МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

МИСЬ Андрій Русланович

Метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання / Method for Risk Prediction in Software Projects Using Machine Learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма – Управління проєктами

Кваліфікаційна робота

Виконав студент групи КНУПм-21
А.Р. Мись
_____

Науковий керівник:
к.е.н., доцент Г.М. Гладій
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.М. Васильків

ТЕРНОПІЛЬ – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Управління проєктами

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
МИСЬ Андрій Русланович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання / Method for Risk Prediction in Software Projects Using Machine Learning
керівник роботи к.е.н., доцент Г.М. Гладій.
затверджені наказом по університету від 12 грудня 2023 року № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 4 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- предметна область дослідження;
- аналіз моделей розробки програмного забезпечення;
- аналіз підходів до управління ризиками в сучасних програмних проєктах;
- постановка задачі дослідження;
- концептуальні основи методу прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання;
- збір даних та формування навчальної і тестової вибірок;
- система рекомендацій для розробки програмного забезпечення;
- методика проведення експериментальних досліджень;
- реалізація класифікаторів машинного навчання;
- результати експериментальних досліджень.

5. Перелік графічного матеріалу у роботі
- діаграма потоку даних;
- архітектура системи.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 30.11.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2024 р.	

Студент _____ А.Р. Мись
підпис

Керівник роботи _____ к.е.н., доцент Г.М. Гладій
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Управління проєктами» написана обсягом в 74 сторінок і містить 7 ілюстрацій, 2 таблиці, 2 додатки та 36 використаних джерел.

Метою кваліфікаційної роботи є підвищення ефективності управління проєктами розробки програмного забезпечення шляхом впровадження системи прогнозування ризиків на основі машинного навчання.

Методи досліджень: аналіз літературних джерел, методи машинного навчання, методи обробки та підготовки даних, емпіричний аналіз, методи статистичного аналізу, моделювання та оптимізація, візуалізація даних.

Результати дослідження: полягають у удосконаленні методу прогнозування ризиків у проєктах розробки програмного забезпечення за допомогою інтеграції алгоритмів машинного навчання, зокрема таких як машина опорних векторів, метод k-ближчих сусідів, штучна нейронна мережа з багатошаровим персептроном та випадковий ліс, що дозволило підвищити точність прогнозування та знизити ймовірність невдач у реалізації програмних проєктів. Запропонована система дозволяє автоматизувати процес оцінки ризиків та вибору відповідної моделі розробки програмного забезпечення залежно від вимог та умов проєкту, що сприяє зниженню ймовірності невдач, оптимізації управлінських рішень та підвищенню ефективності роботи команд розробників.

Результати роботи можуть бути використані для впровадження системи в середовищах з високими вимогами до надійності, таких як IoT та ін.

Ключові слова: ПРОГНОЗУВАННЯ, РИЗИКИ ПРОЄКТІВ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАШИННЕ НАВЧАННЯ, ОПТИМІЗАЦІЯ УПРАВЛІНСЬКИХ РІШЕНЬ, ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ КОМАНДИ.

ABSTRACT

Qualification work on the topic «Method for Risk Prediction in Software Projects Using Machine Learning» for Master's degree on speciality 122 «Computer Science» educational and professional program «Project Management» is written on 74 pages and it contains 7 figures, 2 tables, 2 annexes and 36 sources.

The purpose of this qualification work is to improve the efficiency of software development project management by implementing a risk prediction system based on machine learning.

Research methods: literature review, machine learning methods, data processing and preparation methods, empirical analysis, statistical analysis methods, modeling and optimization, and data visualization.

Research results: the research involves enhancing the method for predicting risks in software development projects by integrating machine learning algorithms, including support vector machines, k-nearest neighbors, artificial neural networks with a multilayer perceptron, and random forest. This integration has improved prediction accuracy and reduced the likelihood of project failures. The proposed system automates the risk assessment process and the selection of appropriate software development models based on the project's requirements and conditions. This contributes to reducing failure rates, optimizing managerial decisions, and enhancing the efficiency of developer teams.

The results of the work can be applied to implement the system in environments with high reliability requirements, such as IoT and others.

Keywords: PREDICTION, PROJECT RISKS, SOFTWARE, MACHINE LEARNING, OPTIMIZATION OF MANAGERIAL DECISIONS, TEAM EFFICIENCY IMPROVEMENT.

ЗМІСТ

ВСТУП

Актуальність теми. Розвиток інформаційних технологій і динамічний прогрес у розробці програмного забезпечення призводять до зростання складності управління проєктами, що вимагає нових підходів до оцінки та управління ризиками. Успіх програмного продукту залежить від ефективного виконання проєкту, який часто стикається з ризиками, пов'язаними з непередбачуваними змінами у вимогах, затримками, технічними проблемами та обмеженими ресурсами. Сучасні програмні проєкти, такі як ті, що залучають технології Інтернету речей, граничних і хмарних обчислень, створюють додаткові виклики через високі вимоги до надійності, адаптивності та безпеки.

Традиційні методи управління ризиками зазвичай покладаються на експертні оцінки, що часто обмежує їхню точність і ефективність в умовах динамічних змін. Використання машинного навчання відкриває нові можливості для автоматизації процесів прогнозування ризиків, аналізу великих обсягів даних і виявлення прихованих закономірностей. Інтеграція таких інструментів дозволяє розробникам та менеджерам приймати обґрунтовані рішення, мінімізуючи потенційні втрати і підвищуючи якість програмного продукту.

Мета і завдання дослідження. Метою кваліфікаційної роботи є підвищення ефективності управління проєктами розробки програмного забезпечення шляхом впровадження системи прогнозування ризиків на основі машинного навчання. Це дозволить зменшити ймовірність невдач, оптимізувати вибір моделей розробки, покращити процес прийняття рішень та знизити вплив потенційних ризиків на строки виконання та якість кінцевого продукту.

Для досягнення поставленої мети потрібно виконати **наступні завдання**:

– провести аналіз літературних джерел та існуючих підходів до прогнозування ризиків у проєктах розробки програмного забезпечення, зокрема методів машинного навчання, які використовуються для оцінки ризиків;

– розробити метод прогнозування ризиків у проєктах розробки програмного забезпечення за допомогою інтеграції алгоритмів машинного

навчання;

– провести збір та обробку даних, отриманих від експертів з розробки програмного забезпечення, а також формування навчальних та тестових вибірок для побудови моделей машинного навчання;

– розробити та налаштувати моделі машинного навчання, такі як машина опорних векторів, метод k-ближчих сусідів, штучна нейронна мережа з багатошаровим персептроном та випадковий ліс для прогнозування ризиків;

– провести навчання моделей та їх оцінку на основі тестових даних з використанням метрики продуктивності;

– провести порівняння ефективності різних моделей та вибір найкращої моделі для прогнозування ризиків у проєктах розробки програмного забезпечення.

Об'єктом дослідження є процес управління ризиками в проєктах розробки програмного забезпечення з використанням методів машинного навчання.

Предметом дослідження є методи та алгоритми машинного навчання, які використовуються для прогнозування ризиків у проєктах розробки програмного забезпечення, а також підходи до оцінки та вибору оптимальної моделі розробки на основі аналізу ризиків.

Методи досліджень. В роботі використовуються наступні методи досліджень: аналіз літературних джерел, методи машинного навчання, методи обробки та підготовки даних, емпіричний аналіз, методи статистичного аналізу, моделювання та оптимізація, візуалізація даних.

Наукова новизна одержаних результатів полягає в удосконаленні методу прогнозування ризиків у проєктах розробки програмного забезпечення за допомогою алгоритмів машинного навчання, зокрема таких як машина опорних векторів, метод k-ближчих сусідів, штучна нейронна мережа з багатошаровим персептроном та випадковий ліс, що дозволило підвищити точність прогнозування та знизити ймовірність невдач у реалізації програмних проєктів.

Практичне значення отриманих результатів полягає у створенні ефективної системи прогнозування ризиків у проєктах розробки програмного

забезпечення на основі методів машинного навчання. Запропонована система дозволяє автоматизувати процес оцінки ризиків та вибору відповідної моделі розробки програмного забезпечення залежно від вимог та умов проєкту, що сприяє зниженню ймовірності невдач, оптимізації управлінських рішень та підвищенню ефективності роботи команд розробників. Отримані результати можуть бути використані для впровадження системи в середовищах з високими вимогами до надійності, таких як IoT та ін.

Публікації та апробація КР. Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах:

– міжнародної мультидисциплінарної наукової інтернет-конференції «Світ наукових досліджень» (випуск 35), 20-21 листопада 2024 р.;

– міжнародної наукової Інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (випуск 94), 11-12 грудня 2024 р.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ УПРАВЛІННЯ РИЗИКАМИ У ПРОЄКТАХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Опис предметної області управління ризиками у проєктах розробки програмного забезпечення

Предметна область дослідження охоплює управління ризиками у проєктах розробки програмного забезпечення, зосереджуючись на застосуванні методів машинного навчання для прогнозування ризиків і надання рекомендацій щодо вибору моделі розробки. У сучасному цифровому світі, коли технології постійно розвиваються, розробка програмного забезпечення стикається з численними викликами, такими як мінливість вимог, зростаючий вплив нових технологій, нестабільні ресурси та складність проєктів. Це ускладнює управління розробкою програмного забезпечення і підвищує ризик невдачі проєктів.

Перелік ризиків проєктів розробки програмного забезпечення можна умовно поділити на кілька основних категорій [1, 2, 15]:

1. Технічні ризики:

– недостатня якість вихідних вимог (неповнота, неоднозначність);

– невідповідність обраної технології або інструментів проєктним вимогам;

– проблеми з інтеграцією компонентів або систем;

– відмова обладнання або збій інфраструктури;

– недостатній рівень тестування (відсутність покриття тестами ключових функцій).

– технічна застарілість або несумісність програмного забезпечення.

2. Організаційні ризики:

– нестача кваліфікованих фахівців у команді;

– неефективна комунікація між членами команди або з замовником;

– відсутність чіткої стратегії управління проєктом;

– втрата ключових співробітників під час виконання проєкту;

– недостатня мотивація команди або перевантаження співробітників;

3. Фінансові ризики:

– перевищення бюджету проєкту;

– недооцінка витрат на розробку та підтримку;

– непередбачені витрати на ліцензії або додаткові ресурси4

– невиправдане скорочення бюджету, що впливає на якість продукту.

4. Часові ризики:

– недотримання графіку розробки через затримки у виконанні етапів;

– неврахування залежностей між завданнями або їх неправильно оцінений час виконання;

– затримки через зміни у вимогах або погодження з замовником;

– затримки в прийнятті рішень через відсутність відповідальних осіб.

5. Ризики, пов’язані з вимогами:

– постійні зміни у вимогах під час реалізації проєкту;

– недостатня участь замовника у формуванні та уточненні вимог;

– конфлікти між замовником та виконавцем щодо інтерпретації вимог.

6. Ризики безпеки:

– вразливості в розробленому програмному забезпеченні;

– неавторизований доступ до даних або системи;

– неправильне налаштування системи безпеки;

– неврахування вимог до конфіденційності даних.

7. Зовнішні ризики:

– вплив змін у законодавстві або нормативних актах;

– нестабільність ринку або економічна криза;

– залежність від третіх сторін (постачальників, підрядників, партнерів);

– втрата доступу до критичних сервісів або ресурсів.

8. Ризики управління:

– недостатній контроль за виконанням проєкту;

– неефективне управління змінами в проєкті;

– відсутність плану реагування на ризики.

– недоліки у звітності та моніторингу прогресу.

9. Ризики, пов’язані з користувачами:

– невідповідність кінцевого продукту очікуванням користувачів;

– складність в адаптації користувачів до нового програмного забезпечення;

– відсутність зворотного зв’язку від кінцевих користувачів.

Цей перелік може бути доповнений залежно від специфіки проєкту, обраних технологій і методологій розробки.

Управління ризиками у проєктах розробки програмного забезпечення є критично важливим елементом для забезпечення успішного завершення проєктів. Це включає ідентифікацію, аналіз та оцінку потенційних ризиків, таких як затримки у виконанні, перевищення бюджету, технічні проблеми, відсутність відповідності вимогам замовника та інші загрози. Традиційні підходи до управління ризиками зазвичай базуються на експертних оцінках і застосовуються вручну, що може призводити до суб'єктивних рішень і недостатньої точності у виявленні ризиків на ранніх етапах проєкту.

З появою новітніх технологій, таких як методи машинного навчання, управління ризиками у проєктах розробки програмного забезпечення зазнало значних змін. Машинне навчання надає можливість автоматизувати процес аналізу ризиків, що дозволяє обробляти великі обсяги даних та виявляти приховані закономірності, які складно врахувати при використанні традиційних методів. Це дає змогу забезпечити більш точне прогнозування ризиків, а також своєчасно коригувати стратегії розробки для зниження ймовірності виникнення загроз.

Застосування машинного навчання в управлінні ризиками включає використання різних алгоритмів, таких як машина опорних векторів (SVM), метод k-ближчих сусідів (k-NN), штучні нейронні мережі (ANN), зокрема багатошаровий персептрон (MLP), та алгоритми ансамблевого навчання, такі як випадковий ліс (Random Forest). Ці алгоритми дозволяють виявляти складні взаємозв’язки між різними факторами ризику та створювати моделі для їхнього

прогнозування на основі історичних даних, що були зібрані з попередніх проєктів.

Інтеграція систем прогнозування ризиків з управлінням проєктами розробки програмного забезпечення забезпечує переваги для менеджерів проєктів, власників продуктів та розробницьких команд. Це дозволяє приймати обґрунтовані рішення щодо стратегій розробки, забезпечувати адаптацію до нових вимог і змін у проєкті, а також мінімізувати потенційні втрати через непередбачувані події.

Таким чином, предметна область дослідження охоплює комплексний підхід до прогнозування ризиків у проєктах розробки програмного забезпечення з використанням методів машинного навчання, аналізу даних та розробки систем рекомендацій. Це дослідження спрямоване на вирішення актуальних проблем у сфері управління програмними проєктами, що дозволяє досягти більшої точності та ефективності у розробці сучасних програмних продуктів.

1.2 Аналіз моделей розробки програмного забезпечення

Під час дослідження були відібрані чотири основні моделі розробки програмного забезпечення: водоспадна, інкрементна, еволюційна та гнучкі методології. Ці моделі забезпечують різні підходи до управління розробкою програмного забезпечення та задоволення вимог замовників. Вибір відповідної моделі залежить від специфіки проєкту, визначення вимог та рівня ризиків, які можуть виникнути під час виконання проєкту [1, 2, 15].

– Водоспадна модель.

Водоспадна модель є класичним підходом до розробки програмного забезпечення, що передбачає послідовне виконання етапів, таких як визначення вимог, планування, проектування, кодування, тестування та впровадження, завершуючи підтримкою програмного продукту. Цей підхід є ефективним у випадках, коли вимоги до програмного забезпечення чітко визначені та стабільні. Завдяки цьому знижується рівень невизначеності на кожному етапі,

оскільки кожен наступний етап не починається, поки не завершений попередній. Проте, недоліком водоспадної моделі є її низька гнучкість. Зміни на пізніх етапах розробки можуть вимагати значних зусиль та витрат, оскільки повернення до попередніх етапів є складним. Це робить модель менш придатною для динамічних проєктів, де вимоги можуть змінюватися в процесі розробки [1, 2, 15].

– Інкрементна модель.

Інкрементна модель розробки програмного забезпечення забезпечує створення та поступове постачання функціональних можливостей через серію інкрементів або релізів. Кожен новий інкремент доповнює попередній, що дозволяє поступово розширювати функціональність продукту, надаючи користувачам можливість оцінити окремі частини системи на ранніх етапах. Такий підхід дозволяє отримувати зворотний зв'язок від користувачів, що сприяє кращому розумінню їхніх потреб та вимог. Інкрементна модель є ефективною у ситуаціях, коли неможливо визначити усі вимоги на початку проєкту, але водночас потрібно поступово надавати користувачу корисний продукт. Водночас, цей підхід потребує ретельного планування кожного інкременту, щоб забезпечити сумісність та інтеграцію між частинами продукту [1, 2, 15].

– Еволюційні процеси: прототипування та спіральна модель.

Еволюційні процеси включають підходи прототипування та спіральної моделі, які орієнтовані на поступовий розвиток програмного забезпечення з урахуванням змін та уточнень вимог. Прототипування передбачає створення спрощеної версії продукту (прототипу) для швидкого отримання зворотного зв'язку від користувачів. Це дає змогу розробникам та замовникам чіткіше визначити вимоги та виявити недоліки на ранньому етапі розробки. З іншого боку, спіральна модель акцентує увагу на управлінні ризиками. Вона включає циклічний підхід, де кожна ітерація додає нові функціональні можливості та одночасно знижує рівень ризику завдяки поступовому уточненню вимог та регулярним оцінкам прогресу. Спіральна модель є особливо корисною для проєктів з високим рівнем ризиків, оскільки вона дозволяє виявляти та

пом'якшувати ризики на кожній ітерації, забезпечуючи високу надійність розробки [1, 2, 15].

– Гнучкі методології: екстремальне програмування та Scrum.

Гнучкі методології, такі як екстремальне програмування та Scrum, спрямовані на швидку адаптацію до змін у вимогах та тісну взаємодію з замовниками. Екстремальне програмування включає низку практик, таких як часті релізи, постійне тестування, парне програмування та безперервна інтеграція, що забезпечують швидку розробку програмного забезпечення з високим рівнем якості. Водночас, Scrum зосереджений на коротких циклах розробки (спринтах) та регулярних зустрічах для оцінки прогресу та планування наступних етапів. Гнучкі методології є особливо корисними для проєктів з високою динамікою змін, коли вимоги можуть швидко змінюватися в процесі розробки. Вони сприяють швидкому реагуванню на зворотний зв'язок від замовників, забезпечують прозорість процесу розробки та ефективно залучають команду до прийняття рішень [1, 2, 15].

Кожна з розглянутих моделей розробки має свої переваги та обмеження, які визначають їхню придатність для конкретних проєктів. Водоспадна модель підходить для проєктів із чіткими та стабільними вимогами, тоді як інкрементна модель забезпечує більшу гнучкість та можливість поступового розширення продукту. Еволюційні моделі, такі як прототипування та спіральна модель, надають можливість ефективно працювати в умовах невизначеності та високих ризиків. Гнучкі методології, такі як XP та Scrum, є оптимальними для динамічних проєктів, де важливе значення мають швидкість адаптації та регулярний зворотний зв'язок із замовником.

Для успішної реалізації проєкту з розробки програмного забезпечення важливо правильно вибрати модель розробки, яка відповідатиме специфіці вимог, рівню ризиків та динаміці змін у проєкті. Використання інструментів машинного навчання для аналізу ризиків та вибору оптимальної моделі дозволяє підвищити точність прогнозування та знизити ймовірність виникнення проблем у процесі розробки. Це робить процес управління розробкою програмного

забезпечення більш надійним та ефективним, забезпечуючи кращу відповідність вимогам замовника та успішне завершення проєктів.

1.3 Аналіз підходів до управління ризиками в сучасних програмних проєктах

У сучасну цифрову епоху розробка програмного забезпечення стикається з багатьма викликами, серед яких найбільш значущими є нестабільність вимог, складність взаємодії різних пристроїв і систем, а також обмеження ресурсів. Ефективне управління ризиками стає ключовим елементом для успішного завершення проєктів з розробки програмного забезпечення, особливо в умовах швидких змін і зростаючої складності [4].

Одним із головних викликів є складність моніторингу ризиків у великих проєктах. Нерідко в таких проєктах спостерігається нестабільний аналіз вимог, неправильний розрахунок витрат або неефективний контроль виконання, що призводить до затримок і перевитрат. У той же час, локальні проєкти виявляються ефективнішими завдяки простішій структурі управління та меншій кількості змін.

Невизначеність також є невід’ємною частиною розробки програмного забезпечення. Попри те, що сучасні гнучкі методології [5], такі як Agile, забезпечують високу адаптивність, вони часто нехтують систематичним управлінням ризиками. Це може призвести до нестабільності навіть у проєктах, які формально використовують сучасні підходи.

Традиційні метрики програмного забезпечення, як-от кількість рядків коду або вартість дефектів, показали свою неефективність у багатьох випадках. Водночас метрики, що відображають складність коду або функціональні точки, стають дедалі важливішими, адже вони краще відображають реальну продуктивність і складність проєктів.

У сфері Інтернету речей виникають додаткові виклики, пов’язані з обмеженнями обчислювальних ресурсів, енергоспоживанням і недостатньою

якістю мережевих послуг. Ці проблеми можуть мати серйозні наслідки для критично важливих систем, таких як розумні міста, де відсутність належного управління ризиками може призвести до катастрофічних наслідків.

Для подолання цих викликів використовуються сучасні рішення. Наприклад, штучний інтелект допомагає автоматизувати процес оцінки ризиків, аналізувати великі обсяги даних і прогнозувати можливі загрози. Алгоритми машинного навчання, такі як нейронні мережі або випадкові ліси, дозволяють не тільки визначати ризики, але й пропонувати ефективні способи їх зниження [6-14].

Гнучкі методології [5], такі як Scrum, дозволяють командам швидко адаптуватися до змін і оперативно вирішувати проблеми. Проте для досягнення стабільності в проєктах важливо доповнювати такі методології системними підходами до управління ризиками.

Ще одним важливим інструментом є мультикритеріальний аналіз, наприклад, метод AHP, який допомагає враховувати різні фактори ризиків і визначати оптимальні стратегії їхнього пом’якшення.

Порівняльний аналіз зазначених вище підходів подано в таблиці 1.1.

Таблиця 1.1 - Порівняльний аналіз підходів

Підхід	Переваги	Недоліки
Традиційні методи	Простота, підходять для невеликих проєктів.	Суб’єктивність, складність масштабування для великих і динамічних проєктів.
Гнучкі методології (Agile)	Адаптивність, швидкість реагування, орієнтованість на клієнта.	Відсутність довгострокового планування ризиків, менш ефективні для великих проєктів.
Методи на основі ШІ	Автоматизація, висока точність прогнозування, аналіз великих даних.	Висока складність впровадження, потреба у якісних даних, висока вартість.
Мультикритеріальні методи	Систематизація ризиків, врахування специфіки проєкту.	Залежність від експертних оцінок, витрати часу.

Таким чином, управління ризиками в проєктах з розробки програмного забезпечення є багатогранним завданням, яке потребує використання різних підходів і технологій. Поєднання сучасних методів, таких як штучний інтелект, із традиційними та гнучкими підходами дозволяє підвищити ефективність розробки та забезпечити стабільність програмних проєктів навіть в умовах високої складності. Це дасть можливість розробникам не лише знижувати ризики, але й досягати більш високих результатів у коротші терміни, зберігаючи високу якість продукту.

1.4 Постановка задачі дослідження

Розробка програмного забезпечення є складним процесом, який супроводжується різними ризиками, що можуть негативно вплинути на строки виконання, бюджет та якість кінцевого продукту. У сучасних умовах розвитку інформаційних технологій проєкти часто стикаються з високим рівнем невизначеності, що особливо актуально для таких сфер, як Інтернет речей, обчислення на периферії та хмарні обчислення. Ці проєкти вимагають високого рівня надійності та безпеки, що робить управління ризиками одним із ключових завдань для успішного виконання програмних проєктів [29].

Традиційні методи управління ризиками у програмному забезпеченні часто мають обмежені можливості через відсутність систематичного підходу до прогнозування та врахування динаміки проєктів. Зазвичай, аналіз ризиків здійснюється на основі експертних оцінок та минулого досвіду, що може бути неефективним в умовах швидких технологічних змін та розвитку нових вимог. Це призводить до потреби в автоматизованих системах, які здатні аналізувати великі обсяги даних та враховувати взаємозв’язки між різними факторами ризику, забезпечуючи точні прогнози та рекомендації для керівництва.

Основною задачею кваліфікаційної роботи є розробка методу та впровадження на його основі системи прогнозування ризиків у проєктах розробки програмного забезпечення, яка використовує сучасні методи

машинного навчання для автоматизації процесу оцінки ризиків. Це дозволить покращити процес прийняття рішень, мінімізувати можливі загрози та забезпечити вибір оптимальної стратегії розробки для конкретних умов проєкту.

Для досягнення цієї мети необхідно вирішити низку задач:

1. Аналіз сучасних підходів до управління ризиками у розробці програмного забезпечення, включаючи методи машинного навчання та алгоритми класифікації, які використовуються для прогнозування ризиків. Це дозволить визначити сильні та слабкі сторони існуючих методів та обґрунтувати вибір найбільш підходящих підходів для автоматизованого аналізу ризиків.

2. Збір даних та формування навчальних та тестових вибірок, що включають інформацію про різні моделі розробки програмного забезпечення та фактори ризику, такі як метрики безпеки, продуктивність, час виконання, затримки, тощо. Для цього буде проведено опитування експертів з розробки програмного забезпечення, що дозволить сформувати базу знань для навчання моделей.

3. Розробка та налаштування моделей машинного навчання для прогнозування ризиків на основі зібраних даних. Це включає використання різних алгоритмів, таких як машина опорних векторів (SVM), метод k-ближніх сусідів (k-NN), штучні нейронні мережі з багатошаровим персептроном (ANN-MLP) та випадковий ліс (Random Forest). Різноманітність методів дозволить провести порівняльний аналіз їхньої ефективності для вибору найбільш точного підходу до прогнозування.

4. Навчання моделей та оцінка їхньої точності з використанням тестових наборів даних. Це дозволить перевірити здатність моделей до точного прогнозування ризиків, а також оцінити їхню продуктивність за допомогою метрик, таких як точність (accuracy), матриця невідповідностей (confusion matrix) та звіт про класифікацію (classification report). Цей етап є ключовим для визначення надійності моделей та їх здатності забезпечувати ефективні рекомендації.

5. Інтеграція моделі прогнозування у систему управління проєктами розробки програмного забезпечення. Це дозволить забезпечити автоматичний аналіз ризиків у реальному часі та надавати рекомендації щодо оптимальної стратегії розробки, знижуючи потенційні ризики на кожному етапі життєвого циклу програмного забезпечення.

6. Проведення експериментальних досліджень та аналіз результатів роботи системи для визначення сильних та слабких сторін запропонованого підходу. Це включатиме порівняння роботи системи з активною та неактивною моделлю прогнозування, що дозволить оцінити вплив використання машинного навчання на зниження ризиків.

7. Розробка рекомендацій для використання системи в умовах реальних проєктів, зокрема у сферах з високою складністю та невизначеністю, таких як IoT та обчислення на периферії. Це дозволить адаптувати запропоновану систему до різних бізнес-вимог та забезпечити її практичну цінність для розробників та керівників програмних проєктів.

Таким чином, постановка задачі дослідження полягає у створенні нових методів та інструментів для прогнозування ризиків у проєктах розробки програмного забезпечення. Використання методів машинного навчання забезпечує точнішу оцінку ризиків, що дозволяє керівникам проєктів приймати більш обґрунтовані рішення, знижувати вплив можливих загроз та підвищувати ефективність розробки. Це сприятиме підвищенню якості програмних продуктів та зменшенню витрат на їхнє створення, забезпечуючи успішне виконання таких проєктів в умовах динамічного розвитку технологій.

Висновки до розділу 1

1. Управління ризиками у розробці програмного забезпечення є критично важливим елементом для забезпечення успішного завершення проєктів. Це включає ідентифікацію, аналіз та оцінку потенційних ризиків, таких як затримки у виконанні, перевищення бюджету, технічні проблеми, відсутність

відповідності вимогам замовника та інші загрози. Традиційні підходи до управління ризиками зазвичай базуються на експертних оцінках і застосовуються вручну, що може призводити до суб'єктивних рішень і недостатньої точності у виявленні ризиків на ранніх етапах проєкту.

2. Для успішної реалізації програмного проєкту важливо правильно вибрати модель розробки, яка відповідатиме специфіці вимог, рівню ризиків та динаміці змін у проєкті. Використання інструментів машинного навчання для аналізу ризиків та вибору оптимальної моделі дозволяє підвищити точність прогнозування та знизити ймовірність виникнення проблем у процесі розробки. Це робить процес управління розробкою програмного забезпечення більш надійним та ефективним, забезпечуючи кращу відповідність вимогам замовника та успішне завершення проєктів.

2 ПРОГНОЗУВАННЯ РИЗИКІВ У ПРОЄКТАХ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Концептуальні основи методу прогнозування ризиків у проєктах з розробки програмного забезпечення за допомогою машинного навчання

Проєкти з розробки програмного забезпечення часто зустрічаються з різноманітними ризиками, які можуть вплинути на терміни виконання, бюджет та якість кінцевого продукту. Враховуючи складність сучасних програмних систем та зростаючий вплив новітніх технологій, таких як машинне навчання, важливо впроваджувати інструменти, які дозволяють передбачати та керувати потенційними ризиками. Метод прогнозування ризиків за допомогою машинного навчання надає можливість здійснювати автоматизований аналіз ризиків та приймати рішення на основі точних прогнозів [30].

На основі запропонованого методу прогнозування ризиків за допомогою машинного навчання можна розробити систему, яка може:

– ідентифікувати ключові ризики, що впливають на виконання проєкту розробки програмного забезпечення;

– прогнозувати ймовірність виникнення ризиків на основі історичних даних та аналітичних моделей;

– надати рекомендації щодо пом'якшення ризиків та підтримки прийняття рішень.

Завданнями цього методу є:

– збір даних з різних джерел, включаючи відповіді експертів, лог-файли процесів та історичні дані минулих проєктів;

– обробка та підготовка даних для створення моделі машинного навчання;

– навчання та оцінка моделей машинного навчання для досягнення високої точності прогнозування;

– інтеграція системи прогнозування з платформою управління проєктами для підтримки управлінських рішень.

Метод прогнозування ризиків базується на концептуальній моделі, яка включає три основні компоненти [30]:

1. Збір та обробка даних.
2. Моделювання та навчання на основі машинного навчання.
3. Прогнозування та підтримка прийняття рішень.

1. Збір та обробка даних.

Цей компонент полягає у зборі якісних даних про ризики з різних джерел, таких як анкети, що заповнюються експертами з розробки програмного забезпечення, історичні дані з попередніх проєктів, дані з журналів процесів та логів систем. Наприклад, анкета може містити інформацію про використані моделі розробки (Agile, Waterfall, Incremental, Evolutionary), характеристики безпеки, пропускну здатність мережі, затримки, минулі помилки, методи розробки та інші атрибути. Обробка даних включає:

– видалення недійсних відповідей;
– нормалізацію даних для усунення масштабних відмінностей;
– розподіл даних на навчальні та тестові набори у співвідношенні 80-20%.

2. Моделювання та навчання на основі машинного навчання.

Основою методу є використання різних алгоритмів машинного навчання для побудови моделей прогнозування ризиків. Найпопулярнішими серед них є:

Машина опорних векторів [29, 32], яка добре працює з високорозмірними даними та використовує лише частину навчальних точок для визначення функцій прийняття рішень, що робить її ефективною у використанні пам'яті.

Метод k-ближчих сусідів [30, 32], який класифікує нові дані на основі найближчих зразків з навчального набору, що дозволяє застосовувати його для класифікації без наявної інформації про розподіл даних .

Штучні нейронні мережі [31, 32], зокрема багатошаровий персептрон, що здатний навчатися складних взаємозв'язків між вхідними ознаками та результатами.

Випадковий ліс [32], який складається з ансамблю дерев рішень, що забезпечує високу точність та зменшує ризик перенавчання.

Навчання моделі проводиться на основі історичних даних, що дозволяє створити вектор ознак, який потім використовується для прогнозування ризиків на нових даних. Під час навчання функція активації RELU у прихованих шарах нейронних мереж сприяє швидшому навчанню за рахунок розрідженості нейронної мережі.

3. Прогнозування та підтримка прийняття рішень.

Після завершення навчання моделі тестовий набір даних передається через натреновану модель, що дозволяє отримати передбачені мітки ризиків для кожного нового проєкту. Цей етап включає:

– прогнозування відсотка ризику для кожної моделі розробки на основі ймовірностей, отриманих від класифікаторів;

– порівняння прогнозованих ризиків з фактичними, що дозволяє оцінити точність моделі;

– генерація рекомендацій для зниження ризиків на основі аналізу факторів, що впливають на ризик.

Результати прогнозування дозволяють менеджерам проєктів та іншим зацікавленим сторонам вибирати оптимальну модель розробки програмного забезпечення з найменшим рівнем ризику, що знижує ймовірність невдач та підвищує ефективність проєкту.

4. Інтеграція з IoT та обчисленнями на периферії.

Запропонований метод може бути інтегрований з системами на основі IoT та периферійними обчисленнями. У такому середовищі система прогнозування ризиків забезпечує постійний моніторинг роботи та прогнозування ризиків у реальному часі. Це дозволяє гнучко адаптувати стратегію управління проєктом та оперативно реагувати на потенційні загрози. Наприклад, під час розробки додатків для розумних міст, система може прогнозувати ризики затримок або втрати даних через несприятливі умови мережі та своєчасно адаптувати процес розробки.

Отже, метод прогнозування ризиків у проєктах з розробки програмного забезпечення на основі машинного навчання є потужним інструментом для проєктних менеджерів та розробників. Він дозволяє ефективно визначати ризики на ранніх етапах розробки, що дає можливість краще керувати ресурсами та приймати обґрунтовані рішення. Результати прогнозування допомагають вибрати оптимальні моделі розробки, що мінімізує можливі проблеми та сприяє успішному завершенню проєктів.

2.2 Збір даних та формування навчальної і тестової вибірок

На сьогоднішній день використання ефективної моделі розробки програмного забезпечення є надзвичайно важливим для відповідності очікуванням споживачів. Незважаючи на існування багатьох моделей розробки програмного забезпечення, розробницькі команди часто обирають конкретні моделі відповідно до встановлених вимог. Для збору даних було проведено опитування серед експертів з розробки програмного забезпечення з різних організацій, яких попросили заповнити анкету, представлену нижче. Ця анкета включає кілька атрибутів, метрик та індикаторів, що стосуються різних аспектів програмного забезпечення та обладнання. Експерти заповнювали анкету, використовуючи свій досвід і знання в галузі. Зібраний набір даних ефективно допоміг ідентифікувати потенційні елементи ризику в різних моделях розробки програмного забезпечення.

Анкета містить деякі моделі розробки програмного забезпечення, які використовуються власниками продуктів та менеджерами під час розробки, зокрема, гнучкі методології, водоспадна модель, інкрементна модель та еволюційна модель. Ці моделі були визначені шляхом обговорень з експертами в галузі та аналізу наукових статей, присвячених моделям розробки програмного забезпечення. Щодо визначених моделей розробки програмного забезпечення, було визначено шість різних метрик: мережа, безпека, програмне забезпечення, машинне навчання, інтерфейс прикладного програмування (API) та інтернет

речей. Крім того, вони були деталізовані до різних атрибутів для більш детального розуміння.

Атрибути включають такі характеристики, як пропускна здатність, затримка та діапазон дії на краю (edge-exposure range) щодо мережі, а також фактори автентифікації та вразливості стосовно безпеки. Далі, у сфері програмного забезпечення враховувалися такі атрибути, як минулі помилки, методи реалізації та графік поставок. У машинному навчанні (ML) до атрибутів відносилися рівень помилок, точність та ефективність [26]. Нарешті, у випадку інтерфейсу прикладного програмування враховувались його ефективність та доступність/навантаження, а для інтернету речей – вартість та обслуговування.

Анкета, представлена у таблиці 2.1, була надіслана майже 150 експертам з розробки програмного забезпечення, проте для подальшого аналізу були враховані 112 коректно заповнених відповідей. Відгуки були отримані від різних експертів з розробки програмного забезпечення або керівників команд та накопичені у вигляді документів Excel. Використовуючи ці Excel-документи, було створено систему прогнозування. Завдяки цій системі розробник може вводити необхідні параметри, отримані через форму вимог від замовника. Відповідно, система прогнозування рекомендуватиме найбільш підходящу модель розробки програмного забезпечення [15].

Шість метрик, перелічених у цій анкеті, були визначені після аналізу кількох відповідних робіт у сфері розробки програмного забезпечення. Як зазначалося вище, для цих метрик були визначені відповідні атрибути для оцінки їхньої значущості. Опис атрибутів для кожної з метрик подано для подальшого розуміння.

По-перше, у метриці «мережа» пропускна здатність стосується каналу, через який передаються дані; затримка показує тривалість часу, необхідного для обробки запиту та відповіді в межах мережі; діапазон дії на краю (edge-exposure range) визначає фокус на граничних обчисленнях стосовно конкретної моделі програмного забезпечення.

Таблиця 2.1 – Відповіді на анкету, отримані від експертів з розробки програмного забезпечення – зразок рейтингу від 0 до 9 (найнижча надійність починається з 0, а найвища надійність - 9).

Модель	Мережа			Безпека		ПЗ			МН			API		IoT		Загальний ризик
	Пропускна здатність	Затримка	Діапазон меж	Фактори автентифікації	Вразливість	Минулі помилки/Досвід	Метод	Графік доставки	Рівень помилок	Точність	Ефективність	Ефективність інтерфейсу	Доступність/Трафік	Вартість	Обслуговування	Перевага 0/1; 0- не ризиковано 1-ризиковано
Гнучка	5	5	5	7	7	5	5	8	7	7	7	8	7	5	5	0
Водоспадна	5	5	5	7	7	5	5	8	7	7	7	8	7	6	6	0
Інкрементна	5	5	5	7	7	5	5	7	7	7	7	8	7	6	6	1
Еволюційна	5	5	5	7	7	5	5	5	7	7	7	6	7	5	5	1

По-друге, у метриці «безпека» фактори автентифікації визначають елементи безпеки, які є важливими для використання системи, зокрема, паролі та біометричні дані; вразливість відображає рівень уразливості системи до несанкціонованого доступу.

По-третє, у метриці «програмне забезпечення» минулі помилки або досвід означають серйозні інциденти та їхній вплив під час минулих проєктів; метод ілюструє, наскільки важливо правильно вибрати метод розробки програмного забезпечення; графік постачання відображає значення ефективного управління часом у постачанні продукту.

Далі, у метриці «машинне навчання» очікувана частота помилок (EER) означає кількість неправильних передбачень, зроблених класифікатором, шляхом порівняння передбаченої мітки з фактичною інформацією про мітку (рівень помилок), а ефективність показує співвідношення виконаних продуктивних завдань протягом заданого часу [16].

Наступна метрика – інтерфейс прикладного програмування, де ефективність інтерфейсу означає простоту структурування необхідної платформи; доступність визначає надійність системи, коли вона знаходиться в робочому стані та готова до використання.

Остання метрика – інтернет речей, де вартість визначає обсяг коштів, які потрібно виділити на закупівлю обладнання та його довговічність, а обслуговування відображає час та фінанси, необхідні для підтримки роботи інструментів.

Крім того, експертам було запропоновано оцінити ризики, пов’язані з кожною моделлю розробки програмного забезпечення, враховуючи ці метрики. Відомо, що під час розробки програмного забезпечення існує безліч ризиків, і управління ризиками відіграє важливу роль у цьому процесі. З іншого боку, більшість сучасних досліджень з аналізу ризиків програмного забезпечення зосереджуються на виявленні зв’язку між ризиковими аспектами та результатами проєктів. Крім того, невдачі у програмних проєктах найчастіше є наслідком недостатнього та неефективного управління ризиками. Для того, щоб досягти належного та цінного управління ризиками, необхідно проводити планування ризиків на основі причинно-наслідкових зв’язків ризиків, що дозволяє отримувати додаткову інформацію для прийняття управлінських рішень [27].

Для ефективного використання набору даних було здійснено поділ на 80-20%, де 80% використовується для навчання, а 20% – для тестування. Причина вибору 80% для навчання полягає в тому, щоб надати моделі можливість навчитися на великому обсязі даних та здійснювати точне прогнозування. Для навчального набору надаються дані (або ознаки) разом з мітками, тоді як у випадку з тестовим набором надаються лише дані без міток.

На початковому етапі навчання в модель або класифікатори надходить максимальна кількість даних, що дозволяє їм вивчати ознаки та конвертувати їх у представлення ознак. Це представлення ознак або вектор ознак генерується для подальшої обробки. Під час навчання, коли надходять нові дані, система за

допомогою вектора ознак намагається передбачити та класифікувати ці дані, а згодом за допомогою мітки перевіряється, чи система правильно їх класифікувала. Дані, які були правильно класифіковані, використовуються для обчислення точності навчання. На основі цієї точності навчання модель перевіряється та проходить валідацію.

Після цього до натренованої моделі передаються решта 20% тестових даних без міток, щоб перевірити, чи здатна модель коректно передбачити нові дані. Таким чином, тестові дані обробляються натренованою моделлю, і для кожної точки даних генеруються передбачені мітки. Порівнюється фактична мітка (ground truth) з передбаченою міткою, на основі чого обчислюється точність тестування. Крім того, відсоток ризику розраховується шляхом активації та деактивації натренованої моделі.

2.3 Система рекомендацій для розробки програмного забезпечення

Система рекомендацій для розробки програмного забезпечення представлена у вигляді діаграми потоку даних на рисунку 2.1.

Ця діаграма ілюструє систему рекомендацій для розробки програмного забезпечення та взаємодію між ключовими зацікавленими сторонами, що беруть участь у процесі управління програмними проєктами. У системі виділено чотири основні ролі: спонсор продукту (Product Sponsor), менеджер продукту (Product Manager), власник продукту (Product Owner) та власник архітектури (Architecture Owner). Давайте детально розглянемо, як відбувається взаємодія між цими ролями та системою рекомендацій.

Опис компонентів діаграми:

1. Система рекомендацій (Recommendation System).

Цей центральний компонент є ядром системи, яке здійснює аналіз даних та надає рекомендації щодо вибору відповідної моделі розробки програмного забезпечення або підходу.

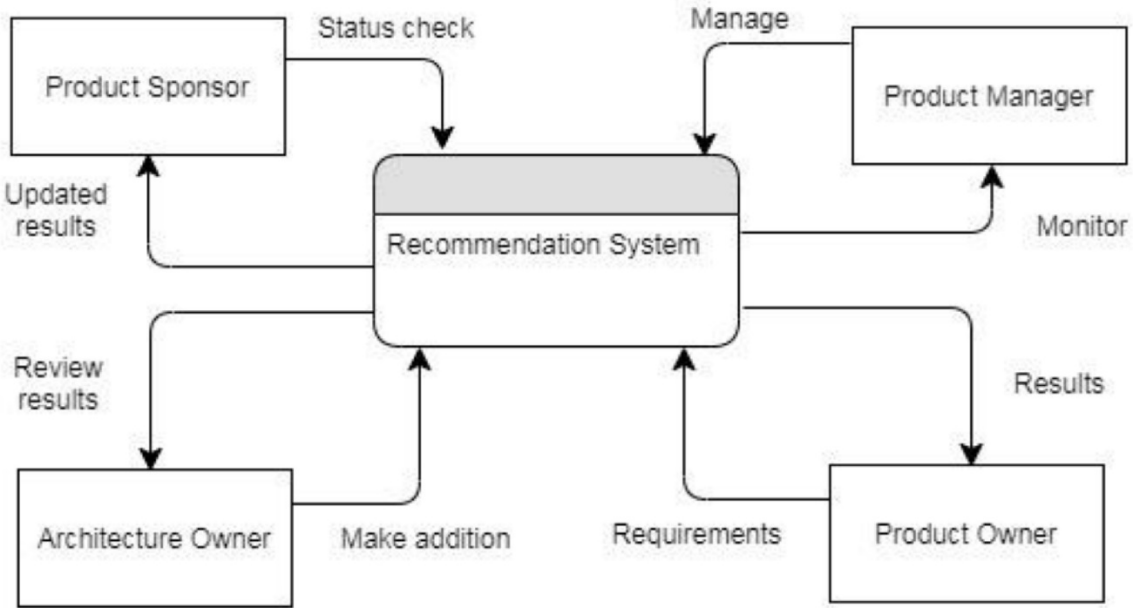


Рисунок 2.1 – Діаграма потоку даних

Система опрацьовує вхідні дані та вимоги, отримані від власника продукту, і генерує результати, які потім надаються менеджеру продукту, спонсору продукту та власнику архітектури.

2. Спонсор продукту (Product Sponsor).

Взаємодія з системою рекомендацій включає:

2.1. Status Check: спонсор продукту може перевіряти статус виконання проєкту або системи рекомендацій. Це дозволяє отримувати інформацію про поточний стан розробки.

2.2. Updated Results: система надає спонсору продукту оновлені результати, що можуть включати аналіз ризиків, рекомендації щодо вибору моделей розробки та інші ключові інсайти.

3. Менеджер продукту (Product Manager).

Основні функції та взаємодії з системою рекомендацій:

3.1. Manage: менеджер продукту відповідає за управління процесом розробки та може коригувати параметри роботи системи рекомендацій.

3.2. Monitor: менеджер має можливість відслідковувати результати та ефективність роботи системи, що дозволяє оперативно реагувати на зміни та оновлювати стратегію розробки.

4. Власник продукту (Product Owner).

Його роль зосереджена на забезпеченні вимог замовника:

4.1. Requirements: власник продукту передає в систему рекомендацій вимоги до продукту, що є основою для формування рекомендацій системою.

4.2. Results: після обробки вимог система надає результати, що дозволяють власнику продукту отримати уявлення про ефективний підхід до розробки або можливі ризики.

5. Власник архітектури (Architecture Owner).

Ця роль відповідає за технічний бік проєкту:

5.1. Make Addition: власник архітектури може вносити зміни в систему рекомендацій, додаючи нові технічні вимоги або коригуючи параметри роботи.

5.2. Review Results: після оновлення параметрів система рекомендацій надає результати, які власник архітектури може переглядати та аналізувати.

Основні взаємодії та процеси:

1. Система рекомендацій слугує зв'язуючим елементом між усіма зацікавленими сторонами. Кожна роль має можливість взаємодіяти з системою, вносити зміни або отримувати результати для прийняття рішень.

2. Процес починається з отримання вимог від власника продукту, після чого система обробляє ці дані та надає рекомендації щодо оптимальної моделі розробки.

3. Менеджер продукту відслідковує та коригує процес розробки, спираючись на дані, отримані від системи, а спонсор продукту отримує інформацію про поточний стан проєкту.

4. Власник архітектури може коригувати технічні вимоги та переглядати результати, щоб переконатися, що система рекомендацій пропонує оптимальні рішення.

Система рекомендацій для розробки програмного забезпечення є важливим інструментом, який допомагає управляти ризиками у проєктах. Її основна функція полягає у забезпеченні взаємодії між ключовими зацікавленими сторонами та аналізі вхідних даних для вибору оптимальної моделі розробки. Ця

система дозволяє отримувати вимоги від власника продукту, обробляти їх та надавати рекомендації для менеджера продукту, спонсора продукту та власника архітектури. Завдяки цьому забезпечується як технічна, так і бізнесова узгодженість у межах проєкту.

Система тісно пов’язана з управлінням ризиками на всіх етапах проєкту. Наприклад, вона автоматично аналізує вимоги для виявлення потенційних проблем, таких як неоднозначність чи неповнота. Це допомагає мінімізувати ризики, пов’язані з нестабільністю вимог, які є однією з головних причин невдачі проєктів.

Рекомендації системи дозволяють менеджеру продукту контролювати процеси, дотримуватися графіку та уникати перевитрати ресурсів. Вона також може виявляти затримки або відхилення від плану, пропонуючи рішення для коригування. Власник архітектури використовує систему для вибору оптимальних технологій і компонентів, що дозволяє уникнути ризиків, пов’язаних із технічними обмеженнями чи невідповідністю вибраних рішень.

Спонсор продукту отримує від системи прозору інформацію про стан проєкту, що дозволяє своєчасно виявляти ризики та приймати обґрунтовані рішення. Завдяки цьому можна уникнути непрозорості в управлінні проєктом і забезпечити залучення всіх зацікавлених сторін.

Система також сприяє узгодженню бізнесових і технічних цілей. Вона інтегрує бізнесові вимоги замовника з технічними обмеженнями, створюючи баланс між ними. Це допомагає запобігати конфліктам, які могли б вплинути на якість чи строки виконання проєкту.

Отже, система рекомендацій є важливим компонентом управління ризиками у розробці програмного забезпечення. Вона забезпечує своєчасне виявлення та усунення загроз, підтримує прозорість процесів і сприяє досягненню узгодженості між технічними та бізнесовими цілями, що значно підвищує ймовірність успіху програмного проєкту.

Висновки до розділу 2

1. Метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання є ключовим інструментом для забезпечення успіху сучасних програмних проєктів. Він дозволяє виявляти та передбачати ризики, які можуть вплинути на хід розробки, терміни виконання та якість кінцевого продукту. Використовуючи історичні дані та потужність машинного навчання, цей метод забезпечує точний аналіз та автоматизацію процесу управління ризиками.

2. Збір даних та формування навчальної і тестової вибірок є критичним етапом у побудові системи прогнозування ризиків у проєктах програмного забезпечення. Опитування серед експертів дозволило зібрати якісні дані, що містять інформацію про метрики та атрибути різних моделей розробки, таких як Agile, Waterfall, інкрементна та еволюційна моделі. Зібраний набір даних був поділений на навчальну та тестову вибірки у співвідношенні 80-20, що забезпечує ефективне навчання моделей машинного навчання та їх перевірку на нових даних.

3. Система рекомендацій для розробки програмного забезпечення є важливим інструментом, який допомагає управляти ризиками у проєктах. Її основна функція полягає у забезпеченні взаємодії між ключовими зацікавленими сторонами та аналізі вхідних даних для вибору оптимальної моделі розробки. Ця система дозволяє отримувати вимоги від власника продукту, обробляти їх та надавати рекомендації для менеджера продукту, спонсора продукту та власника архітектури. Завдяки цьому забезпечується як технічна, так і бізнесова узгодженість у межах проєкту.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНИХ РІШЕНЬ ПРОГНОЗУВАННЯ РИЗИКІВ У ПРОЄКТАХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

3.1 Методика проведення експериментальних досліджень

Останнім часом машинне навчання стає важливою дослідницькою галуззю. Набір даних, сформований на основі коректних відповідей, отриманих від експертів з розробки програмного забезпечення, можна використовувати для навчання класифікаторів машинного навчання, зокрема, таких як машина опорних векторів (SVM), метод k-ближчих сусідів (k-NN), штучна нейронна мережа (ANN) та випадковий ліс (Random Forest).

Випадковий ліс є мета-оцінювачем, який будує низку дерев рішень на основі підвибірок даних та здійснює усереднення результатів для підвищення точності прогнозування та уникнення перенавчання. Крім того, випадковий ліс застосовується як для регресії, так і для класифікації. Він складається з послідовності дерев рішень, кожне з яких діє як слабкий класифікатор, що зазвичай має незначну точність прогнозування. Проте у сукупності ці дерева забезпечують більш точний прогноз. Використовуючи випадкові ліси, кожне дерево в ансамблі генерується з вибірки, створеної з поверненням із навчального набору. Випадкові ліси також зменшують дисперсію за рахунок комбінування різних дерев, інколи при цьому злегка збільшуючи похибку. На практиці зменшення дисперсії є значним, що дозволяє досягти загалом більш ефективної моделі [17].

Машини опорних векторів (SVM) – це сукупність методів навчання з учителем, які використовуються для класифікації, регресії та виявлення аномалій. Основні переваги SVM полягають у тому, що вони ефективні в умовах з високою розмірністю і особливо корисні, коли кількість вимірів перевищує кількість зразків [18]. Крім того, SVM використовує частину навчальних точок для функцій прийняття рішень, що робить їх економними щодо пам'яті. Вони

також є гнучкими, оскільки для функцій прийняття рішень можна використовувати різні ядра.

Штучні нейронні мережі (ANN) є методологією вирішення складних завдань, що базується на конекціоністській моделі. Вони складаються з групи організованих нейронів, ваги яких коригуються до отримання необхідного результату. Штучні нейронні мережі запозичені з біологічних нейронних мереж, виявлених у мозку ссавців, щоб імітувати здатність таких біологічних структур до обробки інформації.

Далі, багатошаровий персептрон (MLP) – це класифікатор з навчанням з учителем, який вивчає функцію, наприклад, f(.): R^m -> R^o, навчаючись на наборі даних, де m – кількість вимірів вхідних даних, а o – кількість вимірів вихідних даних.

Однак MLP відрізняється від логістичної регресії тим, що може мати приховані шари між вхідним та вихідним шаром [19]. З одного боку, вхідний шар містить нейрони, що відображають вхідні ознаки. Кожен нейрон у прихованому шарі перетворює значення з попереднього шару, використовуючи зважену лінійну суму разом із нелінійною функцією активації, такою як гіперболічний тангенс. З іншого боку, вихідний шар приймає значення з останнього прихованого шару та перетворює їх на вихідні значення.

Класифікатор k-ближчих сусідів (k-NN) є інерційним класифікатором з навчанням з учителем, який використовувався в багатьох сферах, таких як статистика, добування даних та розпізнавання образів. Класифікація даних здійснюється на основі найближчих навчальних зразків у просторі ознак [20]. Мета полягає в класифікації невідомих вхідних зразків на основі визначеної кількості їхніх найближчих сусідів. Крім того, k-NN ефективний, коли інформація про розподіл даних недоступна заздалегідь. Класифікація на основі сусідів є видом навчання на прикладах і не намагається побудувати типову внутрішню модель, а просто зберігає зразки навчальних даних.

Після навчання з використанням мови програмування Python можна прогнозувати відповідну модель розробки програмного забезпечення залежно

від вимог. Підготовка даних та реалізація алгоритмів машинного навчання виконувались за допомогою пакету Python scikit-learn.

Запропонована архітектура представлена на рисунку 3.1.

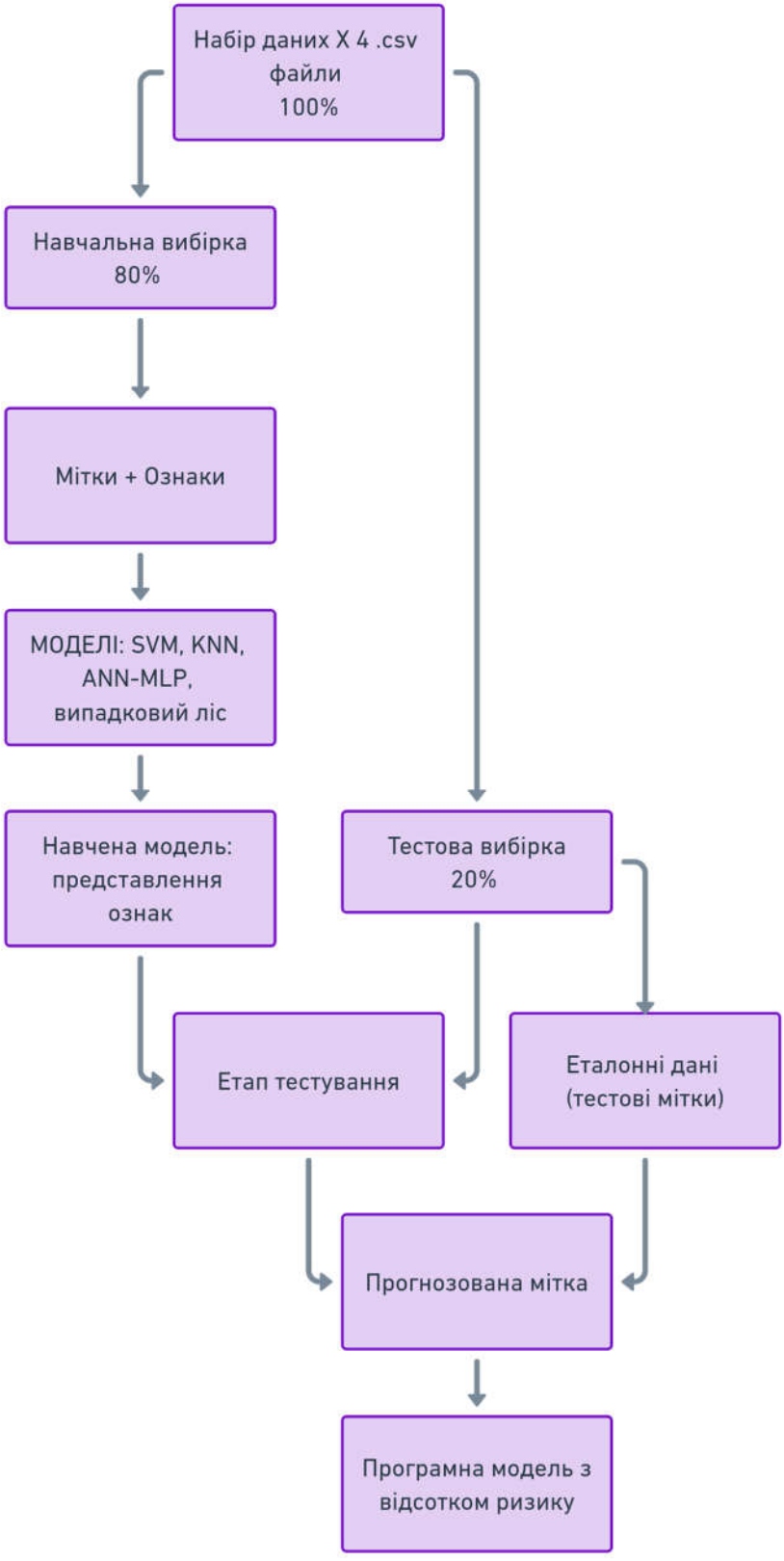


Рисунок 3.1 – Архітектура системи

Робота системи передбачає введення вихідних даних у співвідношенні 80-20, де 80% даних використовуються як навчальний набір. Класифікатори, такі як SVM, k-NN, RF та ANN-MLP, навчаються на навчальному наборі після того, як до моделі передаються навчальні дані. Зазначені класифікатори навчаються на навчальному наборі, і на основі отриманих знань генеруються представлення ознак у вигляді векторів ознак. Ці представлення ознак використовуються для передбачення точності навчання моделі за допомогою класових міток з навчального набору.

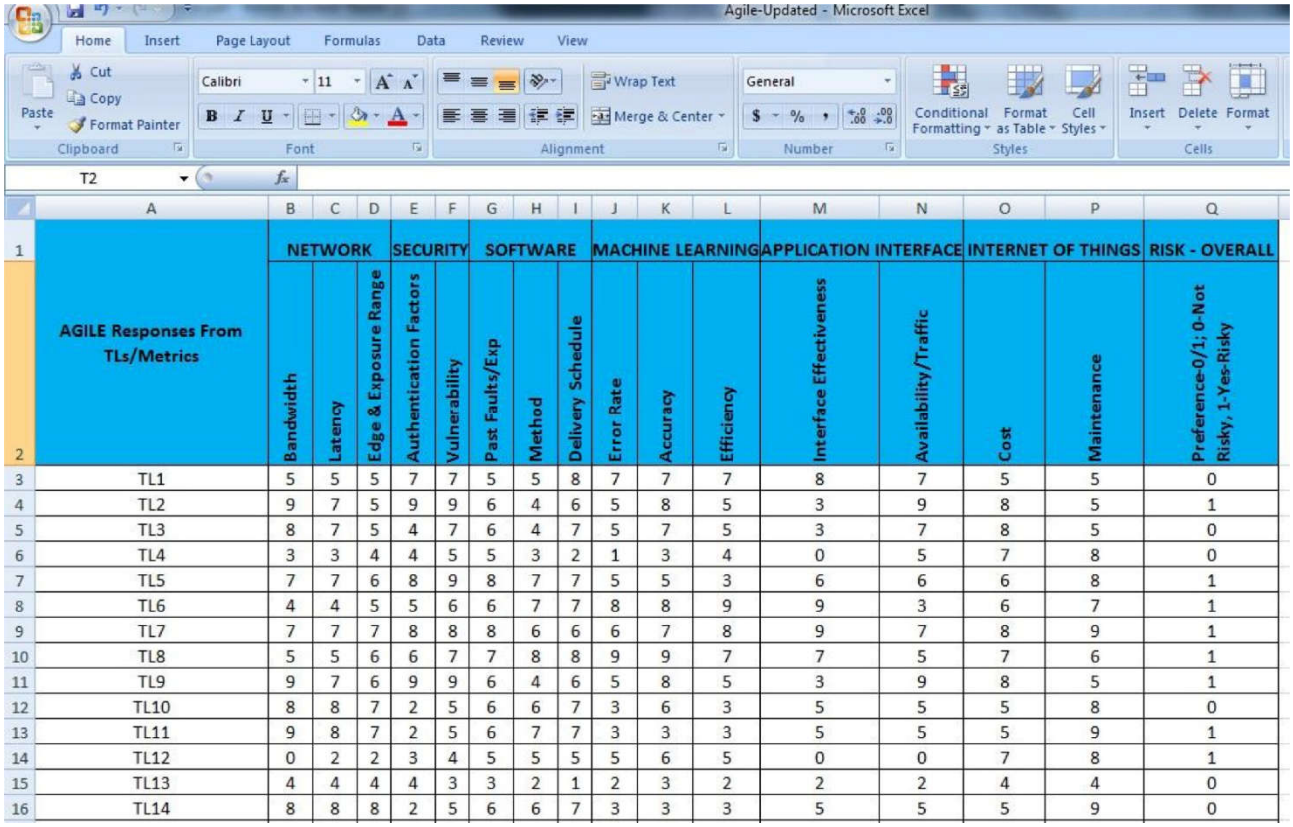
Далі тестовий набір даних передається до натренованої моделі (або класифікатора) для виконання завдання класифікації з учителем (оскільки класифікація з мітками називається класифікацією з учителем), і отримані через цей процес мітки є фактичними мітками. Для додаткової інформації, у модулі передбачених міток також доступні відсотки ризику для інших моделей [28]. Запропонована система надає рекомендації щодо зниження оціненого ризику процесу. Ризики оцінюються шляхом аналізу дерев рішень, створених на основі логів процесів. Крім того, система прогнозування значно зменшує помилки, пов’язані з процесом, та їхню суворість.

3.2 Реалізація класифікаторів машинного навчання

У цьому дослідженні відповіді, отримані від експертів з розробки програмного забезпечення, обробляються за допомогою запропонованої архітектури з метою рекомендації ефективної моделі розробки програмного забезпечення з прогнозуванням ризику для використання під час розробки програмного забезпечення. Оцінка ризику має ключове значення в галузях з підвищеними вимогами до безпеки. Водночас, вона стикається з низкою загальних проблем, частково пов'язаних з технологічними інноваціями та новими вимогами.

Початковий набір даних складався з близько 150 відповідей, але містив деякі недійсні відповіді, тому його було скориговано до 112 валідних відповідей. Використовуючи ці 112 відповідей, їх розподілили як окремі CSV-файли відповідно до чотирьох різних моделей розробки програмного забезпечення. Зокрема, 112 відповідей було додано до 4 різних файлів, таких як agile.csv, до якого включено відповіді, що стосуються цієї моделі, та waterfall.csv, до якого було додано відповіді для цієї моделі. Аналогічно було створено файли incremental.csv та evolutionary.csv.

Нижче на рисунку 3.2 показано файл agile.csv як приклад.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		NETWORK			SECURITY		SOFTWARE			MACHINE LEARNING			APPLICATION INTERFACE		INTERNET OF THINGS		RISK - OVERALL
2	AGILE Responses From TLs/Metrics	Bandwidth	Latency	Edge & Exposure Range	Authentication Factors	Vulnerability	Past Faults/Exp	Method	Delivery Schedule	Error Rate	Accuracy	Efficiency	Interface Effectiveness	Availability/Traffic	Cost	Maintenance	Preference-0/1; 0-Not Risky, 1-Yes-Risky
3	TL1	5	5	5	7	7	5	5	8	7	7	7	8	7	5	5	0
4	TL2	9	7	5	9	9	6	4	6	5	8	5	3	9	8	5	1
5	TL3	8	7	5	4	7	6	4	7	5	7	5	3	7	8	5	0
6	TL4	3	3	4	4	5	5	3	2	1	3	4	0	5	7	8	0
7	TL5	7	7	6	8	9	8	7	7	5	5	3	6	6	6	8	1
8	TL6	4	4	5	5	6	6	7	7	8	8	9	9	3	6	7	1
9	TL7	7	7	7	8	8	8	6	6	6	7	8	9	7	8	9	1
10	TL8	5	5	6	6	7	7	8	8	9	9	7	7	5	7	6	1
11	TL9	9	7	6	9	9	6	4	6	5	8	5	3	9	8	5	1
12	TL10	8	8	7	2	5	6	6	7	3	6	3	5	5	5	8	0
13	TL11	9	8	7	2	5	6	7	7	3	3	3	5	5	5	9	1
14	TL12	0	2	2	3	4	5	5	5	5	6	5	0	0	7	8	1
15	TL13	4	4	4	4	3	3	2	1	2	3	2	2	2	4	4	0
16	TL14	8	8	8	2	5	6	6	7	3	3	3	5	5	5	9	0

Рисунок 3.2 – Приклад набору даних з відповідями на анкету для моделі гнучкої розробки програмного забезпечення

На основі цих файлів був реалізований підхід навчання з учителем із застосуванням класифікаторів машинного навчання, таких як машина опорних

векторів (SVM), метод k-ближчих сусідів (k-NN), штучна нейронна мережа з багатошаровим персептроном (ANN-MLP) та випадковий ліс (Random Forest).

Використовувана функція активації – «RELU» з одним вхідним, трьома прихованими та двома вихідними шарами [23]. Причина вибору RELU для прихованих шарів полягає в тому, що в кожний момент активуються лише деякі нейрони, що робить мережу розрідженою, ефективною та легкою для обчислення. Іншими словами, RELU навчається швидше, ніж інші функції активації, такі як Sigmoid та Tanh [24].

Використання бібліотеки scikit-learn у Python для розв'язання задачі прогнозування ризиків та вибору моделі розробки програмного забезпечення має кілька важливих переваг. Розглянемо ключові аргументи на користь використання цієї бібліотеки:

1. Зручність використання та простота інтеграції.

1.1. scikit-learn є однією з найпопулярніших та найзручніших бібліотек для машинного навчання в Python. Вона забезпечує простий та зрозумілий API, що дозволяє швидко створювати моделі, тренувати їх та використовувати для прогнозування.

1.2. Завдяки своїй зручній структурі та великій кількості документованих прикладів, scikit-learn підходить для швидкого прототипування та розробки машинного навчання, що дозволяє скоротити час на розробку та зосередитися на вирішенні конкретної задачі.

2. Широкий вибір алгоритмів.

2.1. scikit-learn пропонує широкий спектр алгоритмів для класифікації, регресії, кластеризації та зниження розмірності. Це дозволяє легко порівнювати різні моделі, такі як машина опорних векторів (SVM), метод k-ближчих сусідів (k-NN), нейронні мережі та випадковий ліс (Random Forest), обираючи найкращий підхід для конкретної задачі.

2.2. Випадковий ліс (Random Forest), що реалізований у scikit-learn, є ефективним і стійким до перенавчання алгоритмом, який підходить для задач, де

важливо обробляти велику кількість ознак та даних, що є релевантним у задачі прогнозування моделей розробки програмного забезпечення.

3. Інструменти для підготовки даних та їх оцінки.

3.1. Однією з сильних сторін scikit-learn є його можливості для підготовки та обробки даних. Вона включає функції для нормалізації, стандартизації, видалення пропусків у даних, розбиття на навчальні та тестові набори.

3.2. Для завдань класифікації важливо мати можливість швидко оцінити точність та якість моделі. scikit-learn надає різні інструменти для оцінки моделей, такі як accuracy_score, classification_report, confusion_matrix та крос-валідація. Це дозволяє аналізувати продуктивність моделі та знаходити способи її поліпшення.

4. Ефективність та оптимізація обчислень.

4.1. scikit-learn оптимізовано для роботи з великими наборами даних, що дозволяє обробляти значні обсяги інформації без значного зниження продуктивності. Алгоритм випадкового лісу реалізований у цій бібліотеці з урахуванням ефективності обчислень, що забезпечує швидке навчання та прогнозування.

4.2. Бібліотека використовує компіляцію Cython для швидкого виконання коду, що робить її придатною для завдань, де потрібно обробляти великі набори даних, як у нашій задачі прогнозування моделей розробки програмного забезпечення.

5. Гнучкість та розширюваність.

5.1. scikit-learn дозволяє користувачам легко налаштовувати параметри алгоритмів та здійснювати налаштування гіперпараметрів, такі як кількість дерев у випадковому лісі (n_estimators), максимальна глибина дерева та інші. Це дозволяє оптимізувати модель під специфічні вимоги проєкту.

5.2. Крім того, вона підтримує інтеграцію з іншими бібліотеками, такими як pandas для обробки даних та matplotlib/seaborn для візуалізації результатів, що робить її гнучким інструментом для аналізу та презентації даних.

6. Підтримка спільноти та велика база знань.

6.1. scikit-learn має величезну спільноту користувачів та розробників, що постійно підтримує та вдосконалює бібліотеку. Це забезпечує велику кількість навчальних матеріалів, форумів, документації та прикладів, що допомагають швидко розібратися з будь-якими питаннями, які можуть виникнути під час реалізації.

6.2. Завдяки активній спільноті, scikit-learn постійно оновлюється та вдосконалюється, включаючи нові алгоритми та покращення, що робить її актуальною для використання в сучасних проєктах.

Отже, бібліотека scikit-learn у Python є оптимальним вибором для реалізації класифікатора на основі випадкового лісу для задачі прогнозування ризиків у проєктах програмного забезпечення. Її простота використання, ефективність обчислень, широкий набір алгоритмів та інструментів для підготовки та оцінки даних роблять її відмінним інструментом для створення надійних та точних моделей. Це дозволяє швидко розробити прототип, оптимізувати модель та застосовувати її у реальних умовах для підтримки управлінських рішень у сфері розробки програмного забезпечення.

Розглянемо покроково реалізацію класифікатора на основі алгоритму «Випадковий ліс»:

1. Імпорт необхідних бібліотек.

```
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Завантаження та підготовка даних.

Припустимо, що у вас є дані у форматі CSV, які містять метрики для різних моделей розробки програмного забезпечення (наприклад, agile.csv, waterfall.csv тощо). Наприклад, файл містить стовпці з різними атрибутами, такими як

пропускна здатність, затримка, вразливість, точність, ефективність тощо, а також стовпець Model, що визначає цільову змінну (модель розробки).

```
# Завантаження даних з CSV-файлу
data = pd.read_csv('software_models_data.csv')  # Замініть на
ваш файл CSV

# Перевірка перших кількох рядків
print(data.head())

# Перевірка на відсутні значення та попередня обробка
data = data.dropna()  # Видалення рядків з відсутніми
значеннями

# Розділення на ознаки (X) та цільову змінну (y)
X = data.drop('Model', axis=1)  # Ознаки (метрики)
y = data['Model']  # Цільова змінна (модель розробки)
```

3. Розділення на навчальний та тестовий набори.

Для навчання та перевірки точності моделі розділимо дані на навчальний та тестовий набори у співвідношенні 80-20.

```
# Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(f'Розмір навчального набору: {X_train.shape}')
print(f'Розмір тестового набору: {X_test.shape}')
```

4. Навчання моделі Random Forest.

Налаштуємо та натренуємо класифікатор на основі випадкового лісу.

```
# Створення та навчання моделі Random Forest
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_classifier.fit(X_train, y_train)

# Прогнозування на тестовому наборі
y_pred = rf_classifier.predict(X_test)
```

5. Оцінка точності моделі.

Оцінимо точність класифікатора за допомогою метрик, таких як точність (accuracy), звіт про класифікацію (classification report) та матриця невідповідностей (confusion matrix).

```
# Обчислення точності моделі
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Точність моделі Random Forest: {accuracy:.2f}')

# Звіт про класифікацію
print('Звіт про класифікацію:')
print(classification_report(y_test, y_pred))

# Матриця невідповідностей
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Передбачені значення')
plt.ylabel('Фактичні значення')
plt.title('Матриця невідповідностей')
plt.show()
```

6. Прогнозування нових даних.

Після навчання класифікатора ми можемо використовувати його для прогнозування моделі розробки програмного забезпечення на основі нових даних.

```
# Приклад нових даних для прогнозування
new_data = pd.DataFrame({
    'Bandwidth': [5],
    'Latency': [5],
    'Edge_Exposure_Range': [5],
    'Authentication_Factors': [7],
    'Vulnerability': [7],
    'Past_Faults_Exp': [5],
    'Method': [7],
    'Delivery_Schedule': [5],
    'Error_Rate': [7],
    'Accuracy': [7],
    'Efficiency': [7],
    'Interface_Effectiveness': [8],
    'Availability_Traffic': [7],
    'Cost': [5],
    'Maintenance': [5]
})

# Прогнозування на нових даних
predicted_model = rf_classifier.predict(new_data)
print(f'Прогнозована модель розробки програмного забезпечення:
{predicted_model[0]}')
```

7. Збереження натренованої моделі.

Якщо потрібно зберегти натреновану модель для використання в майбутньому, можна скористатися бібліотекою joblib.

```
import joblib
```

```
# Збереження моделі
joblib.dump(rf_classifier, 'random_forest_model.pkl')

# Завантаження моделі
loaded_model = joblib.load('random_forest_model.pkl')
```

Даний код демонструє реалізацію класифікатора на основі алгоритму випадкового лісу (Random Forest) для прогнозування відповідної моделі розробки програмного забезпечення. Використовуючи набір даних, що містить різні метрики та атрибути, цей підхід дозволяє визначати оптимальну модель розробки на основі вимог замовника. Завдяки високій точності та здатності працювати з великими наборами даних, випадковий ліс є відмінним інструментом для завдань класифікації у сфері управління програмними проєктами.

В додатку А представлено реалізацію на основі машини опорних векторів (SVM), методу k-ближчих сусідів (k-NN), штучної нейронної мережі з багатошаровим персептроном (ANN-MLP).

3.3 Результати експериментальних досліджень

Ймовірнісні оцінки, отримані з моделей, були використані разом, і рішення було ініційоване системою прогнозування з урахуванням відсотка ризику для кожної з чотирьох моделей розробки програмного забезпечення [25]. Набір даних був протестований за допомогою натренованої моделі для прийняття рішення щодо істинності або хибності, де істина позначає обробку даних і навчання моделі, а хибність означає, що дані не обробляються.

Метрика продуктивності – точність (під час навчання та тестування) – представлена у вигляді графіка для кожного набору даних програмної моделі. Графік був побудований з використанням точності навчання та тестування для кожної моделі розробки програмного забезпечення, як показано на рисунках 3.3–3.6.

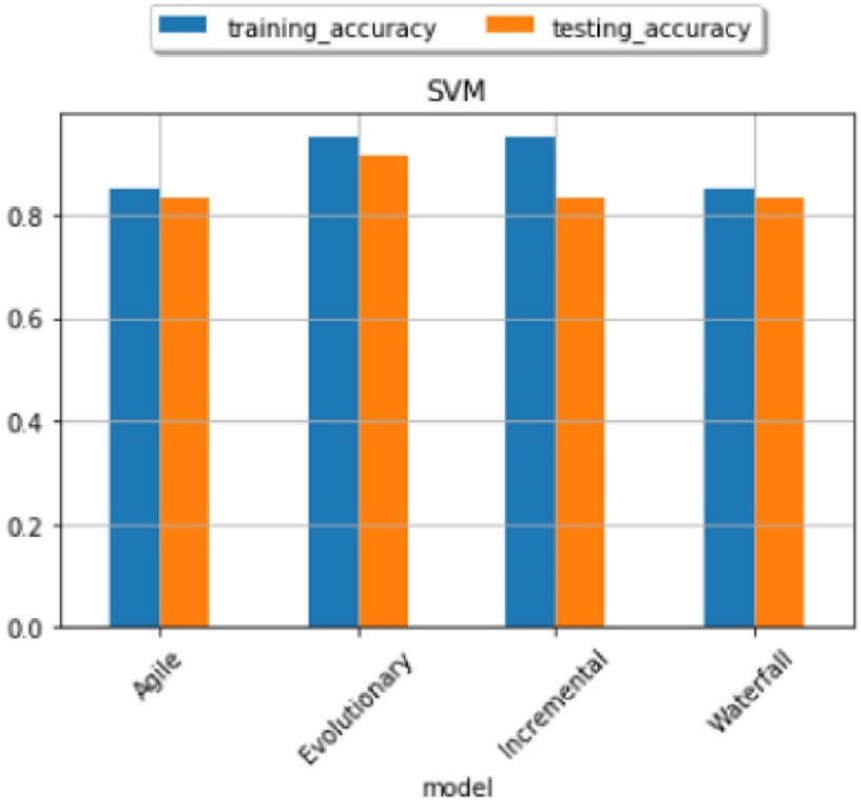


Рисунок 3.3 – Метрика точності: SVM

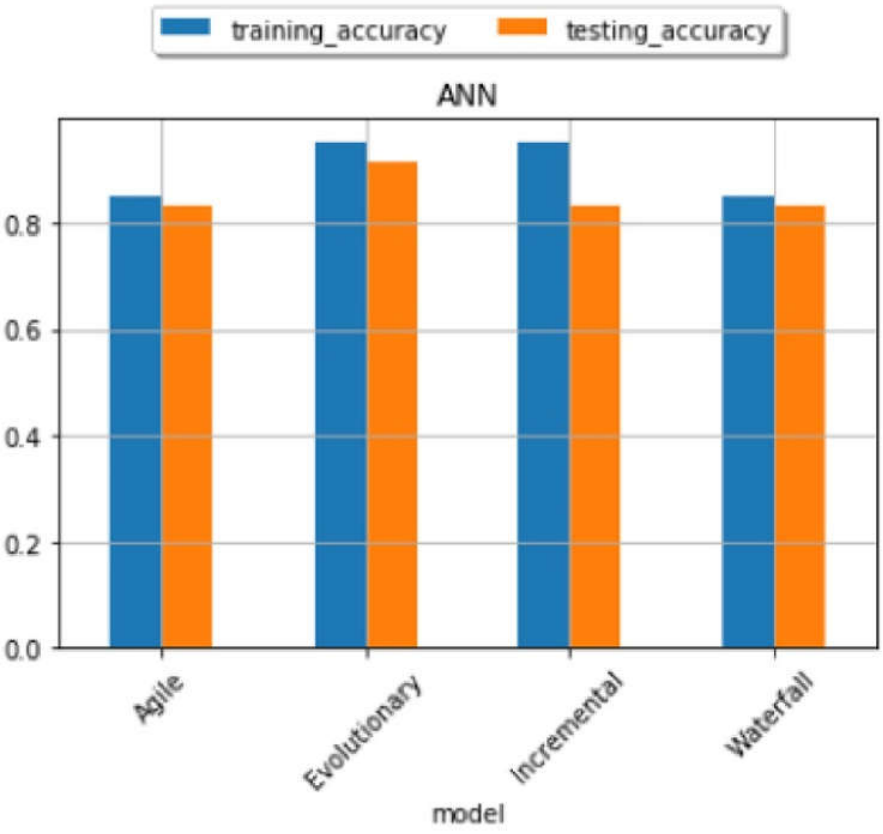


Рисунок 3.4 – Метрика точності: ANN-MLP

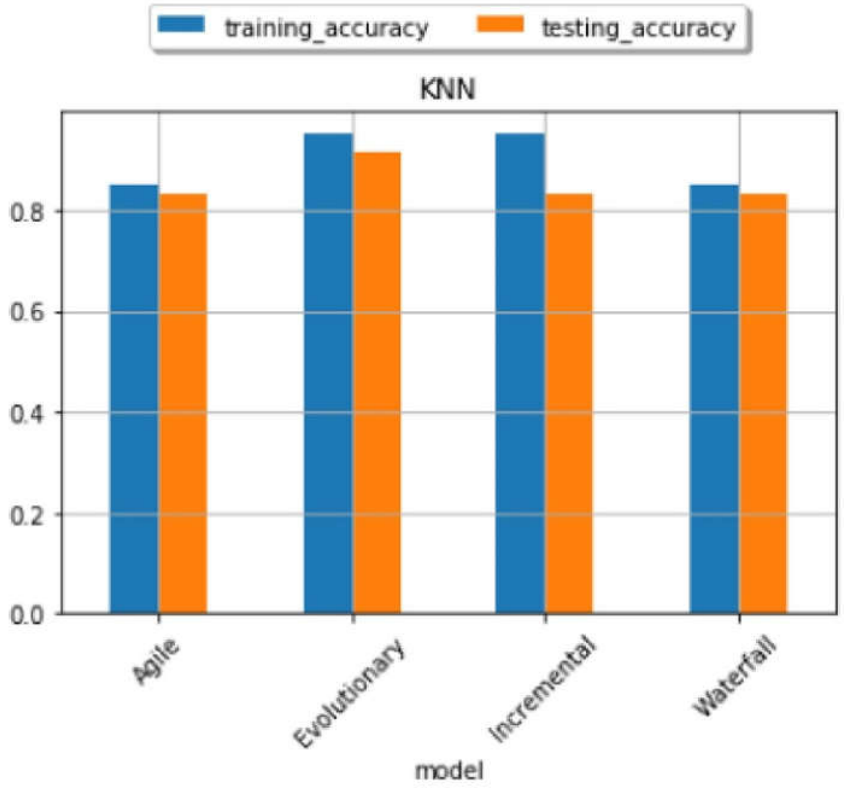


Рисунок 3.5 – Метрика точності: k-NN

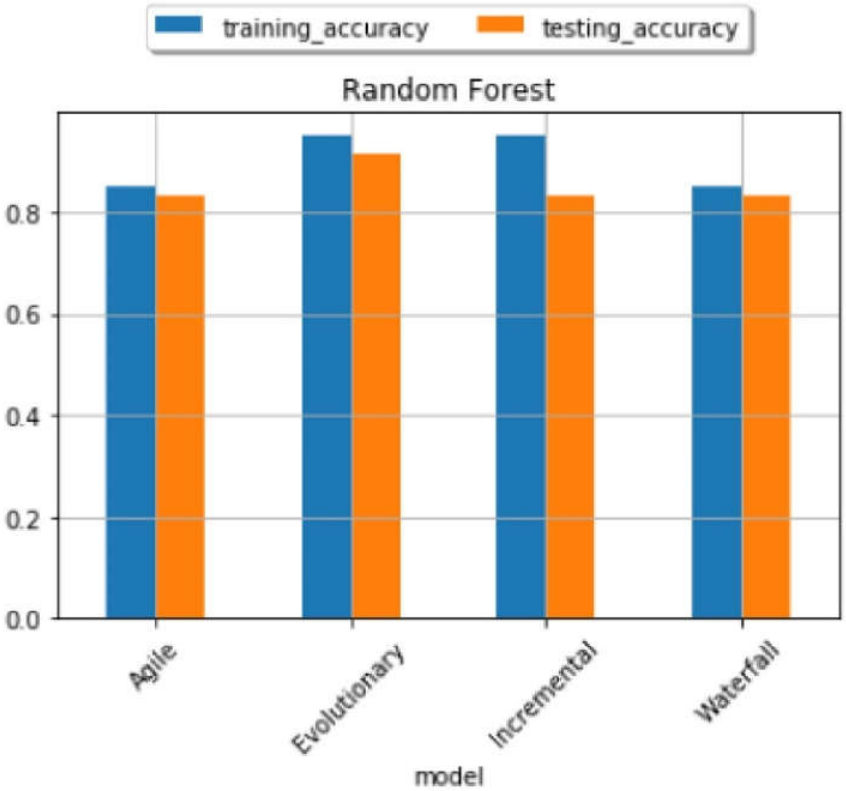


Рисунок 3.6 – Метрика точності: RF

З аналізу видно, що еволюційний та інкрементний підходи забезпечують вищу точність, за ними йдуть гнучка та водоспадна моделі. Виходячи з обчисленої точності для цих моделей розробки програмного забезпечення, відсоток ризику розраховується на основі розробленої моделі прогнозування.

При вимкненій системі прогнозування результати виглядають наступним чином: інкрементний підхід очолює список з 28,34%, водоспадна модель займає 44,4%, еволюційна модель має 79,61%, а гнучка – 99,67%. Це свідчить про те, що гнучка модель, яка найчастіше використовується в індустрії без системи прогнозування, отримала найвищий відсоток ризику, за нею йдуть еволюційна, водоспадна та інкрементна моделі.

З іншого боку, при активній натренованій моделі, результат виявився більш обнадійливим: гнучка модель отримала 0,54% ризику, еволюційна – 27,65%, водоспадна – 40,65%, а інкрементна – 72,9%. Усі аспекти оцінювання показали, що модель з прогнозуванням перевершує модель без прогнозування.

З урахуванням впровадження моделі прогнозування в системі, заснованій на IoT-Fog, стає очевидним, що гнучка модель має найменший відсоток ризику, за нею йдуть еволюційна, водоспадна та інкрементна моделі. Це виявилося корисним для впровадження моделей розробки програмного забезпечення у будь-які системи IoT-Fog для застосування в реальному часі, що призводить до максимальної успішності.

Висновки до розділу 3

1. Методика проведення експериментальних досліджень у розробці програмного забезпечення з використанням машинного навчання передбачає навчання та тестування різних класифікаторів на основі зібраного набору даних. До них належать методи, такі як машина опорних векторів (SVM), k-ближчих сусідів (k-NN), штучна нейронна мережа (ANN) та випадковий ліс (Random Forest). Кожен класифікатор має свої особливості: випадковий ліс забезпечує високу точність через усереднення результатів, SVM ефективно працює з

високовимірними даними, а нейронні мережі та k-NN здатні враховувати складні зв'язки між ознаками.

2. Реалізація класифікаторів машинного навчання в контексті прогнозування ризиків та вибору моделей розробки програмного забезпечення передбачає навчання та оцінку різних моделей, які дозволяють аналізувати вхідні дані, зібрані з опитувань експертів, і створювати точні прогнози щодо відповідної моделі розробки, враховуючи можливі ризики. Використання бібліотеки scikit-learn у Python спрощує процес обробки даних, створення моделей, оцінки їхньої точності та впровадження у реальні проєкти.

3. Результати експериментальних досліджень показали, що використання ймовірнісних оцінок від різних моделей машинного навчання дозволяє системі прогнозування визначати ризики для кожної з чотирьох моделей розробки програмного забезпечення. Під час тестування натреновані моделі приймали рішення щодо обробки даних, що дозволило визначити істинність або хибність прогнозів. Аналіз точності навчання та тестування виявив, що еволюційний та інкрементний підходи демонструють найвищу точність, тоді як гнучка та водоспадна моделі дещо відстають. При цьому, коли система прогнозування не використовувалася, гнучка модель мала найвищий рівень ризику (99,67%), тоді як активне використання системи знизило ризик для цієї моделі до 0,54%. Це підтверджує перевагу застосування системи прогнозування, яка дозволяє значно знизити ризики та підвищити успішність реалізації проєктів у середовищах з високою невизначеністю, таких як IoT-Fog.

ВИСНОВКИ

В даній роботі запропоновано метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання. Теоретично описано та експериментально підтверджено, як ансамбль об’єднаних класифікаторів може підвищити точність прогнозування ризиків для наданих даних. Виходячи з прогнозування ризиків для різних моделей розробки програмного забезпечення, зацікавлені сторони програмного продукту можуть прийняти рішення щодо вибору найбільш ефективної моделі.

Основні висновки роботи:

1. Управління ризиками у розробці програмного забезпечення є критично важливим елементом для забезпечення успішного завершення проєктів. Це включає ідентифікацію, аналіз та оцінку потенційних ризиків, таких як затримки у виконанні, перевищення бюджету, технічні проблеми, відсутність відповідності вимогам замовника та інші загрози. Традиційні підходи до управління ризиками зазвичай базуються на експертних оцінках і застосовуються вручну, що може призводити до суб'єктивних рішень і недостатньої точності у виявленні ризиків на ранніх етапах проєкту.

2. Для успішної реалізації програмного проєкту важливо правильно вибрати модель розробки, яка відповідатиме специфіці вимог, рівню ризиків та динаміці змін у проєкті. Використання інструментів машинного навчання для аналізу ризиків та вибору оптимальної моделі дозволяє підвищити точність прогнозування та знизити ймовірність виникнення проблем у процесі розробки. Це робить процес управління розробкою програмного забезпечення більш надійним та ефективним, забезпечуючи кращу відповідність вимогам замовника та успішне завершення проєктів.

3. Метод прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання є ключовим інструментом для забезпечення успіху сучасних програмних проєктів. Він дозволяє виявляти та передбачати ризики, які можуть вплинути на хід розробки, терміни виконання та якість

кінцевого продукту. Використовуючи історичні дані та потужність машинного навчання, цей метод забезпечує точний аналіз та автоматизацію процесу управління ризиками.

4. Збір даних та формування навчальної і тестової вибірок є критичним етапом у побудові системи прогнозування ризиків у проєктах програмного забезпечення. Опитування серед експертів дозволило зібрати якісні дані, що містять інформацію про метрики та атрибути різних моделей розробки, таких як Agile, Waterfall, інкрементна та еволюційна моделі. Зібраний набір даних був поділений на навчальну та тестову вибірки у співвідношенні 80-20, що забезпечує ефективне навчання моделей машинного навчання та їх перевірку на нових даних.

5. Система рекомендацій для розробки програмного забезпечення є важливим інструментом, який допомагає управляти ризиками у проєктах. Її основна функція полягає у забезпеченні взаємодії між ключовими зацікавленими сторонами та аналізі вхідних даних для вибору оптимальної моделі розробки. Ця система дозволяє отримувати вимоги від власника продукту, обробляти їх та надавати рекомендації для менеджера продукту, спонсора продукту та власника архітектури. Завдяки цьому забезпечується як технічна, так і бізнесова узгодженість у межах проєкту.

6. Методика проведення експериментальних досліджень у розробці програмного забезпечення з використанням машинного навчання передбачає навчання та тестування різних класифікаторів на основі зібраного набору даних. До них належать методи, такі як машина опорних векторів (SVM), k-ближчих сусідів (k-NN), штучна нейронна мережа (ANN) та випадковий ліс (Random Forest). Кожен класифікатор має свої особливості: випадковий ліс забезпечує високу точність через усереднення результатів, SVM ефективно працює з високовимірними даними, а нейронні мережі та k-NN здатні враховувати складні зв'язки між ознаками.

7. Реалізація класифікаторів машинного навчання в контексті прогнозування ризиків та вибору моделей розробки програмного забезпечення

передбачає навчання та оцінку різних моделей, які дозволяють аналізувати вхідні дані, зібрані з опитувань експертів, і створювати точні прогнози щодо відповідної моделі розробки, враховуючи можливі ризики. Використання бібліотеки scikit-learn у Python спрощує процес обробки даних, створення моделей, оцінки їхньої точності та впровадження у реальні проєкти.

8. Результати експериментальних досліджень показали, що використання ймовірнісних оцінок від різних моделей машинного навчання дозволяє системі прогнозування визначати ризики для кожної з чотирьох моделей розробки програмного забезпечення. Під час тестування натреновані моделі приймали рішення щодо обробки даних, що дозволило визначити істинність або хибність прогнозів. Аналіз точності навчання та тестування виявив, що еволюційний та інкрементний підходи демонструють найвищу точність, тоді як гнучка та водоспадна моделі дещо відстають. При цьому, коли система прогнозування не використовувалася, гнучка модель мала найвищий рівень ризику (99,67%), тоді як активне використання системи знизило ризик для цієї моделі до 0,54%. Це підтверджує перевагу застосування системи прогнозування, яка дозволяє значно знизити ризики та підвищити успішність реалізації проєктів у середовищах з високою невизначеністю, таких як IoT-Fog.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pressman R. S. Software Engineering: a practitioners approach. URL: https://invent.ilmkidunya.com/images/Section/12.pdf.

2. Rizwan M., Nadeem A., Sindhu M.A. Analyses of classifier's performance measures used in software fault prediction studies. *IEEE Access*. 2019. Vol. 7. P. 82764–82775.

3. Kühn E. Reusable Coordination Components: Reliable Development of Cooperative Information Systems. *International Journal of Software Engineering and Knowledge Engineering*. 2017. Vol. 25, №. 4. Article 1740001.

4. Patil S., Ade R. Software requirement engineering risk prediction model. *International Journal of Computer Applications*. 2014. Vol. 102, №. 2. P. 1–6.

5. Chow T., Cao D.B. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*. 2008. Vol. 81, №. 6. P. 961–971.

6. Jones C. Software metrics: good, bad and missing. *IEEE Computer*. 1994. Vol. 27, №. 9. P. 98–100.

7. Mattos D.M.F., Velloso P.B., Duarte O.C.M.B. An agile and effective network function virtualization infrastructure for the Internet of Things. *Journal of Internet Services and Applications*. 2019. Vol. 10, №. 1. Article 12.

8. Dam H.K., Tran T., Grundy J., Ghose A., Kamei Y. Towards effective AI-powered agile project management. In: *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 2018. P. 41–44.

9. Rai A.K., Agrawal S., Khaliq M. Identification of agile software risk indicators and evaluation of agile software development project risk occurrence probability. *International Journal of Engineering Technology Management and Applied Sciences*. 2017. Vol. 5, №. 7. P. 403–408.

10. Li Q., Meng S., Zhang S., Wu M., Zhang J., Ahvanooey M.T., Aslam M.S. Safety risk monitoring of cyber-physical power systems based on ensemble learning algorithm. *IEEE Access*. 2019. Vol. 7, №. 24. P. 788–805.

11. Tavares B.G., Da Silva C.E.S., De Souza A.D. Practices to improve risk management in agile projects. *International Journal of Software Engineering and Knowledge Engineering*. 2019. Vol. 29, №. 3. P. 1–19.

12. Tavares A.D., Da Silva Breno Gontijo, De Souza Carlos Eduardo Sanches. Risk management analysis in Scrum software projects. *International Transactions in Operational Research*. 2019. Vol. 26, №. 5. P. 1884–1905.

13. Hu Y., Du J., Zhang X., Hao X., Ngai E.W.T., Fan M., Liu M. An integrative framework for intelligent software project risk planning. *Decision Support Systems*. 2013. Vol. 55, №. 4. P. 927–937.

14. Asif M., Ahmed J. A novel case base reasoning and frequent pattern-based decision support system for mitigating software risk factors. *IEEE Access*. 2020. Vol. 8. P. 102278–102291.

15. Zhang X., Ji L., Tao X., Li Y., Chen F., Luo Y., Zhu M. Coupling a fast Fourier transformation with a machine learning ensemble model to support recommendations for heart disease patients in a telehealth environment. *IEEE Access*. 2017. Vol. 5. P. 10674–10685.

16. Qiu J., Luo W., Pan L., Tai Y., Zhang J., Xiang Y. Predicting the impact of Android malicious samples via machine learning. *IEEE Access*. 2019. Vol. 7. P. 66304–66316.

17. Random Forest Classifier. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата звернення: 15.06.2024).

18. Support Vector Machines. URL: https://scikit-learn.org/stable/modules/svm.html#classification (дата звернення: 15.06.2024).

19. Neural Network Models. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron (дата звернення: 15.06.2024).

20. Nearest Neighbors. URL: https://scikit-learn.org/stable/modules/neighbors.html#classification (дата звернення: 15.06.2024).

21. Moreb M., Mohammed T.A., Bayat O. A novel software engineering approach toward using machine learning for improving the efficiency of health systems. *IEEE Access*. 2020. Vol. 8. P. 23169–23178.

22. Paltrinieri N., Comfort L., Reniers G. Learning about risk: machine learning for risk assessment. *Safety Science*. 2019. Vol. 118. P. 475–486.

23. Multi-layer Perceptron Classifier. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (дата звернення: 15.06.2024).

24. Sankaranarayanan S., Prabhakar M., Satish S., Jain P., Ramprasad A., Krishnan A. Flood prediction based on weather parameters using deep learning. *Journal of Water and Climate Change*. 2019. P. 1–18.

25. Wang Y., Qin Y., Xu W., Zhang W., Zhang Y., Wu X. Machine learning methods for driving risk prediction. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on the Use of GIS in Emergency Management (EM-GIS). 2017. P. 1–6.

26. Yong Hu et al. Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction. *Decision Support Systems*. 2015. Vol. 72. P. 11–23.

27. Yong Hu et al. Software project risk analysis using Bayesian networks with causality constraints. *Decision Support Systems*. 2013. Vol. 56, №. 1. P. 439–449.

28. Confroti R. et al. A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems*. 2015. Vol. 69. P. 1–9.

29. Cortes C., Vapnik V.N. Support-vector networks. Mach. Learn. 1995. Vol. 20. P. 273–297.

30. Altman Naomi S. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 1992. Vol. 46 (3). P.175–185.

31. Haykin S. Neural Networks: A Comprehensive Foundation (2nd. ed.). Prentice Hall PTR, USA, 1998. 823 p.

32. William W.H. Machine learning methods in the environmental sciences. New York: Cambridge University Press, 2009. 345 p.

33. Мись А. Р., Рипіч Б. В., Шевчук В. М. Сучасні підходи до управління проєктами: машинне навчання та блокчейн-технології. *Світ наукових досліджень* : матеріали міжнародної мультидисциплінарної наукової інтернет-конференції (випуск 35), м. Тернопіль, Україна, м. Ополе, Польща, 20-21 листопада 2024 р. Тернопіль, 2024. С. 104–106.

34. Мись А. Р. Концептуальні основи методу прогнозування ризиків у проєктах програмного забезпечення за допомогою машинного навчання. *Інформаційне суспільство: технологічні, економічні та технічні аспекти* становлення : матеріали міжнародної науково-практичної інтернет-конференції (випуск 94), м. Ополе, Польща, 11-12 грудня 2024 р. URL: http://www.konferenciaonline.org.ua/ua/articles/year-4/rozdil-40/pidrozdil-121/pidrozdil2-0/.

35. Островерхов В.М., Біловус Л.І., Возьний К.З., Луцишин О.О., Монастирський Г.Л., Надвиничний С.А., Питель С.В., Шандрук С.К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

36. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Турченко І.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Управління проєктами» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 32 с.

Додаток А

Програмний код реалізації класифікаторів машинного навчання

А.1. Код для реалізації класифікатора на основі машини опорних векторів.

```
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

# 1. Завантаження та підготовка даних
# Завантаження даних з CSV-файлу
data = pd.read_csv('software_models_data.csv')  # Замініть на ваш файл CSV

# Перевірка перших кількох рядків
print(data.head())

# Видалення рядків з відсутніми значеннями
data = data.dropna()

# Розділення на ознаки (X) та цільову змінну (y)
X = data.drop('Model', axis=1)  # Ознаки (метрики)
y = data['Model']  # Цільова змінна (модель розробки)

# 2. Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f'Розмір навчального набору: {X_train.shape}')
print(f'Розмір тестового набору: {X_test.shape}')
```

```
# 3. Навчання моделі SVM
svm_classifier = SVC(kernel='linear', probability=True, random_state=42)
svm_classifier.fit(X_train, y_train)

# Прогнозування на тестовому наборі
y_pred = svm_classifier.predict(X_test)

# 4. Оцінка точності моделі
accuracy = accuracy_score(y_test, y_pred)
print(f'Точність моделі SVM: {accuracy:.2f}')
print('Звіт про класифікацію:')
print(classification_report(y_test, y_pred))

# Матриця невідповідностей
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Передбачені значення')
plt.ylabel('Фактичні значення')
plt.title('Матриця невідповідностей для SVM')
plt.show()

# 5. Прогнозування нових даних
# Приклад нових даних для прогнозування
new_data = pd.DataFrame({
    'Bandwidth': [5],
    'Latency': [5],
    'Edge_Exposure_Range': [5],
    'Authentication_Factors': [7],
    'Vulnerability': [7],
    'Past_Faults_Exp': [5],
    'Method': [7],
    'Delivery_Schedule': [5],
```

```
    'Error_Rate': [7],
    'Accuracy': [7],
    'Efficiency': [7],
    'Interface_Effectiveness': [8],
    'Availability_Traffic': [7],
    'Cost': [5],
    'Maintenance': [5]
})

# Прогнозування на нових даних
predicted_model = svm_classifier.predict(new_data)
print(f'Прогнозована модель розробки програмного забезпечення: {predicted_model[0]}')

# 6. Збереження натренованої моделі (необов’язково)
joblib.dump(svm_classifier, 'svm_model.pkl')
print("Модель збережено у файл 'svm_model.pkl'")

# Завантаження моделі (для перевірки, що все працює)
loaded_model = joblib.load('svm_model.pkl')
print("Модель завантажено з файлу 'svm_model.pkl'")
```

А.2. Код для реалізації класифікатора на основі машини опорних векторів.

```
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

```
# 1. Завантаження та підготовка даних
# Завантаження даних з CSV-файлу
data = pd.read_csv('software_models_data.csv')  # Замініть на ваш файл CSV

# Перевірка перших кількох рядків
print(data.head())

# Видалення рядків з відсутніми значеннями
data = data.dropna()

# Розділення на ознаки (X) та цільову змінну (y)
X = data.drop('Model', axis=1)  # Ознаки (метрики)
y = data['Model']  # Цільова змінна (модель розробки)

# 2. Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f'Розмір навчального набору: {X_train.shape}')
print(f'Розмір тестового набору: {X_test.shape}')

# 3. Навчання моделі k-NN
k = 5  # Кількість сусідів
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train, y_train)

# Прогнозування на тестовому наборі
y_pred = knn_classifier.predict(X_test)

# 4. Оцінка точності моделі
accuracy = accuracy_score(y_test, y_pred)
print(f'Точність моделі k-NN (k={k}): {accuracy:.2f}')
```

```
print('Звіт про класифікацію:')
print(classification_report(y_test, y_pred))

# Матриця невідповідностей
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Передбачені значення')
plt.ylabel('Фактичні значення')
plt.title(f'Матриця невідповідностей для k-NN (k={k})')
plt.show()

# 5. Прогнозування нових даних
# Приклад нових даних для прогнозування
new_data = pd.DataFrame({
    'Bandwidth': [5],
    'Latency': [5],
    'Edge_Exposure_Range': [5],
    'Authentication_Factors': [7],
    'Vulnerability': [7],
    'Past_Faults_Exp': [5],
    'Method': [7],
    'Delivery_Schedule': [5],
    'Error_Rate': [7],
    'Accuracy': [7],
    'Efficiency': [7],
    'Interface_Effectiveness': [8],
    'Availability_Traffic': [7],
    'Cost': [5],
    'Maintenance': [5]
})
```

```
# Прогнозування на нових даних
predicted_model = knn_classifier.predict(new_data)
print(f'Прогнозована модель розробки програмного забезпечення: {predicted_model[0]}')

# 6. Збереження натренованої моделі (необов’язково)
joblib.dump(knn_classifier, 'knn_model.pkl')
print("Модель збережено у файл 'knn_model.pkl'")

# Завантаження моделі (для перевірки, що все працює)
loaded_model = joblib.load('knn_model.pkl')
print("Модель завантажено з файлу 'knn_model.pkl'")
```

А.3. Код для реалізації класифікатора на основі штучної нейронної мережі з багатошаровим персептроном.

```
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

# 1. Завантаження та підготовка даних
# Завантаження даних з CSV-файлу
data = pd.read_csv('software_models_data.csv')  # Замініть на ваш файл CSV

# Перевірка перших кількох рядків
print(data.head())
```

```
# Видалення рядків з відсутніми значеннями
data = data.dropna()

# Розділення на ознаки (X) та цільову змінну (y)
X = data.drop('Model', axis=1)  # Ознаки (метрики)
y = data['Model']  # Цільова змінна (модель розробки)

# 2. Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f'Розмір навчального набору: {X_train.shape}')
print(f'Розмір тестового набору: {X_test.shape}')

# 3. Навчання моделі ANN-MLP
# Визначення структури нейронної мережі
mlp_classifier = MLPClassifier(hidden_layer_sizes=(64, 64, 64),  # Три приховані шари по 64 нейрони кожний
                               activation='relu',  # Функція активації ReLU
                               max_iter=500,  # Максимальна кількість ітерацій
                               random_state=42)

# Навчання нейронної мережі
mlp_classifier.fit(X_train, y_train)

# Прогнозування на тестовому наборі
y_pred = mlp_classifier.predict(X_test)

# 4. Оцінка точності моделі
accuracy = accuracy_score(y_test, y_pred)
print(f'Точність моделі ANN-MLP: {accuracy:.2f}')
print('Звіт про класифікацію:')
```

```
print(classification_report(y_test, y_pred))

# Матриця невідповідностей
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Передбачені значення')
plt.ylabel('Фактичні значення')
plt.title('Матриця невідповідностей для ANN-MLP')
plt.show()

# 5. Прогнозування нових даних
# Приклад нових даних для прогнозування
new_data = pd.DataFrame({
    'Bandwidth': [5],
    'Latency': [5],
    'Edge_Exposure_Range': [5],
    'Authentication_Factors': [7],
    'Vulnerability': [7],
    'Past_Faults_Exp': [5],
    'Method': [7],
    'Delivery_Schedule': [5],
    'Error_Rate': [7],
    'Accuracy': [7],
    'Efficiency': [7],
    'Interface_Effectiveness': [8],
    'Availability_Traffic': [7],
    'Cost': [5],
    'Maintenance': [5]
})

# Прогнозування на нових даних
```

```
predicted_model = mlp_classifier.predict(new_data)
print(f'Прогнозована модель розробки програмного забезпечення: {predicted_model[0]}')

# 6. Збереження натренованої моделі (необов’язково)
joblib.dump(mlp_classifier, 'mlp_model.pkl')
print("Модель збережено у файл 'mlp_model.pkl'")

# Завантаження моделі (для перевірки, що все працює)
loaded_model = joblib.load('mlp_model.pkl')
print("Модель завантажено з файлу 'mlp_model.pkl'")
```

Додаток Б

Копії публікацій автора

УДК 001 (063)

Світ наукових досліджень. Випуск 35: матеріали Міжнародної мультидисциплінарної наукової інтернет-конференції (м. Тернопіль, Україна, м. Ополе, Польща, 20-21 листопада 2024 р.) / за ред. : О. Патряк та ін. ГО "Наукова спільнота", WSZIA w Opolu. Тернопіль: ФО- П Шпак В.Б. 2024. 278 с.

Збірник наукових публікацій укладено за матеріалами доповідей наукової мультидисциплінарної інтернет-конференції «Світ наукових досліджень. Випуск 35», які оприлюднені на інтернет-сторінці www.economy-confer.com.ua

Оргкомітет

Адреса оргкомітету:
46005, Україна, м. Тернопіль, а/с 797
тел. +380977547363 e-mail: economy-confer@ukr.net

ISSN 2786-6823 (print)

Література:

1. Golovko V., Kroshchanka A., Mikhno E., Komar M., Sachenko A. Deep convolutional neural network for detection of solar panels. Lecture Notes on Data Engineering and Communications Technologies. 2020. 48. 371-389.
2. Zotov V. B., Demin S. S., Glazkova I. Y. Features of material flow accounting for the efficient supply chain management. *International Journal of Supply Chain Management*, 2019. 8.
3. Kalinov I., Petrovsky A., Ilin V., Pristanskiy E., Kurenkov M., Ramzhaev V., et al. WareVision: CNN barcode detection-based UAV trajectory optimization for autonomous warehouse stocktaking. *IEEE Robotics and Automation Letters*, 2020, 5. http://dx.doi.org/10.1109/LRA.2020.3010733.
4. Molling G., Klein, A. Z. Value proposition of IoT-based products and services: A framework proposal. *Electronic Markets*, 2022, 32. http://dx.doi.org/10.1007/s12525-022-00548-w.

СУЧАСНІ ПІДХОДИ ДО УПРАВЛІННЯ ПРОЄКТАМИ: МАШИННЕ НАВЧАННЯ ТА БЛОКЧЕЙН-ТЕХНОЛОГІЇ

Мись Андрій Русланович
магістрант, Західноукраїнський національний університет, м. Тернопіль

Рипіч Богдан Віталійович
магістрант, Західноукраїнський національний університет, м. Тернопіль

Шевчук Вадим Михайлович
магістрант, Західноукраїнський національний університет, м. Тернопіль

Інтернет-адреса публікації на сайті:
https://www.economy-confer.com.ua/full-article/5879/

В умовах зростаючої складності програмних проєктів і невизначеності у плануванні дедалі більшої популярності набувають інтелектуальні технології для управління проєктами. Традиційні методи, хоча й забезпечують певний рівень ефективності, часто не здатні враховувати всі взаємозв'язки між параметрами проєкту, що призводить до неточних оцінок і ризиків перевитрат.

Проблема оцінки зусиль для розробки програмного забезпечення залишається відкритою через наявність складних взаємозв’язків між численними параметрами проєкту, що часто призводить до неточних прогнозів [1]. Сучасні методи оцінки стикаються з труднощами при обробці непослідовних, неповних, розріджених та недостатньо задокументованих наборів даних, що є однією

з основних причин неточності результатів [2]. Існуючі моделі не можуть ефективно обробляти нелінійні зв'язки між характеристиками проєктів та обсягом зусиль, що знижує їх ефективність і надійність [3].

Глибокі нейронні мережі дозволили покращити точність оцінки, проте їх використання обмежується через недоліки, такі як тривалий час навчання, перенавчання та недонавчання [4].

Недоліки існуючих підходів оцінки зусиль розробки програмного забезпечення можна сформулювати наступним чином:

- відсутність достатньої уваги до диференціації функціональності програмного забезпечення, що призводить до неточних оцінок зусиль для програмних проєктів;
- відсутність групування проєктів за специфікаціями, що ускладнює обробку неточних та неякісних даних;
- невідповідний вибір ознак, що призводить до навчання моделей на непідходящих даних і, відповідно, до неточних прогнозів;
- використання необроблених даних з історичних проєктів без попередньої обробки, що обмежує точність класифікації.

З огляду на вищезазначені недоліки, актуальність полягає в розробці підходу для оцінки зусиль розробки програмного забезпечення, який дозволить підвищити точність прогнозування, зменшити час навчання та забезпечити адаптацію моделі до характеристик проєктів. Для вирішення цієї задачі необхідно дослідити методи класифікації та обробки даних, що враховують специфіку програмних проєктів, нелінійні зв'язки між параметрами, а також забезпечують попередню обробку даних для підвищення якості класифікації.

Останніми роками розробники програмного забезпечення зосереджують свою увагу на використанні конкретних методологій або моделей розробки відповідно до вимог програмного продукту. Після вивчення наукових праць та обговорення з експертами галузі було обрано чотири моделі, які будуть розглядатися. Це були водоспадна модель, інкрементна модель, еволюційна модель та гнучкі методології [5].

Щоб відповідати останнім нововведенням, таким як граничні обчислення, інтернет речей та машинне навчання, було б надзвичайно корисно, якби існувала система, яка могла б передбачати відповідну модель розробки програмного забезпечення залежно від вимог до програмного продукту. Оцінювання якості програмного забезпечення є важливим, але також складним процесом. З цієї причини моделі прогнозування дефектів програмного забезпечення отримали широке застосування. З іншого боку, визначення відповідної моделі та вибір найбільш ефективної з наявних моделей залежать від факторів продуктивності [6]. Отже, задача прогнозування ризиків у проєктах програмного забезпечення є актуальною.

Перший крок до успіху проєкту – це створення надійного та точного графіка. Один із методів складання графіків – це метод критичного шляху (Critical path method, CPM), який обчислює ранні та пізні дати початку і завершення робіт, аналізуючи шляхи проєктної мережі вперед і назад. Після цього до кожної діяльності додаються необхідні ресурси. Власники завдань також додають резерви часу або страхові запаси для кожної діяльності, щоб подолати невизначеності [7, 8].

Інтеграція технології блокчейн в управління проєктами може покращити можливості планування та контролю проєктів. Блокчейн – це розподілена база даних, яка ділиться між вузлами комп'ютерної мережі, забезпечує надійне збереження інформації та унеможливлює її підробку. Блокчейн може допомогти керівникам проєктів знизити витрати на моніторинг і контроль, підвищуючи загальну ефективність проєкту. Він також може покращити моніторинг виконання завдань за часом, дозволяючи кожному учаснику оновлювати свій прогрес у реальному часі без необхідності залучення координатора проєкту, зменшуючи витрати на вимірювання для контролю термінів виконання.

Література:

1. Fadhil A. A., Alsarraj R. G., Altaie A. M. Software cost estimation based on dolphin algorithm. IEEE Access. 2020. 8. 75279-75287.
2. Singal P., Kumari A. C., Sharma P. Estimation of software development effort: A differential evolution approach. Procedia Computer Science. 2020. 167. 2643-2652.
3. Ezghari S., Zahi A. Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method. Applied Soft Computing. 2018. 67. 540-557.
4. Zare F., Zare H. K., Fallahnezhad M. S. Software effort estimation based on the optimal Bayesian belief network. Applied Soft Computing. 2016. 49. 968-980.
5. Pressman R. S. Software Engineering: a practitioners approach. URL: https://invent.ilmkidunya.com/images/Section/12.pdf.
6. Rizwan M., Nadeem A., Sindhu M.A. Analyses of classifier’s performance measures used in software fault prediction studies. IEEE Access. 2019. 7. 82764-82775.
7. Herroelen W., Leus R. On the merits and pitfalls of critical chain scheduling. Journal of Operations Management. 2001. 19(5). 559-577.
8. Kannan J., Chitra G. Critical chain over critical path in construction projects. International Journal of Engineering and Management Research. 2017. 7 (1). 338-344.

Відкрити тези доповіді »

01.12.2024 13:32

КОНЦЕПТУАЛЬНІ ОСНОВИ МЕТОДУ ПРОГНОЗУВАННЯ РИЗИКІВ У ПРОЄКТАХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

[1. Інформаційні системи і технології]

Автор: **Мись Андрій Русланович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

01.12.2024 13:35

ІНТЕГРАЦІЯ ЕВОЛЮЦІЙНИХ АЛГОРИТМІВ І МЕТОДУ АНАЛОГІЙ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ОЦІНКИ ВАРТОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

[1. Інформаційні системи і технології]

Автор: **Мормуль Андрій Романович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

01.12.2024 13:39

МЕТОД ВІДНОВЛЕННЯ ПРОПУЩЕНИХ ТА ПОШКОДЖЕНИХ ДАНИХ У МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ

[1. Інформаційні системи і технології]

Автор: **Пархін Антоній Романович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

01.12.2024 13:44

ПІДВИЩЕННЯ НАДІЙНОСТІ ВУЗЛІВ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ АНСАМБЛЕВОГО ГЛИБОКОГО НАВЧАННЯ

[1. Інформаційні системи і технології]

Автор: **Пилип'як Назар Богданович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

01.12.2024 13:46

УПРАВЛІННЯ СКЛАДСЬКИМИ ЗАПАСАМИ НА ОСНОВІ ІНТЕГРАЦІЇ ТЕХНОЛОГІЙ КОМП'ЮТЕРНОГО ЗОРУ ТА МАШИННОГО НАВЧАННЯ

[1. Інформаційні системи і технології]

Автор: **Процик Олександр Михайлович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Мись Андрій Русланович, магістр

Західноукраїнський національний університет, м. Тернопіль

КОНЦЕПТУАЛЬНІ ОСНОВИ МЕТОДУ ПРОГНОЗУВАННЯ РИЗИКІВ У ПРОЄКТАХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

Оцінка зусиль для розробки програмного забезпечення ускладнюється через складні взаємозв’язки між параметрами проєкту, що часто призводить до неточних прогнозів, особливо при роботі з неповними та розрідженими даними [1, 2]. Існуючі моделі не завжди враховують нелінійні залежності, що знижує їх точність і надійність [3]. У той же час розробка програмного забезпечення супроводжується численними ризиками, які впливають на терміни, бюджет і якість продукту. З огляду на складність сучасних систем і роль новітніх технологій, таких як Інтернет речей, периферійні обчислення та машинне навчання, стає актуальним впровадження інструментів для прогнозування й управління ризиками. Зокрема, метод прогнозування ризиків за допомогою машинного навчання надає можливість здійснювати автоматизований аналіз ризиків та приймати рішення на основі точних прогнозів.

На основі запропонованого методу прогнозування ризиків за допомогою машинного навчання можна розробити систему, яка може:

– ідентифікувати ключові ризики, що впливають на виконання проєкту розробки програмного забезпечення;

– прогнозувати ймовірність виникнення ризиків на основі історичних даних та аналітичних моделей;

– надати рекомендації щодо пом'якшення ризиків та підтримки прийняття рішень.

Завданнями цього методу є:

– збір даних з різних джерел, включаючи відповіді експертів, лог-файли процесів та історичні дані минулих проєктів;

– обробка та підготовка даних для створення моделі машинного навчання;

– навчання та оцінка моделей машинного навчання для досягнення високої точності прогнозування;

– інтеграція системи прогнозування з платформою управління проєктами для підтримки управлінських рішень.

Метод прогнозування ризиків базується на концептуальній моделі, яка включає три основні компоненти:

4. Збір та обробка даних.
5. Моделювання та навчання на основі машинного навчання.
6. Прогнозування та підтримка прийняття рішень.

1. Збір та обробка даних.

Цей компонент полягає у зборі якісних даних про ризики з різних джерел, таких як анкети, що заповнюються експертами з розробки програмного забезпечення, історичні дані з попередніх проєктів, дані з журналів процесів та логів систем. Наприклад, анкета може містити інформацію про використані моделі розробки (Agile, Waterfall, Incremental, Evolutionary), характеристики безпеки, пропускну здатність мережі, затримки, минулі помилки, методи розробки та інші атрибути. Обробка даних включає: видалення недійсних відповідей; нормалізацію даних для усунення масштабних відмінностей; розподіл даних на навчальні та тестові набори у співвідношенні 80-20%.

2. Моделювання та навчання на основі машинного навчання.

Основою методу є використання різних алгоритмів машинного навчання для побудови моделей прогнозування ризиків. Найпопулярнішими серед них є:

– машина опорних векторів, яка добре працює з високорозмірними даними та використовує лише частину навчальних точок для визначення функцій прийняття рішень, що робить її ефективною у використанні пам'яті;

– метод k-найближчих сусідів, який класифікує нові дані на основі найближчих зразків з навчального набору, що дозволяє застосовувати його для класифікації без наявної інформації про розподіл даних;

– штучні нейронні мережі, зокрема багатошаровий персептрон, що здатний навчатися складних взаємозв'язків між вхідними ознаками та результатами.

– випадковий ліс, який складається з ансамблю дерев рішень, що забезпечує високу точність та зменшує ризик перенавчання.

Навчання моделі проводиться на основі історичних даних, що дозволяє створити вектор ознак, який потім використовується для прогнозування ризиків на нових даних. Під час навчання функція активації RELU у прихованих шарах нейронних мереж сприяє швидшому навчанню за рахунок розрідженості нейронної мережі.

3. Прогнозування та підтримка прийняття рішень.

Після завершення навчання моделі тестовий набір даних передається через натреновану модель, що дозволяє отримати передбачені мітки ризиків для кожного нового проєкту. Цей етап включає: прогнозування відсотка ризику для кожної моделі розробки на основі ймовірностей, отриманих від класифікаторів; порівняння прогнозованих ризиків з фактичними, що дозволяє оцінити точність моделі; генерація рекомендацій для зниження ризиків на основі аналізу факторів, що впливають на ризик.

Результати прогнозування дозволяють менеджерам проєктів та іншим зацікавленим сторонам вибирати оптимальну модель розробки програмного забезпечення з найменшим рівнем ризику, що знижує ймовірність невдач та підвищує ефективність проєкту.

Література

1. Fadhil A. A., Alsarraj R. G., Altaie A. M. Software cost estimation based on dolphin algorithm. IEEE Access. 2020. 8. 75279–75287.

2. Singal P., Kumari A. C., Sharma P. Estimation of software development effort: A differential evolution approach. Procedia Computer Science. 2020. 167. 2643–2652.

3. Ezghari S., Zahi A. Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method. Applied Soft Computing. 2018. 67. 540–557.