єдину ознаку для подальшої векторизації.

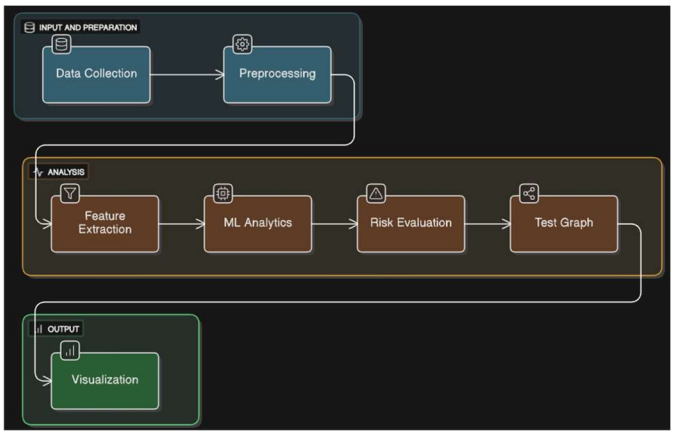

Рисунок 1. Архітектура системи аналізу логів

Для реалізації прототипу системи було використано мову програмування Python 3.10 та набір спеціалізованих бібліотек для аналізу даних і машинного навчання. Ключовим компонентом системи є модель класифікації на основі градієнтного бустингу, реалізована за допомогою бібліотеки LightGBM [2]. Для обробки текстових даних та їх векторизації було використано два підходи: класичний TF-IDF з бібліотеки Scikit-learn та семантичні вектори, згенеровані за допомогою попередньо навченої моделі paraphrase-multilingual-MiniLM-L12-v2 з бібліотеки Sentence-Transformers. Головним викликом дослідження став екстремальний дисбаланс класів у даних (~4 % збоїв), що призводило до поганої калібрації ймовірностей початкової моделі. Для вирішення цієї проблеми було застосовано двоступеневий підхід: постобробка за допомогою CalibratedClassifierCV з ізотонічною регресією для корекції ймовірностей та подальша оптимізація порогу рішення для максимізації метрики F1-score.

У ході дослідження було порівняно кілька підходів, зокрема векторизацію на основі TF-IDF та семантичних векторів (Sentence Embeddings), а також проведено реплікацію дослідження [3] на штучно збалансованому наборі даних (табл. 1).

* науковий керівник

Аналіз показав, що модель на основі Sentence Embeddings продемонструвала значно кращі результати на реалістичних, незбалансованих даних, досягнувши F1-score 0.31. Водночас реплікація дослідження на збалансованому датасеті 1:1, хоч і показала надзвичайно високий F1-score 0.97, ілюструє так звану "ілюзію високих метрик" і не є репрезентативною для реальних умов. Для об'єктивної оцінки отриманих результатів було проведено їх порівняння з типовими показниками, що наводяться у сучасних наукових публікаціях [4] у сфері аналізу програмного забезпечення. Отримані показники AUC ≈ 0.72 та F1-score ≈ 0.31 відповідають або перевищують середній рівень результатів у дослідженнях для задач з сильним дисбалансом класів. Це свідчить про високу якість розробленої моделі та її практичну корисність. Здатність системи автоматично ідентифікувати близько третини реальних збоїв при суттєвому зменшенні обсягу ручного аналізу є вагомим результатом для інтеграції в реальні CI/CD процеси

Таблиця 1. Зведені результати порівняльного аналізу моделей.

Метрика	TF-IDF Модель (незбалансовані дані)	**Sentence Embedding Модель (незбалансовані дані)**	Реплікація дослідження (збалансований датасет 1:1)
AUC Score	0.649	**0.720**	*Не повідомлявся*
F1-Score (Fail/Flaky)	0.237	**0.310**	**0.971**
Precision (Fail/Flaky)	0.208	**0.273**	**0.944**
Recall (Fail/Flaky)	0.276	**0.321**	**1.000**
Таблиця 1 – Зведені результати порівняльного аналізу			

Для перевірки практичної цінності розробленого підходу було реалізовано програмний прототип, що функціонує як автоматизований конвеєр обробки. Система включає модуль для синтаксичного розбору даних безпосередньо зі звітів у поширеному форматі Allure Report, що дозволяє легко інтегрувати її в існуючі процеси безперервної інтеграції. Найкраща модель, навчена на семантичних векторах, була інтегрована в цей конвеєр. Апробація на реальних звітах продемонструвала високу ефективність: система успішно ідентифікувала збої відповідно до очікуваних показників точності та повноти. На практиці розроблена модель використовується як система оцінки ризиків. Для кожного окремо-

* науковий керівник

го тесту, незалежно від його фінального статусу, розраховується коефіцієнт ризику, який потім порівнюється з оптимальним порогом для надання фінального вердикту. Такий підхід дозволяє не тільки знаходити очевидні збої, але й, що є більш цінним, пріоритезувати аналіз, автоматично виявляючи тести з аномально високим ризиком, навіть якщо вони на даний момент пройшли успішно. Це дає змогу інженерам проактивно ідентифікувати потенційно нестабільні тести та виправляти приховані проблеми ще до того, як вони стануть критичними.

Для подальшого покращення системи доцільно розглянути кілька напрямків. По-перше, реалізація кластеризації збоїв за допомогою алгоритмів, як-от HDBSCAN, дозволить автоматично групувати помилки за типами. По-друге, варто дослідити альтернативні методи боротьби з дисбалансом, зокрема SMOTE, або підхід, який розглядає послідовності журналів як послідовності природної мови та використовує тонко налаштовану модель BERT, як у дослідженні [5]. Нарешті, використання інструментів інтерпретації, наприклад SHAP, може надати глибше розуміння факторів, що впливають на ризик збою.

Висновок. Запропонована система забезпечує автоматичне групування помилок за їх суттю, дозволяє виявляти нові типи збоїв на ранніх етапах та надає інструмент для оцінки стабільності тестів. Архітектурне рішення створює умови для швидкого реагування команди розробки на проблеми та значно знижує витрати часу на рутинний аналіз логів, підвищуючи загальну ефективність процесу забезпечення якості.

Список використаних літературних джерел

[1] The Westermo test results dataset. https://github.com/westermo/test-results-dataset

[2] LightGBM documentation. https://lightgbm.readthedocs.io

[3] Pinto, G., et al. "A large-scale empirical study on the effects of flaky tests." 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2016.

* науковий керівник

[4] Akli, A., Haben, G., Habchi, S., Papadakis, M., & Le Traon, Y. (2022). Flakycat: predicting flaky tests categories using few-shot learning. arXiv preprint arXiv:2208.14799.

[5] Chen, S., & Liao, H. (2022). BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. Applied Sciences.

Adaptive system for intelligent analysis of automated testing results

Yurii Kurian, Oleksandr Ovsyak, Natalia Gissovska

An architectural approach and system have been developed for the automated analysis of testing logs using machine learning methods. With the growing volume of data in modern CI/CD processes, manual analysis becomes inefficient. The proposed system performs clustering to group similar errors and classification to detect anomalies and predict outcomes. This approach significantly reduces analysis time, increases responsiveness to failures, and enhances the overall efficiency of the testing process.

* науковий керівник