МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна
наукова праця на правах
рукопису

ОСТАПЮК ВОЛОДИМИР ВІКТОРОВИЧ

УДК 004.91

ДИСЕРТАЦІЯ
АГЕНТНІ МОДЕЛІ В ІНФОРМАЦІЙНІЙ ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ

Спеціальність 121 – «Інженерія програмного забезпечення»

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____В.В. Остапюк

Науковий керівник:
Голуб Сергій Васильович
доктор технічних наук, професор

ЧЕРКАСИ – 2026

АНОТАЦІЯ

Остапюк В.В. Агентні моделі в інформаційній технології інтелектуального моніторингу. – Кваліфікаційна наукова праця на правах рукопису. Спеціальність 121 – «Інженерія програмного забезпечення». Черкаський державний технологічний університет. Черкаси, 2026.

Вирішено науково-прикладне завдання підвищення ефективності функціонування моніторингового програмного агента в інформаційній технології інтелектуального моніторингу шляхом розробки методів побудови та засобів програмної реалізації багатошарових агентних моделей.

Актуальність роботи зумовлена зростаючою складністю та динамічністю середовищ, у яких функціонують сучасні розподілені інформаційні системи та мікросервіс. У процесі експлуатації таких систем відбувається неминуча зміна статистичних властивостей даних та режимів функціонування.

Традиційні статичні засоби моніторингу в таких умовах втрачають адекватність, що призводить до пропуску критичних збоїв [64]. Виникає проблема між динамічною природою об’єкта та статичністю засобів спостереження. Вирішенням є створення моніторингових агентів, здатних до автономного синтезу адаптивних моделей, що забезпечують високу точність без втручання людини.

Складність цього завдання підтверджується сучасними дослідженнями у галузі MLOps (Machine Learning Operations). У реальних умовах статистичні властивості даних постійно змінюються, що призводить до явища "дрейфу концепцій" (concept drift). Оскільки в потоках телеметрії неможливо миттєво отримати істинні мітки (ground truth) для нових даних, традиційні методи моніторингу помилок стають неефективними. Тому виникає гостра потреба у використанні методів виявлення дрейфу без вчителя (unsupervised drift detection), які здатні працювати в режимі реального часу. Саме тому моніторинговий агент повинен вміти автономно адаптувати свою структуру до цих непередбачуваних змін.

Мета дослідження: Підвищення ефективності функціонування моніторингового програмного агента у інформаційній технології інтелектуального моніторингу шляхом розробки методів побудови та програмної реалізації багатошарових агентних моделей.

Наукова новизна отриманих результатів.

1. Вперше запропоновано метод багатошарового синтезу агентних моделей, який заключається у формуванні масиву вхідних даних, синтезу моделі, добавленні сигналу із виходу моделі до масиву вхідних даних і повторного синтезу моделі, який, на відміну від існуючих, використовує кластеризацію вхідних даних і побудову для кожного кластеру моделей, що дозволяє підвищити точність і адекватність результатів моделювання та профілювати метод проектування програмного забезпечення агента за рахунок підвищення однорідності точок спостереження у кластерах.

2. Удосконалено метод синтезу багатошарових моделей та їх програмної реалізації шляхом зростання різноманітності структури шарів, що дозволяє підвищити повноту опису об’єкта та покращити результати моделювання.

3. Набув подальшого розвитку метод багатошарового синтезу агентних моделей за рахунок використання у процесі проєктування алгоритмів синтезу моделей та їх програмного забезпечення спрямованого ациклічного графу (DAG), що дозволяє забезпечити багатоетапність процесу удосконалення структури моделі шляхом обробки сигналів на виході та використання моделей, побудованих у різних середовищах.

У вступі обґрунтовано актуальність теми дисертації, сформульовано мету, завдання, об'єкт та предмет дослідження, розкрито наукову новизну та практичне значення отриманих результатів.

У першому розділі розглянуто проблематику динамічних змін у середовищах інтелектуального моніторингу складних інформаційних систем. Здійснено теоретичний аналіз методів прогнозування часових рядів та обґрунтовано доцільність

застосування алгоритмів машинного навчання для задач табличної регресії. Проаналізовано проблеми MLOps щодо відтворюваності експериментів, форм представлення архітектур та обмежень існуючих форматів серіалізації моделей.

У другому розділі викладено теоретичні основи та математичну формалізацію розроблених методів. Описано метод підвищення однорідності вхідних даних шляхом кластеризації та збагачення простору ознак, а також удосконалений метод багатошарової рециркуляції з використанням гетерогенних пулів алгоритмів (ACM) та пошаровим тестуванням структури.

У третьому розділі наведено результати експериментальних досліджень на структурно відмінних відкритих наборах даних. Доведено, що комплексне застосування методів кластеризації та рециркуляції дозволяє зменшити похибку прогнозування (RMSE, MAE) порівняно з базовими алгоритмами та класичними статичними ансамблями.

У четвертому розділі здійснено проектування інформаційної технології та моніторингового програмного агента на основі декларативного підходу. Розроблено компонентну архітектуру та логіку подання багатоетапних процесів конструювання моделей у вигляді спрямованого ациклічного графа (DAG), що гарантує детермінізм обчислень та машинну відтворюваність результатів.

У висновках узагальнено результати роботи, підтверджено виконання поставлених завдань та доведено ефективність розроблених рішень.

Практичне значення отриманих результатів. Розроблений метод підвищення однорідності даних та удосконалений метод рециркуляції дозволяють моніторинговому агенту автономно розпізнавати різні режими функціонування інформаційної системи та адаптувати під них свою прогностичну логіку. Застосування декларативного підходу та подання процесів на основі DAG вирішують прикладну інженерну проблему MLOps щодо надійного перенесення навчених багатошарових ансамблів у середовище експлуатації із гарантуванням повної машинної відтворюваності обчислень.

Дисертацію виконано у Черкаському державному технологічному університеті, м. Черкаси.

Ключові слова: програмний агент, обробка інформації, модель, інтелектуальний моніторинг, машинне навчання, регресійна залежність, ансамбль моделей, бустинг, ансамблевий стекінг, дерево прийняття рішень, інтелектуальний аналіз даних, набори даних, адаптивні моделі, інформаційна система, мультиагентні системи, випадкові процеси, методи оптимізації, кластеризація, рециркуляція,, відтворюваність обчислень.

ABSTRACT

Ostapiuk V.V. Agent models in intelligent monitoring information technology. – Qualifying scientific work on the rights of a manuscript. Specialty 121 – "Software Engineering". Cherkasy State Technological University. Cherkasy, 2026.

The scientific and applied problem of improving the efficiency of a monitoring software agent in the information technology of intelligent monitoring has been solved by developing methods for constructing and software implementation tools for multilayer agent models.

The relevance of the work is driven by the growing complexity and dynamism of the environments in which modern distributed information systems and microservices operate. During the operation of such systems, an inevitable change in the statistical properties of data and operational modes occurs.

Traditional static monitoring tools lose their adequacy under such conditions, leading to missed critical failures. A problem arises between the dynamic nature of the object and the static nature of observation tools. The solution is the creation of monitoring agents capable of the autonomous synthesis of adaptive models that ensure high accuracy without human intervention.

The complexity of this task is confirmed by modern research in the field of MLOps (Machine Learning Operations). Under real-world conditions, the statistical properties of data constantly change, leading to the phenomenon of "concept drift". Since it is impossible to instantly obtain ground truth labels for new data in telemetry streams, traditional error monitoring methods become ineffective. Therefore, there is an acute need to use unsupervised drift detection methods capable of operating in real-time. That is why a monitoring agent must be able to autonomously adapt its structure to these unpredictable changes.

The goal of the research: improving the efficiency of a monitoring software agent in the information technology of intelligent monitoring through the development of methods for constructing and software implementation of multilayer agent models.

Scientific novelty of the obtained results.

1. For the first time, a method of multilayer synthesis of agent models is proposed, which consists of forming an input data array, synthesizing a model, adding a signal from the model output to the input data array, and re-synthesizing the model. Unlike existing methods, it uses input data clustering and the construction of models for each cluster, which makes it possible to improve the accuracy and adequacy of modeling results and to profile the agent software design method by increasing the homogeneity of observation points within clusters.
2. The method of synthesizing multilayer models has been improved by involving machine learning algorithms for layer formation, which makes it possible to increase the number of models in a layer, enhance the completeness of the object description, and improve the modeling results.
3. The method of multilayer synthesis of agent models received further development by using a Directed Acyclic Graph (DAG) in the process of designing model synthesis algorithms, which ensures the multi-stage nature of the model structure improvement process by processing output signals and using models built in different environments.

In the introduction, the relevance of the dissertation topic is substantiated, the goal, tasks, object, and subject of the research are formulated, and the scientific novelty and practical significance of the obtained results are revealed.

The first chapter considers the issues of dynamic changes in the environments of intelligent monitoring of complex information systems. A theoretical analysis of time series forecasting methods is performed, and the feasibility of applying machine learning algorithms for tabular regression tasks is substantiated. MLOps issues regarding experiment reproducibility, forms of architecture representation, and limitations of existing model serialization formats are analyzed.

The second chapter presents the theoretical foundations and mathematical formalization of the developed methods. It describes a method for increasing the

homogeneity of input data through clustering and feature space enrichment, as well as an improved method of multilayer recirculation using heterogeneous pools of algorithms and layer-by-layer structure testing.

The third chapter presents the results of experimental studies on structurally distinct open datasets. It is proven that the complex application of clustering and recirculation methods significantly reduces the forecasting error (RMSE, MAE) compared to basic algorithms and classical static ensembles.

The fourth chapter covers the design of information technology and a monitoring software agent based on a declarative approach. The component architecture and logic for representing multi-stage model construction processes in the form of a Directed Acyclic Graph (DAG) have been developed, which guarantees computational determinism and machine reproducibility of the results.

The conclusions summarize the results of the work, confirm the fulfillment of the set tasks, and prove the effectiveness of the developed solutions.

Practical significance of the obtained results. The developed method for increasing data homogeneity and the improved recirculation method allow the monitoring agent to autonomously recognize different operating modes of the information system and adapt its predictive logic to them without human intervention. The proposed declarative approach and the DAG-based representation of processes significantly simplify the design and solve the applied engineering MLOps problem of reliably transferring trained complex ensembles to the production environment while preserving full machine reproducibility.

The dissertation was carried out at the Cherkasy State Technological University, Ukraine, Cherkasy.

Keywords: software agent, information processing, model, intelligent monitoring, machine learning, regression dependency, ensemble of models, boosting, ensemble stacking, decision tree, data mining, data sets, adaptive models, information system, multi-agent systems, random processes, optimization methods, clustering, recirculation, reproducibility.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці в яких опубліковані основні наукові результати дисертації:

1. Ostapiuk V., Holub S. Model Synthesis Algorithms for a Monitoring Software Agent. *Mathematical Modeling and Simulation of Systems*. ред. Volodymyr Kazymyr, Anatoliy Morozov, Alexander Palagin, Serhiy Shkarlet, Nikolai Stoianov, Dmitri Vinnikov, Mark Zheleznyak. Cham : Springer Nature Switzerland, 2024. С. 113–129. DOI:10.1007/978-3-031-67348-1_9.

2. Holub S. V., Ostapiuk V. V. Machine learning of multilayer models of a monitoring software agent. *Mathematical machines and systems*. Vol. 2, 2025. P. 76–96. DOI:10.34121/1028-9763-2025-2-76-95.

3. Голуб С.В, Остапюк В. Підвищення однорідності вхідних даних у методах машинного навчання ансамблів моделей. *Актуальні проблеми автоматизації та інформаційних технологій*. Vol. 29, 2025. P. 131–155. DOI:10.15421/432513

4. Holub S. V., Ostapiuk V. V. A DECLARATIVE APPROACH TO THE DESIGN AND REPRODUCIBLE LEARNING OF COMPLEX MODEL STRUCTURES FOR MONITORING SOFTWARE AGENTS. *Ukrainian Journal of Information Technology*. Vol. 7, Issue 2. P. 1–8. DOI:10.23939/ujit2025.02.001

5. Holub S. V., Ostapiuk V. V. DAG-oriented representation of model synthesis algorithm construction processes by monitoring software agents. *Mathematical machines and systems*. Vol. 1, 2026. P. 73–86. DOI:10.34121/1028-9763-2026-1-73-86

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. Остапюк В., Голуб С. Дослідницький застосунок для побудови моніторингових агентів. *Інформаційні технології та комп'ютерне моделювання*. Івано-Франківськ, 2023 Р. 101–103

7. Остапюк В., Голуб С. Методи штучного інтелекту для вирішення задачі виявлення ключових слів публікацій у соціальних мережах. *Інформація, комунікація, суспільство 2024*. Р. 138–140.

8. Остапюк В., Голуб С. Методи штучного інтелекту для вирішення задачі виявлення рівня аварійності обладнання у робототехнічних системах. *ІНТЕГРОВАНІ ІНТЕЛЕКТУАЛЬНІ РОБОТОТЕХНІЧНІ КОМПЛЕКСИ (ІІРТК 2024)*. Київ, 2024 Р. 398 –400.

9. Остапюк В., Голуб С.В Багатошарові архітектури машинного навчання як інструмент аналізу складних даних сенсорів у робототехнічних системах. *Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2025). Вісімнадцята міжнародна науково-практична конференція*. Київ, 2025 Р. 362–364.

10. Остапюк В., Голуб С. Роль методів інтерпретації машинного навчання, зокрема SHAP, в аналізі та сегментації користувачів соціальних мереж. *XIV МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «ІНФОРМАЦІЯ, КОМУНІКАЦІЯ, СУСПІЛЬСТВО 2025» ПРИСВЯЧЕНА ПАМ'ЯТІ ПРОФЕСОРА АНДРІЯ ПЕЛЕЩИШИНА*. Львів, 2025. ISBN 978-966-994-052-0. Р. 83–84. [37]

11. Остапюк В., Голуб С. Новітні підходи до кластерного призначення даних на основі інтерпретації моделей машинного навчання. *Сучасні інформаційні технології та системи штучного інтелекту. Матеріали 1-ї Міжнародної науково-практичної конференції. Частина 1 Сучасні інформаційні технології та системи штучного інтелекту*. Харків-Яремче, 2025 Р. 134–137

12. Остапюк В., Голуб С. Інструменти для моделювання прогнозних алгоритмів. *Modern problems of science, education and society. Proceedings of the 3rd International scientific and practical conference. SPC "Sci-conf.com.ua"*. Kyiv, 2023 P. 21–27[32]

13. Остапюк В., Голуб С. Концептуальні підходи до адаптивного синтезу моделей для задач інтелектуального моніторингу. *Інформаційні технології та комп'ютерне моделювання. Матеріали Міжнародної науково-практичної конференції Інформаційні технології та комп'ютерне моделювання*. Івано-Франківськ, 2025 Р. 142–143

14. Остапюк В. Штучний інтелект, як ресурс посилення цивілізаційних спроможностей. Розвиток наукової думки: актуальні питання, досягнення та інновації. *Молодий вчений*. Хмельницький-Одеса, 2023 Р. 86–89.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСМ — алгоритм синтезу моделей

ГПО — гіперпараметрична оптимізація

МАС — мультиагентна система

МН — машинне навчання

DAG — спрямований ациклічний граф (directed acyclic graph)

DSL — предметно-орієнтована мова (domain-specific language)

GMM — модель гаусових сумішей (gaussian mixture models)

MAE — середня абсолютна похибка (mean absolute error)

MAPE — середня абсолютна відносна похибка (mean absolute percentage error)

MLOps — управління життєвим циклом машинного навчання (machine learning operations)

ONNX — відкритий формат обміну нейронними мережами (open neural network exchange)

RMSE — корінь середньоквадратичної похибки (root mean squared error)

RPA — роботизована автоматизація процесів (robotic process automation)

TPE — алгоритм оцінки парзенового дерева (tree-structured parzen estimator)

ВСТУП

Актуальність теми

Актуальність роботи зумовлена зростаючою складністю та динамічністю середовищ, у яких функціонують сучасні розподілені інформаційні системи та мікросервіси[70]. У процесі експлуатації таких систем відбувається неминуча зміна статистичних властивостей даних та режимів функціонування. Традиційні статичні засоби моніторингу в таких умовах втрачають адекватність, що призводить до пропуску критичних збоїв [64]. Виникає проблема між динамічною природою об’єкта та статичністю засобів спостереження. Вирішенням є створення моніторингових агентів, здатних до автономного синтезу адаптивних моделей, що забезпечують високу точність без втручання людини.

Складність цього завдання підтверджується сучасними дослідженнями у галузі MLOps (Machine Learning Operations). У реальних умовах статистичні властивості даних постійно змінюються, що призводить до явища «дрейфу концепцій» (concept drift) [89]. Оскільки в потоках телеметрії неможливо миттєво отримати істинні мітки (ground truth) для нових даних, традиційні методи моніторингу помилок стають неефективними. Тому виникає гостра потреба у використанні методів виявлення дрейфу без вчителя (unsupervised drift detection), які здатні працювати в режимі реального часу [8; 50].. Саме тому моніторинговий агент повинен вміти автономно адаптувати свою структуру до цих непередбачуваних змін.

Вагомий внесок у розвиток методів системного аналізу, інтелектуального моніторингу, агентного моделювання, машинного навчання та інженерії MLOps зробили такі вітчизняні та зарубіжні вчені, як Сергієнко І.В., Морозов А.О., Голуб С.В., Снітюк В.Є., Breiman L., Wolpert D.H., Quinlan J.R., Gama J., Sculley D., Zaharia M. Та інші.

Незважаючи на значні досягнення у цих напрямках, залишаються відкритими питання щодо рівня автоматизації процесів побудови складних ансамблів. Відомі

засоби AutoML здебільшого сфокусовані на оптимізації поодиноких моделей, що ускладнює їх застосування для формування багатошарових архітектур зі збагаченням простору ознак. Водночас платформи управління життєвим циклом (MLOps) часто потребують імперативного програмування для налаштування взаємодії між алгоритмами, а традиційні формати серіалізації мають обмеження щодо опису багатокрокових конвеєрів. Це створює перешкоди для впровадження повністю автономних моніторингових агентів.

З огляду на це, актуальною науково-прикладною проблемою є вдосконалення методів багатошарового синтезу моделей та розроблення інструментів їх відтворюваного виконання. Вирішення цієї проблеми сприятиме підвищенню точності прогнозування агентами в умовах неоднорідних даних, дозволить автоматизувати процеси керування архітектурою ансамблів та спростить надійне перенесення навчених моделей у середовище інтелектуального моніторингу.

Зв'язок роботи з науковими програмами, планами, темами.

Робота виконана у відповідності до тематики наукових планів кафедри програмного забезпечення автоматизованих систем Черкаського державного технологічного університету в рамках науково-дослідних робіт «Програмні агенти в інформаційній технології кризового інтелектуального моніторингу» (№ державної реєстрації 0121U113866, 2022-2023 рр.), та «Інтелектуальний моніторинг кібербезпеки корпоративної мережі програмними агентами» (№ державної реєстрації 0125U001347, 2025-2027 р.р.), де автор був виконавцем окремих розділів.

Мета і завдання дослідження

Підвищення ефективності функціонування моніторингового програмного агента у інформаційній технології інтелектуального моніторингу шляхом розробки методів побудови та програмної реалізації багатошарових агентних моделей.

Відповідно до поставленої мети в роботі сформульовані такі завдання:

1. Проаналізувати існуючі методи побудови та програмної реалізації моделей машинного навчання, які можуть бути використані при побудові моніторингових програмних агентів.
2. Розробити метод підвищення точності агентних моделей за рахунок зростання однорідності точок спостереження шляхом використання кластеризації вхідних даних та інтеграції прогнозів спеціалізованих кластерних моделей як додаткових ознак.
3. Удосконалити метод синтезу багатошарових моделей (рециркуляція) шляхом залучення різнотипних алгоритмів машинного навчання для формування структурних шарів.
4. Розвинути метод багатошарового синтезу за рахунок формалізації процесів конструювання алгоритмів на основі спрямованого ациклічного графу (DAG) для забезпечення багатоетапності та детермінізму обчислень.
5. Побудова інформаційної технології проектування програмного забезпечення моніторингових програмних агентів у середовищі інтелектуального моніторингу.

Об’єкт дослідження. Процес функціонування моніторингового програмного агента в умовах динамічного середовища інтелектуального моніторингу.

Предмет дослідження. Методи побудови та засоби програмної реалізації багатошарових агентних моделей.

Наукова новизна отриманих результатів

У процесі виконання завдань отримано наукові результати, що становлять наукову новизну та мають практичне значення.

1. Вперше запропоновано метод багатошарового синтезу агентних моделей, який заключається у формуванні масиву вхідних даних, синтезу моделі, добавленні сигналу із виходу моделі до масиву вхідних даних і повторного синтезу моделі, який, на відміну від існуючих, використовує кластеризацію вхідних даних і побудову для кожного кластеру моделей, що дозволяє підвищити точність і адекватність

результатів моделювання та профілювати метод проектування програмного забезпечення агента за рахунок підвищення однорідності точок спостереження у кластерах.

2. Удосконалено метод синтезу багатошарових моделей та їх програмної реалізації шляхом зростання різноманітності структури шарів, що дозволяє підвищити повноту опису об’єкта та покращити результати моделювання.

3. Набув подальшого розвитку метод багатошарового синтезу агентних моделей за рахунок використання у процесі проєктування алгоритмів синтезу моделей та їх програмного забезпечення спрямованого ациклічного графу (DAG), що дозволяє забезпечити багатоетапність процесу удосконалення структури моделі шляхом обробки сигналів на виході та використання моделей, побудованих у різних середовищах.

Практичне значення отриманих результатів

1. Розроблений метод підвищення однорідності вхідних даних (за рахунок використання кластерних моделей) дозволяє моніторинговому агенту розпізнавати різні режими функціонування віртуального робота та адаптуватися до них. Завдяки збагаченню простору ознак, побудовані моделі краще пристосовуються до зміни характеристик середовища, що дозволяє точніше прогнозувати стан системи.

2. Удосконалений метод рециркуляції з використанням різнотипних алгоритмів на різних рівнях структури підвищує здатність системи моніторингу моделювати складні нелінійні процеси. Таке поєднання дозволяє компенсувати алгоритмічні обмеження окремих класів машинного навчання (наприклад, лінійних моделей або дерев рішень) та ефективніше виявляти приховані закономірності, що підвищує загальну точність прогнозування.

3. Застосування декларативного підходу та подання процесів конструювання у вигляді спрямованого ациклічного графу (DAG) дозволило побудувати цілісну інформаційну технологію проектування моніторингових агентів. Це вирішує прикладну інженерну проблему надійного перенесення навчених

складних ансамблів (із вбудованими етапами кластеризації та рециркуляції) у середовище експлуатації зі збереженням машинної відтворюваності обчислень.

Особистий внесок здобувача. До дисертації увійшли наукові результати, отримані здобувачем особисто. З наукових праць, опублікованих у співавторстві, в дисертації використано лише ті ідеї та положення, які є результатом особистої роботи здобувача.

У публікаціях написаних у співавторстві, здобувачеві належать: [16] – визначення проблеми обробки табличних даних, розгляд проблеми прогнозування часових рядів; [56] – представлено результати удосконаленого методу рециркуляції; [27] – описано метод підвищення однорідності; [53] – представлено декларативний підхід до проектування моніторингових агентів; [55] – представлено метод подання процесів конструювання АСМ на основі спрямованого ациклічного графу; [31, 32, 33] – загальні принципи побудови програмного комплексу моніторингового агенту, [30, 34, 35] – демонструє можливі сфери застосування запропонованих методів, [36, 37] – розгляд перспективних напрямків інтерпретації результатів прогнозних моделей, [38] – загальний огляд поточного стану розвитку технологій штучного інтелекту.

Апробація результатів дисертації. Основні теоретичні та практичні результати досліджень доповідались та обговорювались на наступних конференціях:

- XVII Міжнародна науково-практична конференція «Інтегровані інтелектуальні робото-технічні комплекси (ІІРТК-2024) (Київ, Україна, 2024 р.)
- XIII Міжнародна наукова конференція «Інформація, комунікація, суспільство – 2024» (Львів, Україна, 2024 р.)
- 1 Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT&AIS-2025» (Харків, Україна, 2025 р.)
- Міжнародна науково-практична конференція “Інформаційні технології та комп’ютерне моделювання – 2025” (Івано-Франківськ, Україна, 2025 р.)

Публікації. Результати проведеного дисертаційного дослідження було опубліковано у 14 наукових працях, а саме: 4 у наукових фахових виданнях (із них 4 у співавторстві), 1 стаття (у співавторстві) у періодичному науковому виданні, проіндексованому у базі даних Scopus, 9 тези доповідей (із них 8 у співавторстві) у збірниках матеріалів конференцій.

Структура та обсяг роботи. Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг дисертаційної роботи становить 149 сторінок комп'ютерного тексту, у тому числі: основний текст – 119 сторінок; 8 таблиць та 17 рисунків; 4 додатки обсягом 18 сторінок; список використаних джерел складає 100 найменувань на 12 сторінках

РОЗДІЛ 1
МЕТОДИ ТА ЗАСОБИ ПОБУДОВИ

1.1. Проблематика динамічних змін у середовищі інтелектуального моніторингу

Війна в Україні з російськими загарбниками, техногенна катастрофа з Каховським водосховищем та інші кризові події роблять актуальними дослідження процесів кризового моніторингу.

Використання агентного підходу до програмної реалізації інформаційної технології інтелектуального моніторингу дозволяє будувати інформаційні системи кризового моніторингу на базі мультиагентних систем (МАС). Моніторинговий програмний агент є одним з основних елементів структури МАС подібного типу [54]. Однією з функцій програмних агентів цього типу є перетворення інформації від форми двовимірного масиву результатів спостережень до форми агентної моделі, яка забезпечує функціонування цього програмного агента.

Особливої ваги набуває моніторинг автоматизованих програмних процесів (складних інформаційних конвеєрів, мікросервісів, фонових обробників), які забезпечують функціонування критичної інформаційної інфраструктури. Хоча такі роботи існують у цифровому середовищі, їхня поведінка та потоки даних (телеметрія, логи, транзакції) мають природу, схожу з сигналами фізичних сенсорів: високу частоту надходження, зашумленість та схильність до непередбачуваних змін режимів роботи, що являє собою механізм інтелектуальної обробки даних. Даний формат даних має властивості випадкових процесів – ми не можемо передбачити початково яка буде зміна вхідних змінних.

У контексті даного дослідження під моніторинговим агентом розуміється автономна програмна сутність, яка функціонує у інформаційному середовищі. Ключовими характеристиками агента є автономність, реактивність та цілеспрямованість. Автономність реалізується через наявність програмного компонента — синтезатора моделей, який здатний без втручання людини перебирати

та оцінювати різнорідні алгоритми. Реактивність (адаптивність моделі) проявляється у здатності агента реагувати на гетерогенність та зміну вхідних даних і виконувати її, реалізуючи цим принципи адаптивного моделювання. Для забезпечення цієї властивості набір алгоритмів синтезу моделей (АСМ) розширено комплексними методами конструювання ансамблів: методом підвищення однорідності (через кластеризацію) та багатошаровою рециркуляцією, що є одним з видів ансамблевого стекінгу. Ці методи оперують базовими одиничними моделями машинного навчання як будівельними блоками, формуючи складні ієрархічні структури. Цілеспрямованість агента підпорядкована пошуку оптимальної архітектури ансамблю шляхом мінімізації функції втрат (похибки прогнозування). Управління життєвим циклом агента здійснюється через декларативну специфікацію, що гарантує керованість та детермінізм його поведінки на етапі експлуатації.

З інженерної точки зору, процес моніторингу за допомогою програмного агента суттєво відрізняється від традиційного підходу, де модель машинного навчання навчається один раз і залишається незмінною. У динамічному середовищі, де постійно надходять нові дані, агент повинен функціонувати як безперервний цикл спостереження, аналізу та програмної адаптації [12]. Автономність агента вимагає, щоб весь процес обробки даних — від отримання початкової інформації до видачі кінцевого прогнозу — був об’єднаний у єдину, керовану програмну структуру.

У класичних інформаційних системах моніторингу при зміні характеру вхідних даних (наприклад, при зміні режиму роботи системи) інженер має вручну переглядати налаштування алгоритмів та перенавчати моделі [48]. Натомість моніторинговий програмний агент повинен мати вбудовані механізми самостійного відновлення своєї точності. Його реактивність полягає не лише у зменшенні похибки прогнозування на історичних даних, а й у постійному пошуку такої ієрархічної структури алгоритмів, яка забезпечить максимальну стабільність прогнозу в майбутньому. Агент самостійно ухвалюватиме рішення щодо оптимальної кількості шарів моделей та стратегії групування вхідних даних.

Методи аналізу стану, що традиційно застосовується для діагностики фізичного обладнання, є релевантними і для програмних середовищ, оскільки останні також характеризуються показниками «здоров'я» (health metrics), споживанням ресурсів та відмовами. Застосування алгоритмів машинного навчання та нейронних мереж дозволяє виявляти приховані залежності у великих масивах апаратних та програмних логів, згенерованих у процесі функціонування розподілених систем. До таких задач відносяться визначення рівня відмовостійкості, детекція аномалій трафіку та прогнозування витрат ресурсів.

Практичний досвід показує, що жорстко запрограмовані системи автоматизації є вкрай чутливими до змін у програмному середовищі [1]. Щоб подолати цю крихкість, сучасні дослідження пропонують концепцію «самовідновлюваних» (self-healing) систем [66]. Вони використовують машинне навчання для безперервного аналізу телеметрії, що дозволяє автономно виявляти збої та коригувати виконання процесів [18]. Крім того, автоматизовані агенти все частіше застосовуються для наскрізного моніторингу великих медичних або фінансових баз даних, де вони імітують дії користувачів та допомагають виявляти приховані мережеві затримки [17]. Це генерує великі масиви складних даних, для обробки яких потрібні багатошарові предиктивні моделі агента.

Таким чином, задача моніторингу інформаційної системи, попри динамічну природу надходження даних, на рівні моделювання часто зводиться до задачі регресії або класифікації поточного стану системи на основі миттєвого зрізу параметрів (snapshot). Такі дані зручно представляти у формі структурованих таблиць, де кожен рядок відображає стан системи в певний момент. Специфіка цих даних (суміш числових метрик та категоріальних статусів, висока неоднорідність) накладає обмеження на вибір методів моделювання.

Наразі тривають дебати щодо кращих методів вирішення задачі прогнозування – статистичних чи тих, що використовують підходи машинного навчання. У сучасній науковій літературі зокрема розглядаються алгоритми машинного навчання, які

мають певні переваги над статистичними, такі як менша залежність від лінійних залежностей [41] між елементами вхідних даних та можливість застосування без поглибленого аналізу предметної області.

У задачах моніторингу табличні дані залишаються переважаючими в дослідженнях, являючи собою структуровану інформацію у форматі рядків і стовпчиків, що надає можливість описати взаємозв'язки між різними ознаками між собою. Поширеність даного формату зумовлена спорідненістю даного формату до процесу збору та зберігання інформації різними організаціями, наприклад, формат збору даних пацієнтів, даних сенсорів та характеристик поведінки споживачів [40]. Обробка табличних даних має ряд особливостей порівняно з іншими форматами даних, серед яких можна виділити візуальну або текстову інформацію. На противагу зазначеним форматам даних, де глибоке навчання досягло суттєвих здобутків, у сфері обробки табличних даних алгоритми даної категорії зіткнулися з рядом перепон, що обмежує ширину застосування цих підходів у даній сфері. Останні дослідження [9] демонструють, що традиційні методи, засновані на деревах рішень, часто показують кращі результати, ніж комплексні архітектури, засновані на нейронних мережах, що показує особливість даної предметної області. Ці складності виникають через особливості будови табличних даних, а саме неоднорідний характер даних, що поєднує числові та категоріальні ознаки, складні та часто нелінійні зв'язки між ознаками і часту можливість наявності пустих даних у масиві вхідних даних, що вимагає різноманітних алгоритмів передобробки даних [65].

Поточні підходи до обробки табличних даних найбільше використовують алгоритми, які застосовують підходи дерев прийняття рішень та їхні різноманітні поєднання, серед яких можна виділити певні рішення щодо лісів рішень, як-от випадкові ліси (Random Forests)[2] та алгоритми градієнтного покращення[42]. Алгоритми глибокого навчання не показують найкращих результатів у роботі з табличними даними через те, що немає можливості виявляти приховані внутрішні зв'язки у даних, так як цей тип даних вже має свою наперед визначену структуру.

Водночас, існуючі ефективні методи (дерева рішень, бустинг) зазвичай формують статичні моделі, які не здатні автоматично адаптуватися до зміни режимів функціонування інформаційної системи без повного перенавчання, що є критичним обмеженням для автономних моніторингових агентів.

1.2. Аналіз методів прогнозування

1.2.1. Алгоритми прогнозування часових рядів

Одним з ключових аспектів методу побудови моніторингових систем є особливості агентного синтезатора моделей, а саме: якими алгоритмами він оперує, які параметри архітектури моделі певної топології можна налаштувати, та що саме являє собою процес створення конкретного екземпляра певної моделі, процес його навчання на масиві вхідних даних та оцінки моделі за різними показниками.

Оскільки результати моніторингу часто мають часову складову, традиційний підхід до їх аналізу базується на алгоритмах прогнозування часових рядів. Розглянемо, які алгоритми існують для вирішення задачі прогнозування часових рядів, які їхні переваги та недоліки. Для вирішення цієї задачі існує ряд алгоритмів, які мають різні структури та реалізують різні підходи, серед яких можна виділити:

1. Алгоритми, засновані на методах статистичного аналізу (AR, MA, ARMA, ARIMA, SARMA, SARIMA), ETS, CES, Theta;

2. Алгоритми, засновані на використанні нейронних мереж.

Для повноти дослідження необхідно проаналізувати сучасні архітектури нейронних мереж, які демонструють високу здатність розпізнавати складні часові залежності. Це дозволить оцінити їхні переваги та обмеження в контексті застосування всередині автономного моніторингового агента.

Розглянемо специфічні топології нейронних мереж, які аналізувалися на початкових етапах проектування синтезатора моделей. Синтезатор створює набір алгоритмів синтезу моделей (АСМ) на основі цих архітектур, кожен з яких має унікальний набір характеристик, що описують його кінцеву структуру. Розглянемо

певні архітектури нейронних мереж, що можуть бути використанні для вирішення даної задачі.

1.2.1.1. Алгоритми на основі багатошарового перцептрона.

Багатошаровий перцептрон – це топологія нейронної мережі, що складається з повністю зв’язаних шарів і включає щонайменше три шари: вхідний шар, прихований шар і вихідний шар. Також може бути кілька прихованих шарів для ідентифікації складних залежностей у наборі вхідних даних. Ця топологія нейронної мережі може вирішити задачу прогнозування, використовуючи кінцевий вихідний шар з кількістю нейронів, встановленою як $P * N$. Тут P представляє горизонт прогнозування, а N – кількість ознак у екземплярі вхідних даних.

Розглянемо інші спеціалізовані архітектури нейронних мереж, засновані на використанні багатошарових персептронів.

N-BEATS/N-BEATSx. У роботі [91] представлено архітектуру нейронної мережі, щоб продемонструвати, що алгоритм на основі глибокого навчання може перевершити стандартні статистичні методи. Алгоритм розбиває ділянку історичних даних на інтервали, що відповідають довжині горизонту прогнозування. Для кожного заданого періоду блок будується шляхом створення двох послідовностей повністю зв’язаних шарів (багатошаровий персептрон), перший з яких навчається прогнозувати попередній період історичних даних, а другий – прогнозувати наступний період. Ці блоки послідовно об’єднуються в стек, а отримані стеки також формують послідовність для підвищення точності прогнозування. Варто зазначити, що послідовності блоків і стеків формуються послідовно з додаванням залишкових ланок у певному форматі, як запропоновано авторами архітектури.

Вдосконалена версія цієї архітектури, відома як N-BEATSx, описана в [61] є модифікацією моделі N-Beats[91], яка дозволяє навчати модель не лише на цільовому часовому ряді, але й на пов’язаних (екзогенних) змінних часових рядах. Структурні компоненти моделі залишилися незмінними, але підхід до генерації вхідних даних за допомогою використання часових згорткових нейронних мереж був змінений [90].

N-HiTS. Робота [79] представляє дану архітектуру, що є модифікацією архітектури N-BEATS [91]. Основні зміни, що були запропоновані, це метод попередньої обробки вхідних даних шляхом пропускання через шар об’єднання для зменшення розмірності вхідних даних для виявлення більш загальних властивостей часової послідовності, використання регресійних моделей для генерації кінцевого виходу блоку прогнозування майбутнього, «минулого», метод навчання на послідовностях різного розміру з різними параметрами шару об’єднання та формування результуючого прогнозу шляхом інтерполяції на основі важливості кожної послідовності з різними частотами.

1.2.1.2. Алгоритми, засновані на рекурентних нейронних мережах.

Основною характеристикою рекурентних нейронних мереж є їхня здатність обробляти послідовні дані та запам’ятовувати минулі кроки [71], що робить їх придатними для прогнозування часових рядів. На цих мережах базуються різні алгоритми, які будуть розглянуті в наступних розділах.

Стандартна рекурентна нейронна мережа. Цей тип мережі використовується для прогнозування часових рядів відповідно до архітектури «кодер-декодер», де початкова послідовність спочатку перетворюється кодером у певний проміжний стан, а потім декодер перетворює початковий стан у результуючу послідовність, яка визначає прогноз моделі.

У стандартній рекурентній мережі, що використовується для прогнозування часових рядів, вона діє як кодер, основним завданням якого є підготовка проміжного стану для декодера. У кодері може існувати кілька рекурентних рівнів для визначення глибоких залежностей між елементами вхідного набору даних. У цій архітектурі декодер є багатошаровим перцептроном, який навчений перетворювати проміжний етап у результуючу послідовність.

Кожна частина стандартної рекурентної нейронної мережі має на вході вихід попереднього елемента та поточний елемент вхідних даних, які використовуються для генерації виходу компонента.

У випадку кількох рекурентних рівнів, компоненти вищих рівнів отримують вихід відповідного блоку попереднього рівня, а не поточного вхідного елемента.

Цей тип мережі може ігнорувати ознаки послідовності, виявлені на попередніх етапах, що призводить до потенційних проблем зі швидкістю навчання та досягненням прийнятного рівня метрик помилок. Таким чином, рекурентні нейронні мережі використовують спеціалізовані структури для рекурентних одиниць, двома основними прикладами яких є довга короткочасна пам’ять (LSTM) та керована рекурентна одиниця (GRU).

Рекурентна нейронна мережа на основі LSTM. Основною відмінністю між структурою довгої короткочасної пам’яті як блоку рекурентної нейронної мережі є наявність механізму вентилів (забуття, входу, виходу), що дозволяє контролювати потік інформації та уникати проблеми зникаючого градієнта на довгих послідовностях [51].

Рекурентна нейронна мережа, що використовує вентильований рекурентний блок (GRU). Вентильований рекурентний блок (GRU) – це тип структури блоку рекурентної нейронної мережі (RNN). Це спрощена версія блоку довгої короткочасної пам’яті (LSTM) і має лише два вентилі у своїй конструкції: вентиль оновлення та вентиль скидання [81].

Характеристики цих специфікацій рекурентних нейронних мереж такі ж, як і у стандартної версії, зі змінами лише у структурі її основних блоків.

В останні роки також були досліджені та запропоновані інші архітектури рекурентних мереж, які можна використовувати для вирішення задачі прогнозування. Нижче наведено огляд архітектур, доступних у поточному синтезаторі моделей.

Dilated RNN. Цю архітектуру нейронної мережі було запропоновано в дослідженні [80] для оптимізації продуктивності рекурентної нейронної мережі для ідентифікації зв’язків у довгих послідовностях та пришвидшення процесу навчання. Основна ідея цієї архітектури полягає в додаванні рекурентних проміжків зв’язків на кожному рівні рекурентної нейронної мережі (енкодера). Ця архітектура дозволяє

використовувати різні типи рекурентних одиниць як її компоненти, а саме довгу короткочасну пам’ять (LSTM) або керовану рекурентну одиницю (GRU).

1.2.1.3. Алгоритми, засновані на згорткових нейронних мережах.

Згорткові нейронні мережі часто використовуються в обробці зображень, використовуючи двовимірні фільтри для ідентифікації ознак об’єктів на зображеннях. Для застосування цієї методики в області обробки часових рядів використовуються одновимірні згорткові нейронні мережі[41]. Одновимірна згорткова нейронна мережа використовується як кодер для перетворення елементів масиву вхідних даних у проміжний стан, який потім використовується декодером, який є багатошаровим перцептроном, для перетворення в результуючу прогнозовану послідовність. Синтезатор моделей використовує спеціалізовані архітектури нейронних мереж цього типу.

Часова згорткова мережа (TCN). Основний підхід, запропонований у дослідженні[90], полягає у використанні кількох згорткових шарів для виявлення ознак у вхідній послідовності, що подається на вхід кодера. Кожен наступний згортковий шар пов’язаний з попереднім, але використовується підхід дилатацій, щоб зменшити можливість перенавчання та виявити залежності між несумісними елементами вхідної послідовності.

Структура декодера для цього типу мережі подібна до тієї, що використовується в алгоритмах, що використовують рекурентні нейронні мережі.

1.2.1.4. Алгоритми, що використовують архітектуру «трансформера».

Головною особливістю алгоритмів, що використовують архітектуру «перетворювача», є використання принципу «уваги».

Принцип «уваги» був представлений у [87], де було запропоновано формувати вектори важливості елементів прихованого стану, що є спільним між кодером та декодером. Розглянемо архітектури, що використовують цей підхід, що використовуються в синтезаторі моделей.

Трансформер. Ця архітектура нейронної мережі була представлена в [94], де пропонується використовувати підхід «уваги» замість рекурентних або згорткових нейронних мереж для кодера та декодера. Основним коригуванням використання стандартної концепції «уваги» було включення багатоголової «уваги», яка створює кілька векторів важливості для кожного елемента у вхідній частині даних, а потім обчислює зважену суму між ними для визначення результату. Вектори важливості спрямовуються у вхідний шар багатошарового персептрона, який ідентифікує їх взаємозалежності. Декодер у цій архітектурі має два блоки обчислення уваги. Перший блок використовує інформацію виключно з вихідної послідовності, тоді як другий блок використовує як вектори важливості для елементів вихідної послідовності, так і вихід кодера. Після цього результат обчислень уваги обробляється багатошаровим перцептроном для генерації виходу декодера. Однією з головних переваг використання цього принципу є потенціал для паралельного навчання, оскільки ця структура не вимагає впорядкованої обробки вхідної послідовності.

Перетворювач часового злиття (TFT). Це архітектурне рішення було представлено в [59] для вирішення проблеми роботи з часовими рядами шляхом ефективного використання інформації про часовий інтервал, до якого він належить, а також інформації про статичні показники, відомі в минулому та відомі в майбутньому (такі як день тижня, день місяця тощо). Кодер і декодер використовують рекурентну нейронну мережу з довгою короткочасною пам’яттю як тип рекурентної одиниці. Вихідні дані з рекурентних блоків як у кодері, так і в декодері обробляються за допомогою багатошарової перцептронної архітектури із керованими залишковими з’єднаннями між повністю зв’язаними шарами, також відомої як керована залишкова мережа (GRN). Вихідні дані цього етапу обробки потім надсилаються на багатоголові шари обчислення уваги, які, у свою чергу, подаються на замкнені залишкові мережі, що використовуються для генерації кінцевого результату мережі.

Informer. Архітектура, представлена в дослідженні [99], спрямована на оптимізацію структури стандартного трансформера [94], щоб скоротити час навчання

мережі та підвищити точність прогнозування довгих часових рядів. Щоб зменшити як час навчання, так і використання пам’яті, структуру кодера було змінено для використання трьох блоків, що складаються з одновимірного згорткового шару та блоку для обчислення «уваги» та шару об’єднання, з’єднаних послідовно. Це дозволяє алгоритму обчислювати показники уваги на менших розмірах даних після проходження через згортковий шар, що важливо для зменшення складності алгоритму під час обробки довгих послідовностей. Декодер дотримується звичайної структури стандартного трансформера, за винятком того, що він приймає вхідну послідовність у поєднанні з заповнювачем, а не результуючу послідовність. Ця модифікація дозволяє декодеру функціонувати у форматі генеративного прогнозування, що застосовується до завдань обробки природної мови (NLP).

Autoformer. Архітектура, представлена в дослідженні [96], спрямована на зменшення обчислювальної складності моделі, подібно до проблеми, яку вирішує архітектура Informer [99]. Крім того, вона оптимізує прогнозування довгих часових рядів порівняно зі стандартним трансформером [94]. Основний підхід, що використовується в цій моделі, передбачає включення блоку для розкладання часового ряду на трендову та сезонну складові за допомогою методу ковзного середнього. Потім цей блок передає як часовий ряд, так і його розкладені результати на вхід кодера та декодера. Іншим аспектом структури цього алгоритму є використання принципу автокореляції. Це відрізняється від принципу «уваги», оскільки він в першу чергу зосереджений на кореляціях між компонентами в межах одного періоду часу, а не між усіма елементами послідовності.

FedFormer. Ця версія архітектури трансформера [94], як описано в [100], пропонує новий підхід до розбиття часових рядів для подальшої обробки кодером та декодером. Зокрема, вона використовує частотне розкладання, яке може бути виконане різними методами в модулі частотного розкладання, включаючи дискретне перетворення Фур’є або дискретне вейвлет-перетворення. Частотні дані з часового ряду використовуються в блоці обчислення «уваги» для виявлення залежностей між

частотними параметрами вхідних даних та часовими параметрами. Іншим компонентом цієї архітектури є блок виявлення тренду для часового ряду в різних частинах, результат якого також використовується для формування векторів важливості для блоку обчислення «уваги».

Сучасні методи прогнозування часових рядів на основі нейронних мереж є ефективними за умови стабільної періодичності даних. Проте їхнє використання в автономних системах моніторингу має певні інженерні обмеження:

• Залежність від безперервної історії. Алгоритми прогнозування часових рядів очікують даних у вигляді безперервних часових послідовностей. В умовах реальних інформаційних систем метрики часто надходять асинхронно або з пропусками, що порушує логіку роботи таких моделей.

• Обчислювальна складність. Навчання глибоких нейромереж є ресурсоємним. Це ускладнює їх реактивну адаптацію агентом.

Через вплив даних обмежень, у даній роботі надалі розглядається інша задача – задачу відновлення функціональної залежності (регресії), до якої можна звести характер роботи агента, якщо розглядати, що він працює не з періодичними даними, а розглядає зрізи стану системи (snapshots). У наступному підрозділі розглядаються алгоритми машинного навчання на основі дерев рішень, бустингу та регуляризації для вирішення задачі регресії. Також розглянуті певні спеціалізовані архітектури нейронних мереж, що розроблені для вирішення задачі регресії на табличних даних

1.2.2. Аналіз методів машинного навчання для вирішення задачі регресії

Важливим елементом синтезатора моделей є варіативність АСМ, наявних у ньому. Для вирішення задачі регресії на табличних даних розглянуті такі групи алгоритмів.

1.2.2.1. Алгоритми, що використовують принципи дерев рішень

Дерева рішень у своїй основі використовують принципи поділу вхідного простору на частини, застосовуючи різні показники, як-от Gini [46], для задач

класифікації, а також середньоквадратичну похибку для задачі регресії. Фундаментальні засади для цих методів були закладені у класичних працях з машинного навчання. Зокрема, Дж. Квінлан (J. R. Quinlan) формалізував використання дерев рішень та розробив алгоритми ID3 і C4.5, які будують гілки на основі теорії інформації та критеріїв ентропії [20]. Випадкові ліси рішень покращують цей алгоритм шляхом поєднання результатів використання кількох дерев рішень між собою, а також випадковими вибірками різних ознак, що дозволяє зменшити дисперсію значень до 40 %, порівнюючи до використання простого дерева рішень [2]. Екстра-дерева (ExtraTree) зменшують кореляцію різних дерев між собою шляхом використання підходу до введення випадкових порогових значень, вибраних рівномірно з діапазонів ознак, зберігаючи прийом випадкового поділу ознак, що використовується у випадкових лісах рішень, демонструючи подальше зменшення модельної дисперсії значень [6].

У минулому це сімейство алгоритмів було прийнято відносити до сімейства алгоритмів «чорного ящика», так як були складності в аналітичному поясненні результатів тренування моделі, проте у останні роки було запропоновано декілька рішень, що можуть підвищити можливість аналізу моделей даного типу, як-от SHAP [85] (в рамках ширшої концепції Explainable AI [4]), що дозволяє аналізувати модель за впливом різних ознак на результат прогнозування та є одним із важливих компонентів для проектування пояснюваних агентів [68]. Також варто зазначити, що активно пропонуються подальші покращення наявних алгоритмів і підходів, серед яких варто відзначити модель глибокого лісу (DeepForest), що демонструє можливість поєднання моделі дерева рішень із нейронними мережами шляхом заміни типових елементів дерева рішень нейронними мережами [24].

1.2.2.2. Алгоритми, засновані на принципі градієнтного покращення

Алгоритми, засновані на принципі градієнтного покращення, тренуються на масиві вхідних даних, послідовно поєднуючи результати «слабких» моделей

(наприклад, дерев рішень) між собою для зменшення рівня помилки отриманої на попередньому етапі тренування. На кожному кроці нова «слабка» модель тренується для передбачення градієнта функції втрат для поступового зменшення втрат у процесі тренування. Існує ряд найбільш використовуваних алгоритмів даних, серед яких XGBoost [42], що використовує технології L1/L2 регуляризації і поділу даних із використанням характеристик розподілу (формат гістограми даних), LightGBM [58], що застосовує спеціалізовану схему поділу GOSS(Gradient-Based One-Side Sampling) для роботи з великими масивами даних шляхом вилучення даних з значними значеннями градієнтів та CatBoost [92], що використовує порядкове підсилення для мінімізації перенавчання. Дане сімейство алгоритмів показує одні з найкращих показників на табличних масивах вхідних даних, демонструючи кращі показники метрик на масивах вхідних даних середнього розміру порівняно з моделями, що використовують принципи глибокого навчання [40]. Значним недоліком алгоритмів даного роду є їх послідовна манера тренування, бо моделі збільшують число своїх компонентів із кожним наступним кроком, що зменшує можливості щодо паралельного тренування їх, а це може створювати проблеми при роботі з великими масивами вхідних даних. Утім, паралельні обчислення можливі для навчання одиночних дерев рішень, що є саме тими «слабкими» моделями. Варто зазначити, що нові дослідження у даній сфері продовжують з’являтися, серед яких варто виділити такі роботи, як NGBoost [83], що використовує ці алгоритми для формування ймовірнісних передбачень, та роботи, що пропонують методи для тренування та використання даної архітектури, здійснюючи виконання на графічних прискорювачах [98], основною метою якого є вирішення недоліку у необхідності навчати модель послідовно, що було попередньо зазначено як один із недоліків алгоритмів даного типу.

1.2.2.3. Алгоритми, що використовують нейронні мережі

Алгоритми, що використовують нейронні мережі, демонструють позитивні результати в обробці однорідних даних, як-от зображення та тексти, для якої можуть

бути підготовлені великі масиви вхідних даних. У своїй основі нейронні мережі складаються зі штучних нейронів, що обробляють дані, які обробляють дані за допомогою зважених зв'язків, застосовуючи різноманітні нелінійні функції активації для запам'ятовування складних шаблонів. Ці мережі навчаються завдяки зменшенню функції втрат при кількісних зворотних проходах (backpropagation) від отриманого результату до початкових вхідних даних. Проте даний алгоритм має недоліки у випадку з табличними даними через те, що ознаки в даних масивах даних є неоднорідними і їх відносна величина є невеликою. Якщо розглянути різницю між текстовими або візуальними даними (зображеннями) та табличними, то однією з основних відмінностей є характер зв'язків між ознаками — послідовні або просторові зв'язки у даних першого типу і неоднорідні типи і комплексні зв'язки у табличних даних. Але останні дослідження наводять спеціалізовані алгоритми, що намагаються вирішити ці проблеми при використанні алгоритмів даного типу. Наприклад, TabNet реалізовує архітектуру «перетворювач», але застосовує структуру вибору ознак, схожу до підходу дерев рішень, використовуючи процес послідовної уваги [78]. Модель використовує принцип уваги для вибору ознак на кожному етапі прийняття рішень, забезпечуючи цим можливість обчислення впливовості ознаки динамічно та дозволяючи інтерпретувати результати за домопогою аналізу вагових коефіцієнтів уваги.

Інші запропоновані технології намагаються поєднати нейронні мережі з типовими алгоритмами машинного навчання. Наприклад, NCART [86] інтегрує нейронні мережі в комірки дерева рішень, дозволяючи розширити складність рішень, але зберігаючи структуру, яку можна аналітично аналізувати. Насамперед алгоритм моделі DNDT [97] являє собою перетворення традиційних алгоритмів дерев рішень у диференційовані рівні нейронної мережі шляхом реалізації алгоритмів м'якого вибору, замінюючи стандартний алгоритм строгих рішень (правил), що дозволяє в результаті використовувати принцип зменшення градієнта. NET-DNF [13] інтегрує використання математичних правил кон'юнкції і диз'юнкції для визначення

логічних зв’язків між ознаками з подальшою інтеграцією даних правил у структуру нейронної мережі. Ці новітні розробки дозволяють поєднувати переваги нейронних мереж і особливості структурованих даних, проте все ще програють у показниках більш простим моделям класичного машинного навчання, особливо на масивах вхідних даних з обмеженою кількістю даних. Також важливим недоліком є швидкість навчання порівняно з отриманими результати, де традиційні методи все ще мають перевагу.

1.2.2.4. Алгоритми, що засновані на принципі регуляризації

Алгоритми, засновані на принципі регуляризації, являють собою сімейство лінійних методів, головною метою яких є вирішення проблеми перенавчання в задачах регресії. Ці алгоритми ускладнюють процес роботи стандартних лінійних алгоритмів шляхом додавання умови для «штрафів», що надає можливість контролювати коефіцієнти моделей, балансуючи між складністю і точністю моделі [49]. Основним принципом даних моделей є додавання правил регуляризації до стандартної функції втрат, що дозволяє вибирати найбільш впливові ознаки та покращувати модельні прогнози.

Основний алгоритм Ridge (L2 — регуляризація) полягає в обчисленні квадратного коефіцієнта магнітуди для параметрів моделі і дозволяє зменшувати їх значення, близькі до нуля, але не вилучаючи їх повністю. Цей метод можна застосувати для обробки даних із взаємозв’язками між різними незалежними ознаками, так як це дозволяє розподілити коефіцієнти їх впливовості між собою, не вибираючи лише певні з них [52].

Інший алгоритм регуляризації — Lasso (Least Absolute Shrinkage and Selection Operator) реалізує L1 — регуляризацію, що визначає правило «штрафу», яке пропорційне до абсолютних значень коефіцієнтів, що на противагу алгоритму Ridge може повністю вилучити ознаку під час процесу навчання і дозволяє зменшити модель, залишаючи лише найбільш впливові ознаки, що є важливим аспектом при роботі з масивами вхідних даних із великою кількістю ознак [29].

Алгоритм Elastic Net поєднує між собою L1 та L2 регуляризацію, пропонуючи гібридне рішення, що поєднує переваги алгоритмів Ridge і Lasso [25]. Цей метод є ефективним при обробці масивів даних із великою кількістю ознак, певна кількість яких є взаємопов'язана, так як надає можливість згрупувати схожі ознаки, проте вилучаючи найменш впливові.

Хоча дані алгоритми є прості відносно аналізу і демонструють гарні показники щодо обчислювальної складності, їхні результати можуть бути обмеженими через лінійну архітектуру. Незважаючи на недоліки, їх можливо застосовувати як стійкі базові моделі для порівняння більш комплексних моделей з ними, а також розуміння впливовості ознак.

1.2.3. Параметрична оптимізація процесу синтезу моделей

Параметрична оптимізація передбачає визначення набору параметрів моделі та підготовку характерного пошукового простору, за яким відбувається підбір параметрів. Найпростішим варіантом параметричної оптимізації є використання пошуку за сіткою параметрів, при якому кожна комбінація параметрів перевіряється окремо. У даного методу серйозним недоліком є вимога до великих обчислювальних ресурсів, так як поєднань різних комбінацій параметрів моделі може бути значна кількість і для кожної з даних комбінацій повинен тренуватися окремий екземпляр моделі. Наразі набувають популярності методи оптимізації параметрів моделей, які використовують внутрішні моделі для проходження пошукового простору, що дозволяє зменшити кількість ітерацій, необхідних для підбору параметрів моделі, серед яких можна виділити рішення Optuna [77] із внутрішніми моделями оптимізації, як-от TPE [95], CMA-ES [84] , NDGA-II [39].

У [29] описаний процес адаптивного функціонування синтезатора моделей шляхом випробування ACM. У [26] описано кілька стратегій вибору кращого ACM і запропоновано керуватись стратегією оптимальності.

Таким чином, ефективний синтез моделей вимагає не лише вибору алгоритму, а й автоматизованого підбору гіперпараметрів. Сучасні байєсівські методи (TPE)

дозволяють прискорити цей процес порівняно з перебором по сітці, що є критичним для роботи агента в реальному часі.

Вибір фреймворку Optuna та алгоритму Tree-structured Parzen Estimator (TPE) зумовлений суворими архітектурними обмеженнями, які накладаються на автономних моніторингових агентів. У класичному машинному навчанні перебір по сітці (Grid Search) передбачає вичерпне обчислення кожної можливої комбінації параметрів, що призводить до експоненційного зростання обчислювальної складності
. Для багатокрокового процесу конструювання ACM, який включає попередню кластеризацію, навчання локальних моделей та фінального ансамблю, такий підхід є алгоритмічно непрактичним.

На відміну від повного перебору, байєсівська оптимізація на базі TPE застосовує ймовірнісний підхід до дослідження простору пошуку. Алгоритм формує внутрішні генеративні моделі розподілу гіперпараметрів, розділяючи результати попередніх ітерацій на групи «вдалих» та «невдалих» конфігурацій. На кожному наступному кроці система обчислює очікуване покращення (Expected Improvement) і тестує лише ті комбінації параметрів, які з найвищою ймовірністю призведуть до мінімізації функції втрат.

З інженерної точки зору, інтеграція такого методу до складу моніторингового агента вирішує проблему обмеженості апаратних ресурсів у середовищі інформаційної системи. Це гарантує, що процес адаптивного перенавчання моделей (при зміні властивостей вхідного потоку даних) матиме контрольовану та передбачувану тривалість. Замість витрачання обчислювальних потужностей на оцінку завідомо неефективних конфігурацій, агент цілеспрямовано звужує простір пошуку. Отже, автоматизація підбору гіперпараметрів за допомогою алгоритмів класу TPE виступає критично важливим архітектурним компонентом, що забезпечує здатність системи своєчасно та ресурсоефективно оновлювати власну прогностичну структуру.

Проведений аналіз показав, що існує широкий спектр методів прогнозування, кожен з яких має свої переваги. Алгоритми глибокого навчання (Transformer, LSTM) ефективні для складних часових залежностей, але ресурсоємні. Ансамблеві методи на основі дерев (XGBoost, Random Forest) є стандартом для табличних даних, але мають фіксовану структуру. Лінійні моделі з регуляризацією забезпечують стабільність, але не вловлюють нелінійності.
Кожен із розглянутих методів має свої переваги та недоліки щодо застосування. Це обґрунтовує необхідність розробки адаптивного синтезатора та спеціалізованих багатошарових методів, здатних комбінувати переваги різних класів алгоритмів залежно від властивостей масиву вхідних даних.

1.3. Проблеми MLOps: відтворюваність, форми представлення та обмеження форматів серіалізації

1.3.1. Проблема відтворюваності у наукових дослідженнях з МН

Однією з проблем сучасної науки про дані є відтворюваність результатів. Як зазначається у статті [11], багато висновків у роботах зі штучного інтелекту важко або неможливо повторити. Часто це пов’язано не лише з відсутністю доступу до коду чи даних, а й з неповним описом умов експерименту: точних версій програмних бібліотек, усіх гіперпараметрів та неформальних налаштувань, які використовували дослідники. Така ситуація створює «кризу відтворюваності», що знижує довіру до результатів та гальмує науковий прогрес. Це підкреслює потребу в інструментах, які б забезпечували повне та прозоре документування всіх етапів експерименту.

Системи управління життєвим циклом МН (MLOps). Для вирішення проблеми відтворюваності та управління експериментами розроблено низку платформ у рамках концепції MLOps. Такі інструменти, як Mlflow [74] та Kubeflow [72], надають потужні засоби автоматизації [88]. Mlflow дозволяє відстежувати параметри та метрики, керувати версіями моделей та розгортати їх. Kubeflow, у свою чергу, є більш комплексною платформою для оркестрації багатоетапних процесів навчання у середовищі Kubernetes.

Проте ці системи вимагають значних зусиль для налаштування, а логіка експериментів у них описується здебільшого імперативно — через написання об’ємного програмного коду. Вони не надають вбудованих декларативних засобів для опису складних кастомних архітектур (таких як багатошарова рециркуляція з попередньою кластеризацією), що є критичним для систем інтелектуального моніторингу.

Автоматизоване машинне навчання (AutoML). Системи AutoML, такі як Auto-sklearn [47] або TPOT, йдуть ще далі в автоматизації, намагаючись самостійно знайти оптимальну модель та її гіперпараметри для заданого набору даних. Вони є корисними для швидкого отримання базових рішень.

Незважаючи на свою потужність, їхнім головним недоліком часто є підхід «чорної скриньки»: дослідник має обмежений контроль над кінцевою архітектурою моделі [22]. AutoML-системи не призначені для *декларативного проектування*, де людина свідомо конструює складну модельну структуру (наприклад, багатошаровий ансамбль з певними моделями на кожному шарі), а система лише забезпечує надійне та відтворюване виконання цього проекту.

Варто детальніше розглянути обмеження існуючих систем автоматизованого машинного навчання (AutoML) у контексті завдань моніторингового агента. Більшість традиційних AutoML-рішень орієнтовані на пошук найкращих параметрів для однієї окремої моделі або для стандартного набору моделей, які працюють на одному рівні [19]. Проте вони не мають вбудованих механізмів для автоматизації складних, багатоетапних процесів, де, наприклад, результати попереднього групування даних (кластеризації) повинні передаватися як нові ознаки для наступного шару спеціалізованих моделей. Спроба реалізувати таку багатокрокову логіку через існуючі інструменти AutoML часто призводить до втрати контролю над тим, як саме дані передаються між різними рівнями системи.

З іншого боку, класичні інструменти управління машинним навчанням (MLOps), такі як Mlflow, також мають суттєві обмеження при роботі зі складними

структурами. Зазвичай розробники створюють багатошарові моделі шляхом написання значного обсягу програмного коду, який жорстко фіксує послідовність виклику алгоритмів. Коли таку систему потрібно перенести з етапу розробки на етап реальної експлуатації, виникає проблема відтворюваності. Стандартні методи збереження навчених моделей (наприклад, у файли форматів .pkl або .joblib) дозволяють зберегти лише внутрішні параметри самої моделі. Однак при цьому втрачається інформація про загальну логіку процесу: в якій послідовності ці моделі мають запускатися, як вони обмінюються результатами та як обробляються вхідні дані перед подачею в кожну окрему модель. Це створює необхідність у розробці нового підходу, який дозволив би описувати всю архітектуру багатошарового ансамблю у вигляді чіткої конфігурації, незалежної від конкретного програмного коду [7].

Декларативні підходи та DSL в машинному навчанні. Ідея використання декларативного підходу до специфікації для спрощення складних завдань не є новою [3]. Такі інструменти, як TensorFlow (через Keras) [76] або PyTorch [63], також використовують декларативні елементи для опису архітектури нейронних мереж. Дослідник описує шари мережі та зв'язки між ними, а фреймворк сам піклується про обчислення та оптимізацію.

Існуючі декларативні підходи, у свою чергу, в основному зосереджені на описі *архітектури однієї моделі* (переважно нейронної мережі) [23]. Вони не охоплюють весь життєвий цикл експерименту: від передобробки до вибору з пулу різнорідних ACM та автоматичної генерації повного, відтворюваного пакета з усіма артефактами [45]. Таким чином, існує невирішена задача у вигляді відсутності легкого, гнучкого та водночас строгого інструменту для декларативного проектування всього експерименту зі складними, кастомними ансамблевими архітектурами, що і є основним фокусом даної роботи.

1.3.2. Особливості існуючих засобів управління обчисленнями

Спрямований ациклічний граф, фундаментальна структура даних в комп'ютерних науках, є одним з найкращих способів представлення процесів з

односпрямованими залежностями та операціями, що виконуються в певному порядку. Операція або крок обробки представлений набором вершин (вузлів) у такому графі, тоді як потік даних або керування між операціями представлений спрямованими ребрами (стрілками). Основною характеристикою є ациклічність, яка вказує на неможливість повернутися до вузла, який вже був відвіданий, під час навігації вздовж стрілок. Ця особливість гарантує, що процес має чіткий початок і кінець, позбавлений нескінченних циклів і дозволяє дотримуватися суворого набору кроків у кожному виконанні.

У галузі обробки даних та машинного навчання DAG стали де-факто стандартом для опису складних послідовностей обробки даних (data pipelines) [60]. Їхня популярність зумовлена здатністю декларативно описувати складні процеси, що складаються з багатьох взаємозалежних кроків. Провідні інструменти для розподілених обчислень, такі як Apache Spark [75] та Dask [67] активно використовують DAG «під капотом». У них кожна трансформація даних – чи то фільтрація таблиці, чи агрегація, чи об’єднання – представляється як вузол графу. Система генерує повний DAG перед виконанням набору дій, визначених розробником. Це дозволяє механізму виконання ефективно налаштувати розподілене виконання, дослідити весь процес і визначити області, які потребують оптимізації (наприклад, об’єднання кількох операцій в одну або одночасне виконання різних гілок).

В області оркестрації робочих процесів (workflow orchestration), системи як Apache Airflow [43] виводять концепцію DAG на рівень користувача [93]. Розробник явно описує DAG, де кожен вузол – це окреме завдання (наприклад, «завантажити дані з бази», «навчити модель», «зберегти результат»). Ребра, що з’єднують вузли, визначають залежності; завдання B не розпочнеться, доки не буде завершено завдання A. Помилки та повторні спроби обробляються самою системою оркестратора, яка також гарантує, що завдання виконуються у правильному порядку.

Хоча сучасні інструменти управління обчисленнями, такі як Apache Airflow, підтримують механізми динамічної генерації графів обробки [44] цей процес реалізується переважно через явне написання програмного коду. Такий підхід вимагає наявності спеціалізованих навичок для створення логіки генерації та розгортання необхідного середовища. Це обмежує можливість використання таких інструментів як вбудованих компонентів у автономних агентах, які повинні автоматично синтезувати та виконувати моделі в ізольованих середовищах з обмеженими ресурсами.

1.3.3. Формати серіалізації моделей та обмеження їх застосування

Щоб використовувати навчену модель машинного навчання в майбутньому, її необхідно зберегти. Цей процес, відомий як серіалізація, перетворює об’єкт моделі з усіма її параметрами у формат, що можна записати на диск. Згодом, таку модель можна відновити у пам’яті без необхідності повторного навчання. Існує декілька поширених підходів до серіалізації, однак їхні можливості стають обмеженими при переході від роботи з окремими моделями до складних, багатокомпонентних методів конструювання АСМ.

Найбільш прямим та поширеним методом у середовищі Python є використання бібліотек, таких як **pickle** або його більш ефективний для великих масивів NumPy аналог **joblib**. Ці інструменти дозволяють надійно серіалізовувати практично будь-який об’єкт Python, включаючи моделі, навчені за допомогою *scikit-learn* та подібних. Для забезпечення відтворюваності, інформація про версії бібліотек, що використовувалися, зазвичай зберігається, що дозволяє відновити необхідне програмне оточення [1].

Однак, навіть за умови вирішення задачі сумісності версій, фундаментальний виклик полягає в тому, що серіалізація окремих моделей не вирішує задачу відтворення всього процесу конструювання АСМ. Якщо архітектура складається з багатьох взаємопов’язаних компонентів, простого набору збережених файлів недостатньо. Відсутня формалізована інформація про те, в якому порядку ці моделі

мають виконуватися, як поєднуються їхні результати та які дані подаються на вхід кожній з них. Ця логіка взаємодії залишається поза межами самих серіалізованих файлів.

Для вирішення задачі сумісності та портативності моделей між різними програмними середовищами було розроблено стандартизовані формати. Одним з найвідоміших є **ONNX (Open Neural Network Exchange)** [62]. Його головна мета – забезпечити інтероперабельність, дозволяючи експортувати моделі, навчені в одному фреймворку (напр., PyTorch), і виконувати їх в іншому (напр., TensorFlow або на спеціалізованому апаратному забезпеченні). ONNX описує обчислювальний граф самої моделі на рівні низькорівневих математичних операторів, що робить її надзвичайно портативною. Іншим схожим за ідеєю стандартом є **PMML (Predictive Model Markup Language)** [10], який використовує XML для опису моделей, забезпечуючи їхню незалежність від мови програмування та платформи виконання.

Незважаючи на їхню потужність, ці стандартизовані формати мають спільне обмеження в контексті нашого дослідження: вони зосереджені на описі однієї, монолітної моделі. Вони чудово справляються із завданням представити архітектуру нейронної мережі або параметри логістичної регресії. Проте, вони не надають вбудованих засобів для опису всього багатоетапного процесу виконання, що є характерним для складних ансамблів. Наприклад, стандартний ONNX не може описати логіку, де спочатку виконується кластеризація, потім на основі її результату обирається одна з кількох спеціалізованих моделей, а наприкінці їхні прогнози усереднюються.

Таким чином, існуючі формати серіалізації (ONNX, PMML) ефективно зберігають окремі моделі, але не здатні описати складну логіку взаємодії компонентів у різнорідних ансамблях, що об’єднують різні процеси, такі як передобробка даних, кластеризація, рециркуляція. Відсутність єдиного формату для опису всього життєвого циклу складної моделі (від сирих даних до прогнозу) призводить до

розриву між етапом досліджень та експлуатації, унеможливлюючи гарантовану відтворюваність результатів роботи агента.

1.4. Постановка задачі

Проаналізувавши існуючі підходи до моніторингу та методи машинного навчання, ми дійшли висновку, що багатошарові модельні структури є перспективним напрямком для забезпечення точності в умовах динамічних змін середовища. Крім того, виявлено необхідність покращення процесів адаптації моделей до неоднорідності вхідних даних. Тому в цій роботі треба дослідити процеси побудови моделей через застосування гібридних підходів, а також формування багатошарових поєднань алгоритмів для виконання завдань моніторингу та прогнозування стану віртуальних роботів. Для гарантування практичної застосовності отриманих результатів також необхідно дослідити засоби автоматизованого синтезу та відтворюваного виконання таких структур.

Метою роботи є підвищення ефективності функціонування моніторингового програмного агента у інформаційній технології інтелектуального моніторингу шляхом розробки методів побудови та програмної реалізації багатошарових агентних моделей. Для досягнення поставленої мети необхідно виконати такі завдання:

1. Проаналізувати існуючі методи побудови та програмної реалізації моделей машинного навчання, які можуть бути використані при побудові моніторингових програмних агентів.
2. Розробити метод підвищення точності агентних моделей за рахунок зростання однорідності точок спостереження шляхом використання кластеризації вхідних даних та інтеграції прогнозів спеціалізованих кластерних моделей як додаткових ознак.

3. Удосконалити метод синтезу багатошарових моделей (рециркуляція) шляхом залучення різнотипних алгоритмів машинного навчання для формування структурних шарів.
4. Розвинути метод багатошарового синтезу за рахунок формалізації процесів конструювання алгоритмів на основі спрямованого ациклічного графу (DAG) для забезпечення багатоетапності та детермінізму обчислень.
5. Побудова інформаційної технології проектування програмного забезпечення моніторингових програмних агентів у середовищі інтелектуального моніторингу.

РОЗДІЛ 2
ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ ПРОЦЕСУ

2.1. Формалізація завдань досліджень

На основі аналізу проблематики, проведеного у першому розділі, задачу моніторингу інформаційної системи можна сформулювати як задачу відновлення функціональної залежності (регресії) між поточним станом системи, що описується вектором вхідних параметрів, та цільовим показником.

Нехай множина історичних спостережень D представлена у вигляді набору пар (x_i, y_i), де $i = 1, \ldots, n$ — індекс спостереження (миттєвий зріз або snapshot стану системи).

Вектор вхідних ознак $x_i \in X$ характеризує стан об'єкта в i-й момент. Враховуючи специфіку обробки реальних даних, простір ознак X є гетерогенним і може бути представлений як декартовий добуток просторів числових та категоріальних ознак:

$$X = \mathbb{R}^m \times C^k,$$

де m — кількість числових параметрів (наприклад, кількісні виміри процесів, вартісні або фізичні характеристики), а k — кількість категоріальних параметрів (наприклад, категорія об'єкта, статус системи, географічна мітка).

Цільова змінна $y_i \in Y \subseteq \mathbb{R}$ є кількісною характеристикою, яку необхідно спрогнозувати (наприклад, цільова вартість, рівень навантаження або інший кількісний індикатор).

Передбачається, що існує невідома детермінована залежність $F: X \to Y$, яка ускладнена випадковим шумом ϵ:

$$y = F(x) + \epsilon,$$

де ϵ — випадкова величина, що відображає вплив неврахованих факторів та стохастичну природу середовища.

Задачею моніторингового агента є побудова апроксимуючої функції (моделі) f, яка мінімізує відхилення від істинних значень y на нових даних. У багатьох класичних підходах машинного навчання зазвичай заздалегідь фіксується базовий клас алгоритму (наприклад, конкретна архітектура нейромережі чи метод дерев рішень),

для якого здійснюється підбір гіперпараметрів та оптимізація внутрішніх ваг. У даній роботі задача розширюється: агент повинен автономно формувати архітектуру всього конвеєра обробки даних. Це означає, що система самостійно визначає комбінацію різних алгоритмів, способи їхньої взаємодії та необхідність попередньої кластеризації, одночасно підбираючи їхні внутрішні параметри.

Формально синтезовану модель можна записати як:

$$\hat{y}_i = f(x_i, S, \theta) \quad \#(2.1)$$

де:

- $S \in S$ — структура моделі (архітектура ансамблю, кількість шарів, набір обраних алгоритмів ACM, методи агрегації) з простору можливих структур S;
- $\theta \in \Theta_S$ — набір внутрішніх параметрів та гіперпараметрів, що відповідають обраній структурі S.

Тоді задача навчання агента зводиться до пошуку оптимальної пари (S^*, θ^*), яка мінімізує емпіричний ризик на навчальній вибірці:

$$(S^*, \theta^*) = \underset{S \in S, \theta \in \Theta_S}{argmin}\, L(f(X, S, \theta), Y) \#(2.2)$$

де L — функція втрат (Loss Function).

Для оцінки якості прогнозних моделей використовуються середньоквадратична похибка (MSE) або її корінь (RMSE), середня абсолютна похибка (MAE) та коефіцієнт детермінації (R^2):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \#(2.3)$$

RMSE сильніше штрафує модель за великі відхилення, що є важливим для задач виявлення аномалій та збоїв. Водночас для оцінки середнього абсолютного відхилення, яке є стійкішим до поодиноких викидів, застосовується MAE **(Mean Absolute Error)**:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Коефіцієнт детермінації (R^2) використовується для оцінки частки дисперсії цільової змінної, яку здатна пояснити синтезована модель:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \acute{y})^2} \quad \#(2.5)$$

де $\acute{y}$ — середнє значення цільової змінної на вибірці.

Складність вирішення цієї задачі полягає у тому, що простір можливих структур S (можливі комбінації алгоритмів, їхня послідовність та ієрархія) є комбінаторно складним. Як варіант вирішення даної задачі, у даному дослідженні пропонуються нові методи алгоритмічної побудови складних ансамблів. Зокрема, у наступних підрозділах розглядаються архітектурні підходи до підвищення однорідності простору ознак (розділ 2.2) та використання гетерогенної багатошарової рециркуляції (розділ 2.3), які дозволяють формувати структуру S^* для роботи з різнорідними даними.

2.2. Метод підвищення однорідності вхідних даних

2.2.1. Методи та засоби для інженерії ознак та моделювання на основі кластеризації

Запропонований метод підвищення точності заснований на послідовній комбінації кількох методів машинного навчання. Він включає етапи кластеризації для виявлення внутрішньої структури даних та побудови спеціалізованих моделей для кожного сегмента. Завершальним етапом є навчання фінальної моделі на розширеному наборі даних. Далі розглянемо ключові методи, що використовуються на кожному з цих етапів, та їхню роль у запропонованому методі.

2.2.1.1. Методи кластеризації даних

Кластеризація є першим кроком у методі. Її мета – розділити вихідний масив різнорідних даних на групи (кластери) таким чином, щоб об’єкти всередині однієї

групи були максимально схожими між собою, а об’єкти з різних груп – максимально відмінними. Оскільки не існує універсального алгоритму кластеризації, що був би оптимальним для всіх типів даних, вибір конкретного методу та його параметрів суттєво впливає на кінцевий результат.

Формально процес розбиття вхідного простору X на k кластерів можна записати як відображення:

$$X \xrightarrow{\text{Clustering}} \{C_1, C_2, \ldots, C_k\} \quad \#(2.6)$$

де C_j — підмножина даних (сегмент), що належить до j-го кластера, причому $\cup_{j=1}^{k} C_j = X$ та $C_a \cap C_b = \varnothing$ для $a \neq b$.

Протягом дослідження розглядалися наступні методи кластеризації:

• **Метод K-Середніх (K-Means)** [14]: Обчислювально ефективний ітеративний алгоритм. Складається з кроку призначення (екземпляр відноситься до кластера з найближчим центроїдом) та кроку оновлення (переобчислення центроїдів як середніх значень). Незважаючи на швидкодію, метод чутливий до початкової ініціалізації та припускає, що кластери мають випуклу, сферичну форму приблизно однакового розміру. [57] .

• **Гаусові Суміші (Gaussian Mixture Models, GMM)** [5]: Ймовірнісний підхід, що моделює дані як суміш багатовимірних гаусових розподілів. На відміну від K-Means, GMM реалізує «м’яку» кластеризацію: для кожного екземпляра обчислюється ймовірність його належності до кожного компонента. Це дозволяє моделювати кластери складної еліпсоїдної форми та різної щільності за допомогою алгоритму максимізації очікування (EM)[21].

• **Ієрархічна агломеративна кластеризація (Hierarchical Agglomerative Clustering):** Метод будує ієрархію кластерів у вигляді дендрограми, послідовно об’єднуючи найближчі сегменти. Відстань між кластерами визначається за критеріями зв’язку (linkage), такими як ward (мінімізація дисперсії), complete (максимальна відстань) або average (середня відстань) [15].

2.2.2. Побудова спеціалізованих моделей для сегментів

Після розділення вихідного масиву даних на кластери, для кожного з них будується окрема модель регресії. Математично цей етап описується як навчання локальної моделі M_j виключно на підмножині даних відповідного кластера:

$$M_j = \text{Train}\left(X_{C_j}, Y_{C_j}\right), j = 1..k \#(2.7)$$

де X_{C_j} та Y_{C_j} — матриця ознак та вектор цільових значень для екземплярів, що потрапили до сегмента C_j.

Модель, навчена виключно на даних одного сегмента, виконує роль «локального експерта», здатного виявляти специфічні закономірності, притаманні лише цій підгрупі. Для побудови таких моделей агент виконує перебір доступних алгоритмів синтезу (АСМ), автоматично обираючи оптимальну конфігурацію для кожного кластера.

2.2.3. Використання класифікатора кластерів

Класифікатор кластерів є ще одним елементом системи, що використовується у запропонованому методі. На етапі навчання він генерує нову категоріальну ознаку для розширеного масиву даних. Процес навчання класифікатора Cl формалізується наступним чином:

$$Cl = \text{Train}(X, L) \#(2.8)$$

де X — повна матриця початкових ознак, а L — вектор міток кластерів $L = \{l_1, \ldots, l_n\}$, отриманих на попередньому етапі кластеризації ($l_i \in \{1, \ldots, k\}$).

Оскільки якість призначення нових екземплярів безпосередньо впливає на ефективність усього підходу, до вибору класифікатора кластерів застосовуються ті ж самі вимоги: виконується перебір усіх доступних АСМ для знаходження найбільш точної моделі.

2.2.4. Побудова фінальної моделі та ансамблів на основі збагачених даних

Фінальним етапом є побудова глобальної моделі, що використовує всю доступну інформацію: як вихідні ознаки, так і згенеровані на попередніх кроках.

• Формування «збагаченого» масиву даних:

З математичної точки зору, для кожного *i*-го екземпляра формується новий розширений вектор ознак x'_i, який об'єднує оригінальні параметри, прогнози всіх локальних експертів та мітку класифікатора. Цей процес визначається формулою:

$$x'_i = x_i \cup \left\{ M_1(x_i), \ldots, M_k(x_i) \right\} \cup Cl(x_i) \# (2.9)$$

де x_i — початковий вектор ознак, $M_j(x_i)$ — прогноз j-ї спеціалізованої моделі для цього екземпляра.

Таким чином, формується новий збагачений простір ознак X'. На рис. 2.1 наведено функціональну схему формування масиву вхідних даних з доповненим словником ознак та формування шару рециркуляції (стекінгу) на ньому

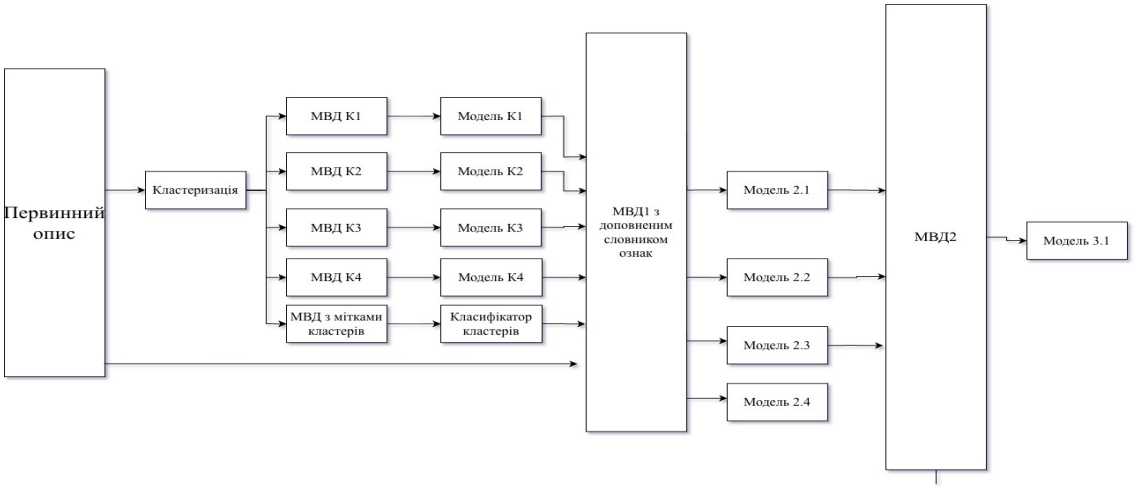


Рис. 2.1 Функціональна схема формування масиву вхідних даних з доповненим словником ознак

На рис. 2.2 наведено діаграму діяльності методу підвищення однорідності вхідних даних

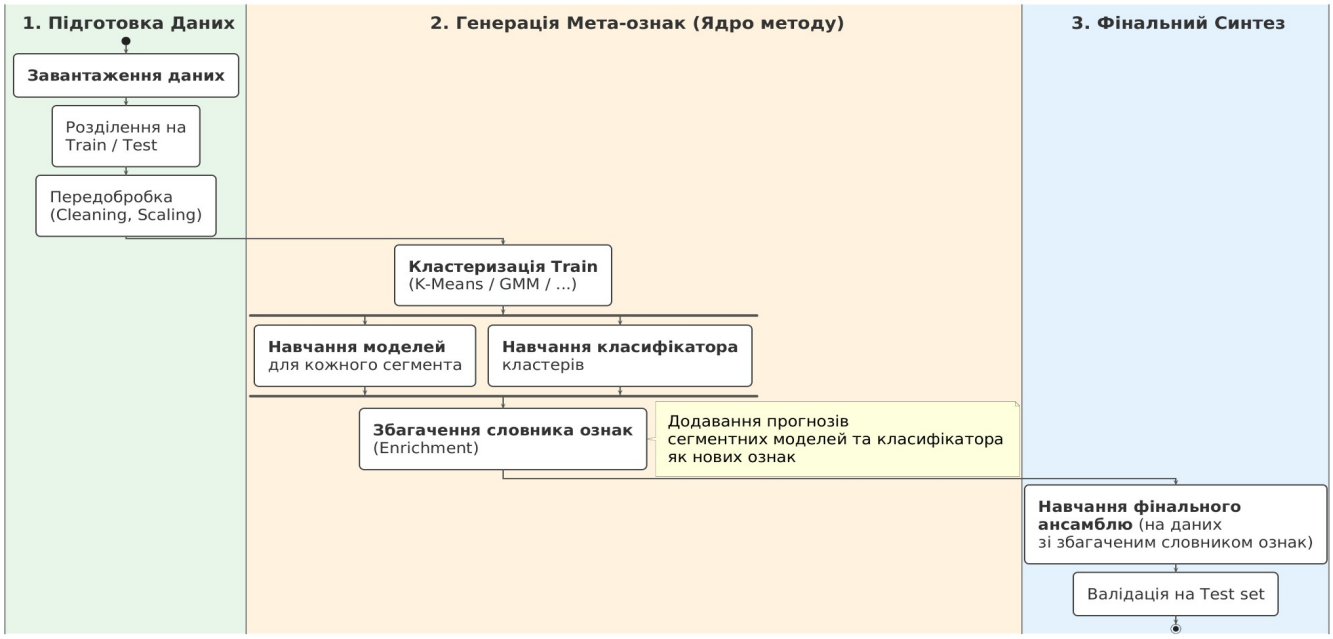


Рис. 2.2 – Діаграма процесу підвищення однорідності вхідних даних

• Побудова фінальних АСМ та ансамблів:

1. **Базові моделі на збагачених даних:** На розширеному масиві даних виконується навчання повного набору АСМ для знаходження найкращої одиночної моделі.

2. **Ансамблеві моделі (рециркуляція):** На цьому етапі передбачається побудова ансамблів за методом рециркуляції. На основі результатів попереднього кроку відбирається N найкращих моделей. Їхні прогнози додаються до розширеного масиву даних як ще один набір нових ознак, на якому навчається наступне покоління моделей.

• **Алгоритми Синтезу Моделей (АСМ), що використовуються:** Для побудови як локальних моделей M_j, так і фінальних ансамблів використовується єдиний набір алгоритмів A, який включає методи на основі дерев рішень, градієнтного бустингу та лінійної регуляризації (детальний аналіз яких наведено у підрозділі 1.2). Вибір конкретного алгоритму $a \in A$ для кожного сегмента здійснюється автоматично синтезатором моделей на основі мінімізації функції втрат L на валідаційній вибірці.

2.3. Удосконалений метод рециркуляції

2.3.1. Основні принципи удосконалення методу

Класичні методи багатошарової рециркуляції переважно використовують однотипні алгоритми (наприклад, поліноміальні функції) на кожному рівні структури. З іншого боку, методи стекованої генералізації (stacking) застосовують різнотипні алгоритми, проте їхня архітектура зазвичай обмежується двома рівнями.

Удосконалення методу в даній роботі полягає у залученні різнорідного набору методів машинного навчання для формування кожного шару в глибоких структурах рециркуляції.

Практична доцільність цього підходу зумовлена зміною характеристик масиву даних у процесі багатошарової обробки. На нульовому шарі моделі опрацьовують початкові ознаки, що можуть мати складні нелінійні зв’язки. На наступних шарах простір ознак збагачується прогнозами моделей попереднього рівня. Ці нові мета-ознаки є висококорельованими, що призводить до проблеми мультиколінеарності. Оскільки різні класи алгоритмів машинного навчання (дерева рішень, бустинг, лінійна регуляризація, нейронні мережі) по-різному реагують на мультиколінеарність, використання різнотипного набору алгоритмів на кожному шарі дозволяє системі емпірично знаходити моделі з найменшою похибкою на конкретному етапі перетворення даних.

2.3.2. Методи та технології

Існуючі підходи до побудови багатошарових синтезаторів [28], зарекомендували себе як ефективний інструмент. Разом із тим, їхнє застосування в автономних моніторингових агентах вимагає оптимізації обчислювальних витрат та контролю за архітектурою ансамблю.

Процес побудови моделі за удосконаленим методом у даному дослідженні містить наступні рішення:

1. Застосування набору різнорідних алгоритмів (АСМ) на кожному рівні з їх подальшим відбором виключно за показниками похибки на валідаційній вибірці.

2. Використання пошарового тестування для оцінки точності побудованої структури на кожному рівні. Це дозволяє проаналізувати динаміку похибки та обрати оптимальну глибину моделі після завершення процесу конструювання.

3. Розмежування процесів: пошук гіперпараметрів виконується на базовому шарі, а на подальших шарах алгоритми можуть тренуватися із зафіксованими або стандартними конфігураціями для зменшення обчислювального навантаження.

4. Застосування параметрів ширини: контроль кількості моделей на базовому рівні та на подальших шарах рециркуляції для управління інформаційною ємністю структури.

2.3.2.1. Концептуальна схема удосконаленого процесу

Загальну логіку процесу відображає концептуальна блок-схема (рис. 2.3). Алгоритм містить наступні етапи:

1. Підготовка даних. Виконується базова передобробка (заповнення пропусків, кодування, масштабування) та розділення масиву на набори для навчання, валідації та тестування.

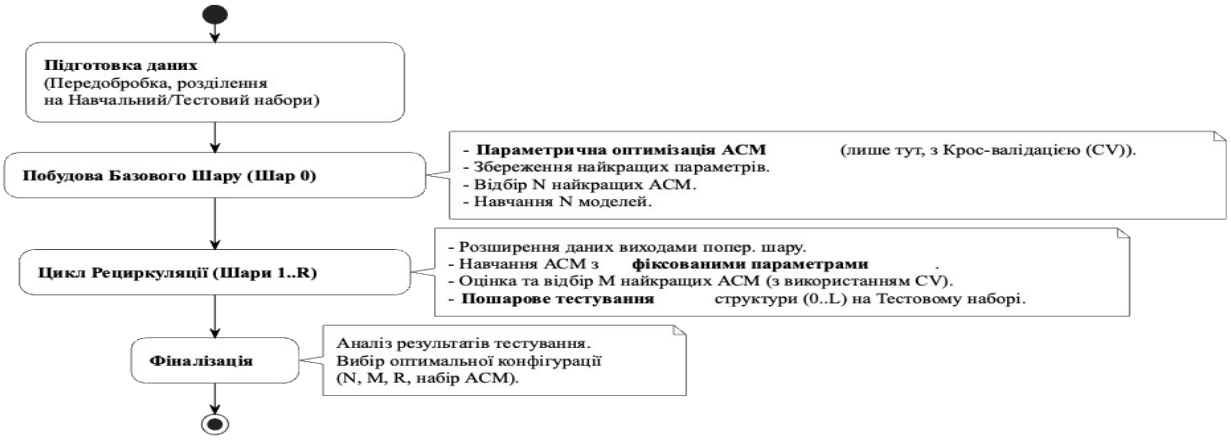


Рисунок 2.3 — Концептуальна блок-схема удосконаленого процесу синтезу багатошарових моделей

2. Побудова базового шару (Шар 0).. На цьому етапі до вхідних даних застосовується заданий набір алгоритмів. За потреби проводиться їх параметрична оптимізація. Формально, для кожного алгоритму з доступного простору здійснюється

пошук вектора гіперпараметрів θ^*, який мінімізує функцію втрат L на валідаційній множині в рамках k-блокової крос-валідації:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \frac{1}{k} \sum_{j=1}^{k} L\left(f\left(X_{val}^{(j)}, \theta\right), Y_{val}^{(j)}\right) \quad \#(2.10)$$

Найкращі знайдені конфігурації параметрів θ^* зберігаються. Після цього відбираються N найкращих базових моделей, які формують ансамбль нульового шару $M^{(0)} = \{f_1^{(0)}, f_2^{(0)}, \ldots, f_N^{(0)}\}$. Застосування параметра N задає ширину базового шару, обмежуючи кількість моделей, що передають свої прогнози на наступний шар.

2. **Цикл рециркуляції (Шари 1..R).** Процес ітеративно повторюється до досягнення заздалегідь заданої максимальної глибини R. На кожному шарі L виконуються такі дії:

• **Формування розширеного простору ознак.** На кожному шарі L (від 1 до максимального R) вхідні дані розширюються прогнозами найкращих моделей попереднього шару. Новий простір ознак $X^{(L)}$ для шару L формується шляхом об'єдннння вихідного простору ознак $X^{(0)}$ (або збагаченого простору X' з етапу кластеризації) та векторів прогнозів моделей попереднього шару:

$$X^{(L)} = X^{(L-1)} \cup \left\{f_1^{(L-1)}\left(X^{(L-1)}\right), \ldots, f_K^{(L-1)}\left(X^{(L-1)}\right)\right\} \quad \#(2.11)$$

де K — кількість моделей, відібраних на попередньому шарі ($K = N$ для переходу від Шару 0 до Шару 1, та $K = M$ для всіх наступних глибших шарів

• **Навчання набору АСМ.** На масиві $X^{(L)}$ повторно навчаються всі доступні класи алгоритмів машинного навчання. Залежно від конфігурації, вони можуть використовувати збережені параметри θ^* з нульового шару або стандартні налаштування θ

• Відбір M моделей**.** З усіх навчених на поточному шарі АСМ відбирається підмножина $M^{(L)}$ з M найкращих моделей за результатами крос-валідації: $M^{(L)} = \{f_1^{(L)}, \ldots, f_M^{(L)}\}$. Застосування параметра M визначає ширину всіх наступних шарів

рециркуляції. Синтезатор ранжує моделі за метрикою помилки і відбирає найточніші, незалежно від їхнього типу.

• Пошарове тестування. Після завершення навчання шару L загальна точність багатошарової структури перевіряється на відкладеному тестовому наборі. Результат тестування зберігається для подальшого аналізу.

5. **Фіналізація (Пост-селекція).** За результатами пошарового тестування порівнюються метрики якості (наприклад, $L = \text{RMSE}$) для кожного з побудованих шарів від 0 до R. Фінальною структурою агента стає той рівень L_{best}, який показав найменшу похибку на тестовій вибірці:

Залучення гетерогенних алгоритмів машинного навчання до процесу рециркуляції дозволяє збільшити варіативність моделей у шарі. Агент емпірично перебирає різні алгоритми та за результатами фінального тестування обирає оптимальну конфігурацію (L_{best}), що підвищує загальну повноту опису об’єкта та мінімізує кінцеву похибку прогнозування.

2.3.3. Деталізація удосконаленого процесу синтезу

Для ілюстрації логіки взаємодії крос-валідації, навчання та пошарового тестування розроблено деталізовані діаграми активності. Процес розділено на етап підготовки та синтезу базового шару (рис. 2.4) та етап ітеративної рециркуляції (рис. 2.5).

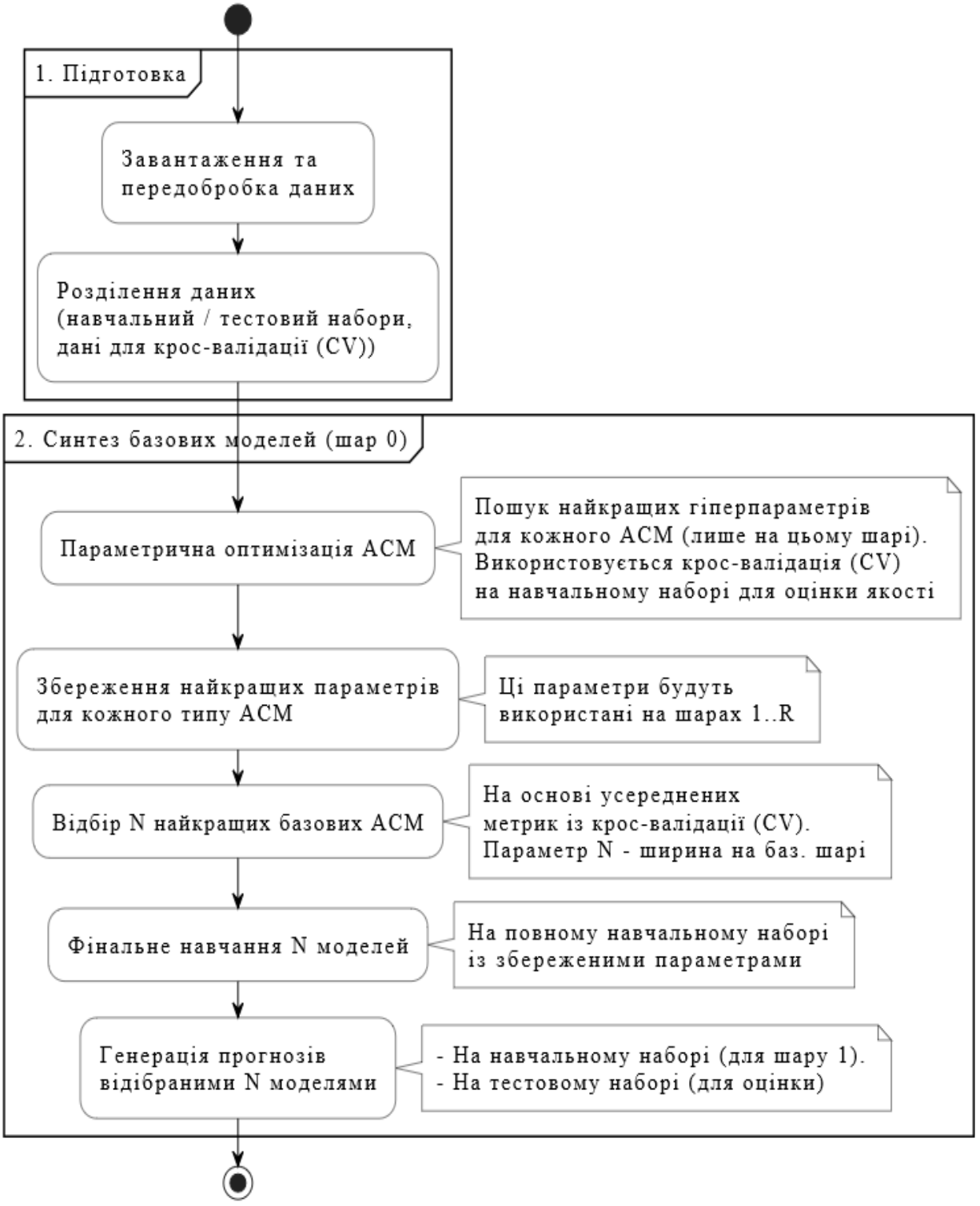


Рисунок 2.4 — Деталізована діаграма активності. Етап підготовки та синтезу базового шару (шар 0)

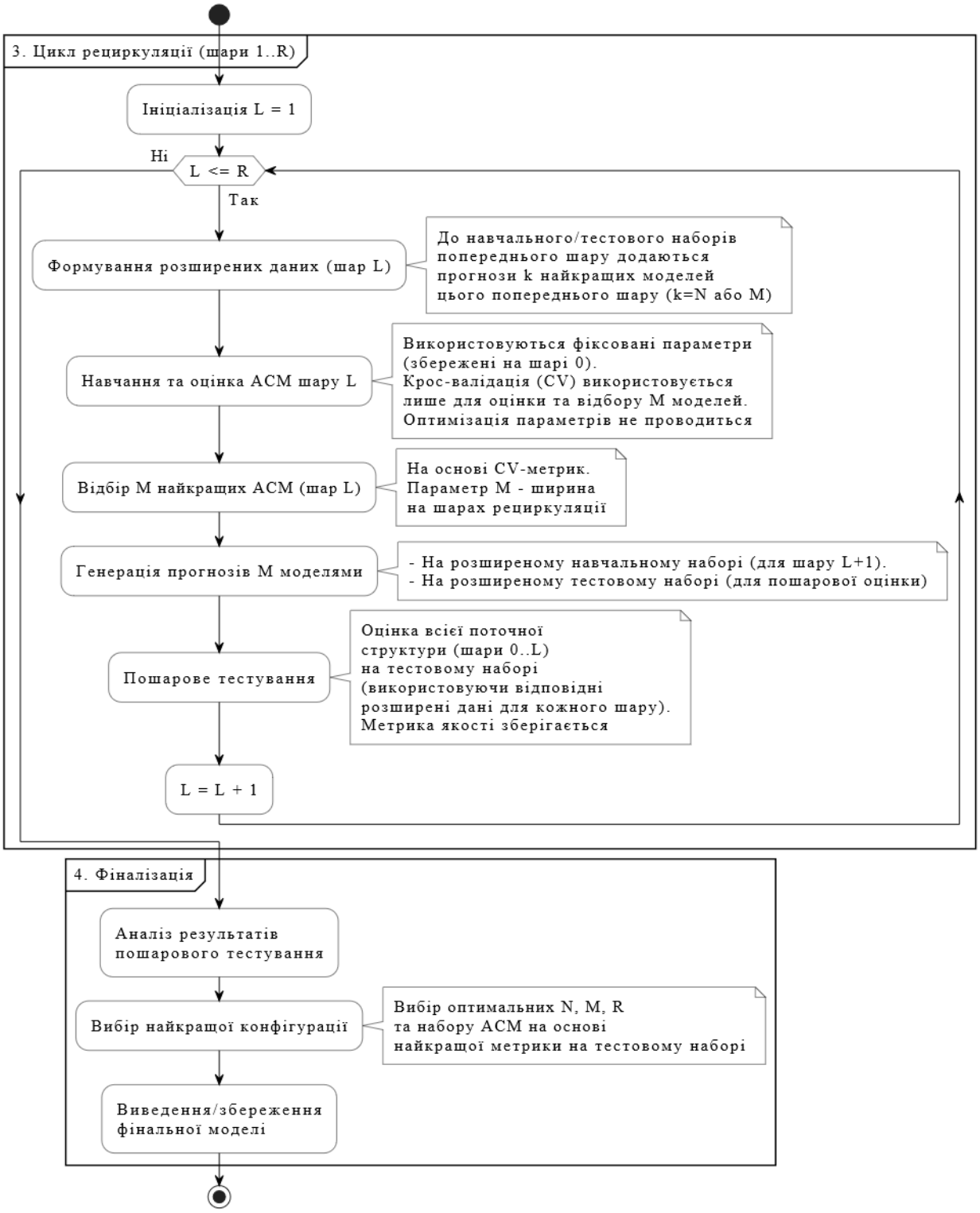


Рисунок 2.5 — Деталізована діаграма активності: Етап рециркуляції (шари 1..R) та фіналізації

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1. Характеристика наборів даних та особливості експериментів

3.1.1. Характеристика наборів даних

Однією з вимог до методів машинного навчання у складі автономних моніторингових агентів є їхня здатність працювати з масивами інформації різної структури. Для об'єктивної перевірки запропонованих методів синтезу моделей було обрано три відкриті набори табличних даних.

Головним критерієм їх відбору стала різноманітність топології простору ознак (різний обсяг вибірки, співвідношення числових та категоріальних змінних, розмірність масивів). Такий вибір дозволяє протестувати алгоритмічне ядро агента на універсальність та перевірити його здатність автоматично адаптуватися до різнотипних вхідних параметрів. У дослідженні використано такі набори даних:

1. **Набір даних 1: Прогноз вартості поїздок (Логістичні та часові метрики)**

Масив містить інформацію про вартість поїздок залежно від супутніх параметрів**.**

- **Обсяг даних:** 1001 спостереження.
- **Простір ознак:** 11 ознак, серед яких операційні (базова вартість, відстань), часові (час прийняття та закінчення) та зовнішні умови (стан трафіку, погода). Цільовою змінною є фінальна ціна.
- **Особливості набору:** Масив характеризується невеликим обсягом вибірки. Його використання дозволяє перевірити ефективність навчання ансамблів в умовах обмеженої кількості екземплярів даних, де цільова змінна формується під впливом різнотипних факторів.

2. **Набір даних 2: Оцінка характеристик фізичних об'єктів (Дорогоцінні камені)**

Масив містить дані щодо оцінки фізичних характеристик каменів та їх впливу на ринкову вартість.

• **Обсяг даних:** 6486 спостережень.

• **Простір ознак:** 18 ознак, що мають змішану природу: 8 числових (геометричні параметри: довжина, ширина, глибина, вага) та 10 категоріальних (якість огранювання, колір, чистота). Цільовою змінною виступає неперервна величина (вартість).

• **Особливості набору:** Датасет відрізняється високою значущістю категоріальних ознак. Тестування на цих даних дозволяє перевірити ефективність методів внутрішньої передобробки (кодування) та здатність агента знаходити закономірності у масивах зі змішаними типами даних.

3. **Набір даних 3: Аналіз демографічних та поведінкових показників (Маркетинг)**

Масив описує взаємодію користувачів із сервісом та їхні реакції на події.

• **Обсяг даних:** 2206 спостережень.

• **Простір ознак:** 39 числових ознак, що включають демографію, поведінкові фактори (сума витрат на різні категорії товарів, давність покупок) та реакції на стимули. Цільовою змінною є кількісний рівень заробітку.

• **Особливості вибору:** Масив характеризується відносно високою розмірністю простору ознак. Тестування на цих даних дозволяє оцінити, наскільки ефективно алгоритми машинного навчання здатні виділяти значущі параметри серед великої кількості вхідних змінних.

3.1.2. Особливості проведених експериментів

Для об’єктивної перевірки та валідації розроблених методів адаптивного синтезу було спроектовано два незалежні експериментальні дослідження. Перше дослідження спрямоване на ізольовану оцінку методу підвищення однорідності масиву даних (через попередню кластеризацію). Друге — на перевірку ефективності удосконаленого методу багатошарової рециркуляції.

3.1.2.1. Простір алгоритмів синтезу моделей (АСМ)

Обидва експерименти використовували єдиний набір базових алгоритмів синтезу моделей (АСМ) для забезпечення порівнюваності результатів. До простору АСМ увійшли:

1. **Методи на основі дерев рішень:** Decision Tree, Random Forest, ExtraTrees.
2. **Методи градієнтного бустингу:** Gradient Boosting, XGBoost, LightGBM, AdaBoost.
3. Лінійні методи з регуляризацією: Ridge, Lasso, ElasticNet.
4. **Нейронні мережі:** Багатошаровий перцептрон (MLP).

Примітка: На етапі попереднього оцінювання алгоритмів (20 ітерацій навчання) також розглядався алгоритм TabNet. Однак він продемонстрував низьку прогнозну спроможність на обраних наборах даних та невиправдано високу ресурсоємність порівняно з деревами рішень і бустингом. Зважаючи на обмеження обчислювальних ресурсів автономних агентів, складні нейромережеві архітектури типу TabNet було вилучено з основного етапу експериментів,

3.1.2.2. Метрики оцінки якості

Для аналізу ефективності моделей на відкладених тестових вибірках використовувався набір стандартних метрик, які наведені у підрозділі 2.1 (формули 2.3 – 2.5):

• **RMSE (Root Mean Squared Error):** основна метрика цільової функції. Особливо чутлива до значних помилок прогнозування, оскільки квадратична функція сильно «штрафує» модель за великі відхилення. Важлива для агента при детекції критичних аномалій.

• **MAE (Mean Absolute Error):** показує середнє абсолютне відхилення прогнозу. На відміну від RMSE, є більш стійкою (робастною) до одиничних статистичних викидів у телеметрії.

• **Коефіцієнт детермінації (R^2):** статистична міра, що показує, яку частку дисперсії вхідних даних здатна пояснити синтезована модель (значення, близьке до 1.0, свідчить про високу якість апроксимації).

3.1.2.3. Умови проведення Експерименту 1: Підвищення однорідності вхідних даних

Метою цього етапу було виміряти ефект від збагачення простору ознак за допомогою кластеризації.

1. **Конфігурації кластеризації:** У рамках дослідження проводився порівняльний аналіз впливу трьох фундаментально різних підходів до сегментації: геометричного (K-Means), ймовірнісного (GMM) та ієрархічного (Agglomerative Clustering). Важливо зазначити, що метою цього етапу є не пошук єдиного "універсального" алгоритму кластеризації, який був би найкращим для всіх задач. Натомість, експеримент має на меті продемонструвати, що оптимальна стратегія розбиття даних жорстко залежить від специфічної топології конкретного набору даних. Це підтверджує тезу про те, що експериментальний перебір (або автоматизований пошук) параметрів сегментації є критично важливою складовою запропонованого методу адаптивного синтезу.

2. **Умови моделювання:** Гіперпараметрична оптимізація (Optuna) для АСМ у цьому експерименті не проводилася; використовувалися стандартні параметри алгоритмів. По-перше, використання стандартних конфігурацій дозволило ізолювати ефект від інженерії ознак: будь-яка зміна точності (метрика Δ RMSE) може бути інтерпретована як результат збагачення простору ознак, а не вдалого підбору гіперпараметрів. Головним критерієм успіху є не досягнення абсолютного мінімуму помилки, а відносне покращення показників моделей на збагачених даних ($X_{enriched}$) порівняно з контрольними моделями на сирих даних (X_{train}). По-друге, це зумовлено надмірною ресурсоємністю процесу: оптимізація кожної локальної моделі для кожного з перевірених методів кластеризації вимагає значних обчислювальних витрат.

3. **Схема порівняння:** Для оцінки приросту точності експеримент передбачав навчання двох паралельних гілок моделей для кожного набору даних:

1. **Контрольна гілка (Baseline):** Одиночні моделі та прості ансамблі навчені на вихідних («сирих») даних. Це дозволяє встановити базовий рівень точності, якого можна досягти традиційними методами.
2. **Експериментальна гілка:** Аналогічні одиночні моделі та ансамблі, навчені на наборі даних зі збагаченим словником ознак (після додавання мета-ознак з етапу кластеризації).

На рис. 3.1 наведено функціональна схема отримання контрольних значень показників ефективності моделей з яким порівнюються запропонований метод

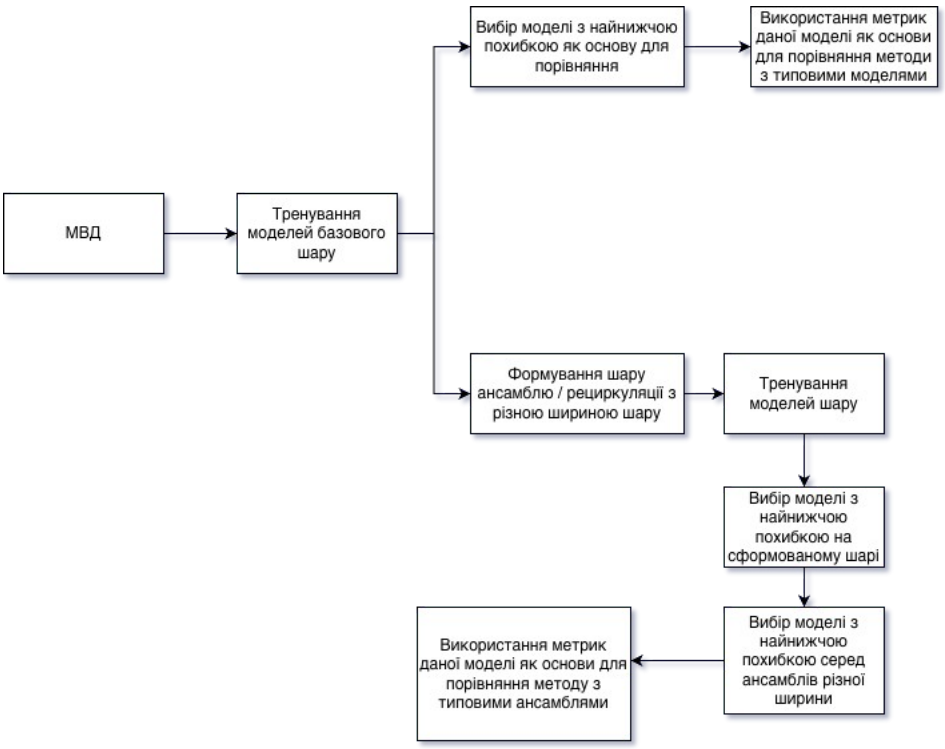


Рис. 3.1 Функціональна схема контрольної гілки експерименту

На рис. 3.2 наведено функціональна схема отримання значень показників моделей, що побудовані на масиві вхідних ознак збагаченого за запропонованим методом.

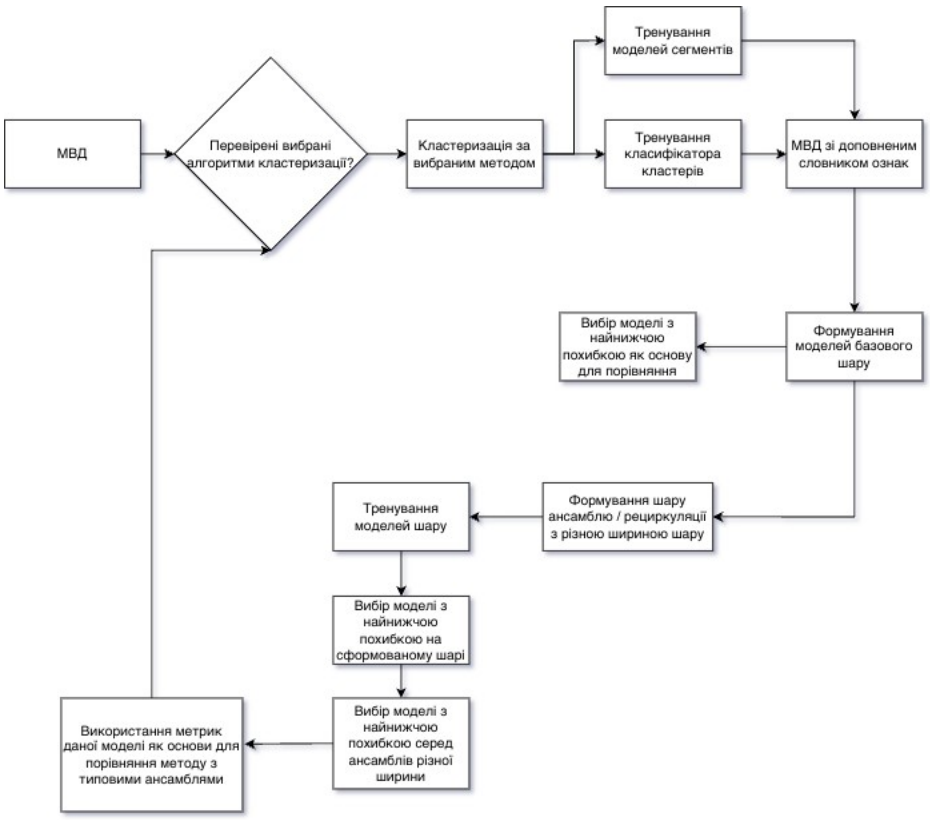


Рис. 3.2 Функціональна схема експериментальної гілки

3.1.2.4. Умови проведення Експерименту 2: Багатошарова рециркуляція

Метою другого етапу була оцінка здатності агента до побудови глибоких гетерогенних ансамблів та адаптивного вибору їхньої структури.

3. **Умови моделювання:** На відміну від першого експерименту, тут ключову роль відігравала параметрична оптимізація базових алгоритмів. Для відбору найкращих моделей на нульовому шарі використовувався фреймворк Optuna (алгоритм TPE, 100 ітерацій пошуку).

4. **Параметри синтезу структури:** Було задано такий пошуковий простір для формування багатошарової архітектури: кількість найкращих базових моделей, що обираються на початковому шарі $N \in \{2,3,4,5\}$; кількість моделей на прихованих шарах $M \in \{1,2,3,4\}$; максимальна дозволена глибина рециркуляції $R=7$.

3.2. Експериментальні результати випробування методу підвищення однорідності

3.2.1. Отримані результати

У даному підрозділі проаналізовано результати порівняння ефективності алгоритмів на «сирих» та «збагачених» даних. Для полегшення інтерпретації результатів кожна зведена таблиця має стандартизовану ієрархічну структуру, яка демонструє покрокову зміну точності:

1. **Одинична контрольна модель:** найкращий одиночний алгоритм, навчений на початкових даних (базова лінія).
2. **Ансамблева контрольна модель:** найкращий класичний ансамбль моделей, навчений на початкових даних (перевірка межі можливостей традиційних підходів)
3. **Одинична модель на даних зі збагаченим словником ознак:** найкращий одиночний алгоритм, що використовує мета-ознаки з етапу кластеризації (оцінка чистого ефекту від запропонованої інженерії ознак)
4. **Ансамблеві моделі на даних зі збагаченим словником ознак:** комбінація запропонованого методу збагачення ознак та ансамблювання (фінальна демонстрація максимальної ефективності розробленої архітектури)

Для кожної моделі наведено показники ключових метрик якості (RMSE, MAE, R^2), що гарантує об’єктивність оцінки їхньої здатності до узагальнення на нових даних..

3.2.1.1. Результати для набору даних «Прогноз вартості поїздок»

Набір даних, що містить інформацію про вартість поїздок на таксі, характеризується високим рівнем випадкових коливань (стохастичності) та залежністю цільової змінної від операційних режимів. Результати експерименту підтверджують зниження похибки при застосуванні розробленого методу. Ключові показники наведено у табл. 3.1.

Для зручності інтерпретації таблиця має таку структуру:

1. **Архітектура:** вказує на тип протестованої структури — одиночна базова модель або багатошаровий ансамбль (результат рециркуляції).

2. **Метод кластеризації:** алгоритм, що використовувався для генерації мета-ознак. Для контрольної групи (на «сирих» даних) вказано статус «Відсутня».
3. **Фінальний алгоритм (ACM):** метод машинного навчання, який показав найкращий результат. Для ансамблів запис має формат *Ансамбль (N)*, де N — кількість базових моделей, чиї прогнози було передано як ознаки на фінальний шар.
4. **RMSE, MAE,** R^2 – значення метрик якості, розраховані на відкладеній тестовій вибірці

Таблиця 3.1 — Порівняння результатів моделей на наборі даних «Прогноз вартості поїздок»

Архітектура	Метод кластеризації	Фінальний алгоритм (ACM)	RMSE	MAE	R2
Контрольні моделі (базовий масив даних)					
Одиночна модель	Відсутня	ExtraTree	10.58	5.67	0.952
Ансамбль	Відсутня	Ансамбль (1)	9.83	5.44	0.959
Запропонований метод (масив зі збагаченими ознаками)					
Одиночна модель	K-Means (k=3)	GB	9.57	4.93	0.961
Ансамбль	K-Means (k=3)	Ансамбль (1)	**7.09**	4.71	**0.978**
Ансамбль	GMM (k=6)	Ансамбль (5)	8.44	**4.37**	0.970
Ансамбль	Ієрархічна (k=2)	Ансамбль (2)	9.31	4.83	0.963

Аналіз результатів. Детальний розгляд результатів, наведених у таблиці 3.2, демонструє ще більш виразну перевагу запропонованого підходу при роботі з даними, що містять логістичні та часові метрики. Найкращий ансамбль, побудований на даних зі збагаченим словником ознак (конфігурація K-means(k=3), розмір ансамблю 1),

досяг значення RMSE на рівні **7.09**. Цей показник на **27.9%** кращий за результат найкращого контрольного ансамблю (**9.83**) і суттєво перевершує базову одиночну модель (**10.58**). Таке різке зниження середньоквадратичної похибки вказує на те, що запропонований метод дозволив алгоритмам машинного навчання уникнути значної кількості великих помилок при прогнозуванні.

Такий значний приріст ефективності на цьому конкретному наборі даних можна пояснити специфікою самої предметної області. Дані про поїздки на таксі зазвичай містять чітко виражені операційні режими функціонування. Імовірно, алгоритм кластеризації K-Means (з розбиттям на 3 кластери) зміг автоматично виявити та ізолювати ці режими — наприклад, короткі поїздки в межах центру міста, середні поїздки між районами та довгі поїздки до передмістя або аеропорту. Коли система навчає окрему спеціалізовану модель для кожного з таких виявлених режимів, ці локальні моделі не намагаються узагальнити всю різнорідну інформацію одразу. Вони фокусуються лише на закономірностях свого сегмента. Це припущення також узгоджується з результатами за метрикою MAE.

Аналіз абсолютної похибки (MAE) показує, що найкращий результат (**4.37**) був отриманий при використанні конфігурації GMM (k = 6), що на **19.7%** краще за контрольну модель (**5.44**). Той факт, що для мінімізації різних типів помилок (RMSE та MAE) оптимальними виявилися різні конфігурації кластеризації (K-means(k=3) та GMM(k=6) відповідно), підкреслює важливість комплексного підходу. Це свідчить про те, що універсального методу групування ознак не існує, і архітектура моніторингового агента повинна підтримувати автоматизований перебір методів кластеризації для вибору тієї структури, яка найкраще відповідає поточним цілям моніторингу.Для наочної візуалізації переваги запропонованого методу на рис. 3.1 та рис. 3.4 продемонстровано порівняння значень метрик RMSE та MAE для найкращих ансамблевих моделей.

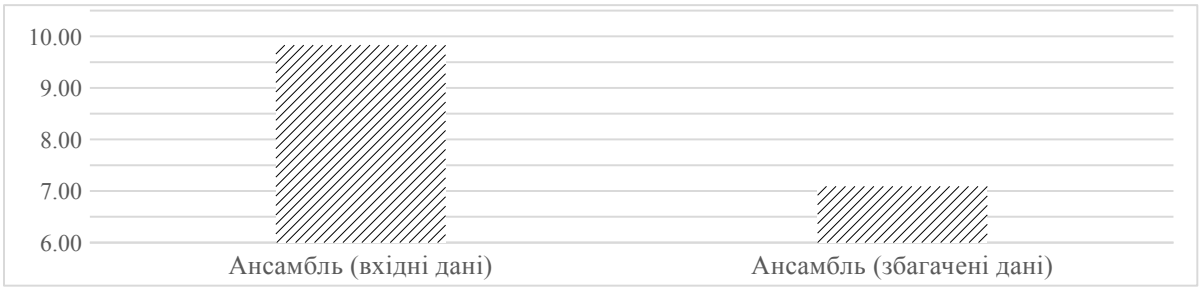


Рисунок 3.1 — Середньоквадратична похибка (RMSE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних «Прогноз вартості поїздок»

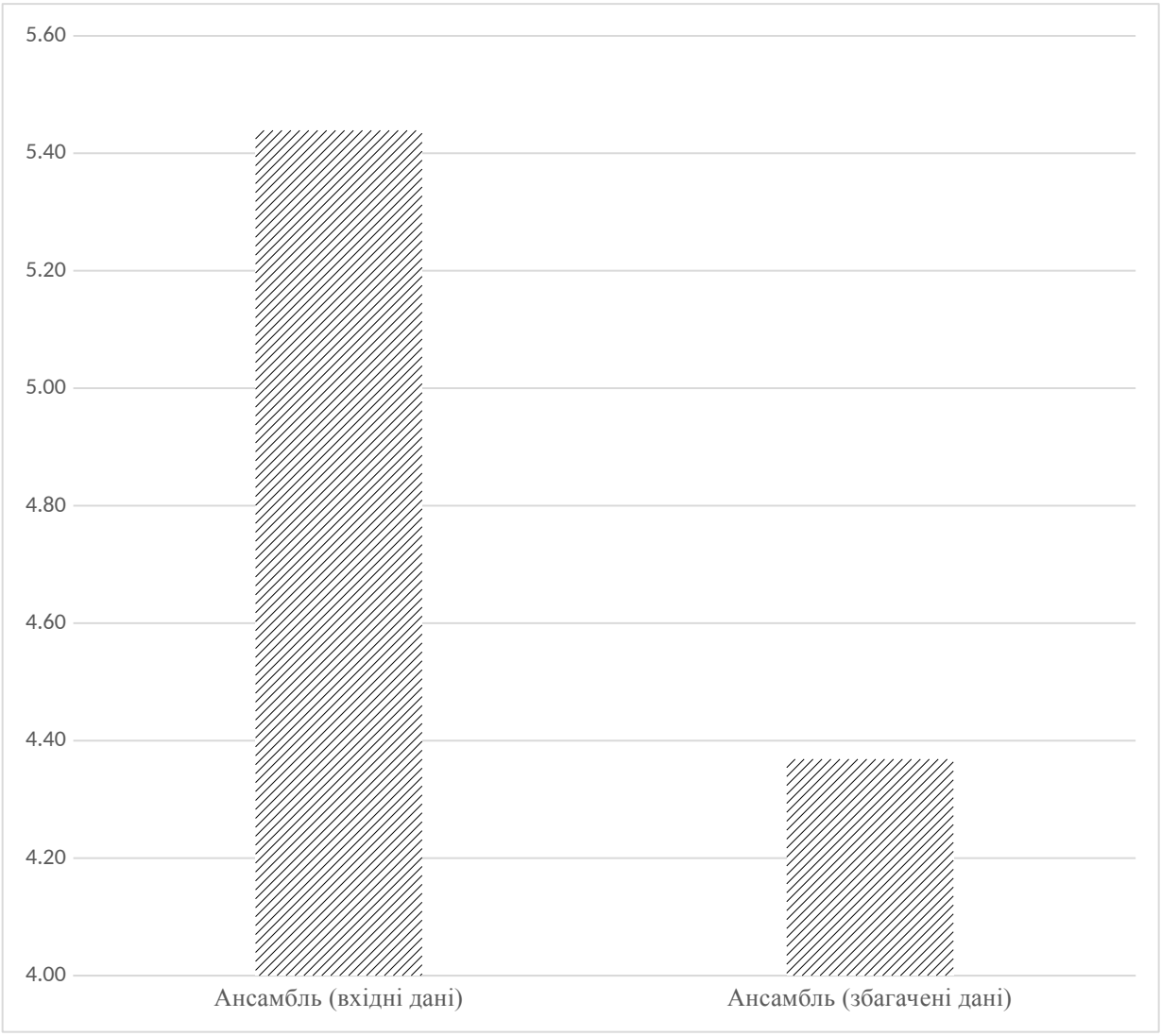


Рисунок 3.2 — Середня абсолютна похибка (MAE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних «Прогноз вартості поїздок»

3.2.1.2. Результати для набору даних 2: Характеристики фізичних об’єктів (Дорогоцінні камені)

Для першого набору даних, що містить характеристики та ціни дорогоцінних каменів, було проведено моделювання та отримано результати, ключові з яких наведено в табл. 3.2.

Таблиця 3.2 — Порівняння результатів моделей на наборі даних «Дорогоцінні камені»

Архітектура	Метод кластеризації	Фінальний алгоритм (АСМ)	RMSE	MAE	R2
Контрольні моделі (базовий масив даних)					
Одиночна	Відсутня	RF	1303.20	426.23	0.840
Ансамбль	Відсутня	Ансамбль (1)	1192.67	421.91	0.866
Запропонований метод (масив зі збагаченими ознаками)					
Одиночна	GMM(k=5)	RF	1242.95	422.60	0.855
Ансамбль	GMM(k=3)	Ансамбль (4)	**1168.79**	396.55	**0.872**
Ансамбль	K-Means(k=4)	Ансамбль (5)	1173.07	**389.98**	0.871
Ансамбль	Ієрархічна(k=3)	Ансамбль (5)	1196.79	410.53	0.865

Аналіз результатів. Аналіз результатів, наведених у таблиці 3.1, дозволяє зробити кілька важливих спостережень щодо ефективності запропонованого методу на наборі даних з характеристиками фізичних об’єктів. Насамперед, експериментальні дані підтверджують основну гіпотезу дослідження: модель, побудована за запропонованим методом підвищення однорідності даних, демонструє вищу точність порівняно з найкращою контрольною моделлю. Найкращий результат за основною метрикою RMSE (**1168.79**) було досягнуто при використанні конфігурації

кластеризації GMM(k=3). Цей показник на **2.0%** кращий, ніж результат найкращого контрольного ансамблю (**1192.67**), та значно перевершує результат найкращої одиночної моделі (**1303.20**).

Отримані результати дають підстави припустити, що попереднє розділення даних на кластери допомагає алгоритмам регресії краще адаптуватися до прихованих закономірностей у масиві даних. Успіх методу гаусових сумішей (GMM) з трьома кластерами може свідчити про те, що розподіл цін має певні неявні сегменти (наприклад, камені низької, середньої та високої цінової категорії), межі між якими не є абсолютно жорсткими. Оскільки метод GMM обчислює ймовірність належності до кластера (м'яка кластеризація), імовірно, це дозволило фінальному ансамблю більш гнучко враховувати характеристики об'єктів, що знаходяться на межі різних цінових сегментів.

Більш детальний аналіз інших метрик виявляє додаткові закономірності. Найкращий результат за метрикою середньої абсолютної похибки MAE (**389.98**) було отримано з використанням конфігурації кластеризації K-Means(k=4). Це покращення складає **7.6%** порівняно з найкращим контрольним ансамблем (**421.91**). Зменшення MAE є суттєвим показником, оскільки ця метрика відображає середнє відхилення прогнозу від реального значення без надмірного штрафування за окремі великі помилки. Можна припустити, що алгоритм K-Means, який формує більш жорсткі та чітко розділені групи, допоміг локальним моделям (експертам сегментів) точніше налаштуватися на типові представники своїх груп, що в результаті зменшило загальну середню помилку системи. Коефіцієнт детермінації (R2) також демонструє стабільне зростання для всіх конфігурацій збагачених даних (до 0.872 порівняно з **0.866** у контролі), що підтверджує здатність нових просторів ознак краще пояснювати дисперсію цільової змінної.

Для наочної візуалізації переваги запропонованого методу на рис. 3.3 та рис. 3.4 продемонстровано порівняння значень метрик RMSE та MAE для найкращих ансамблевих моделей.

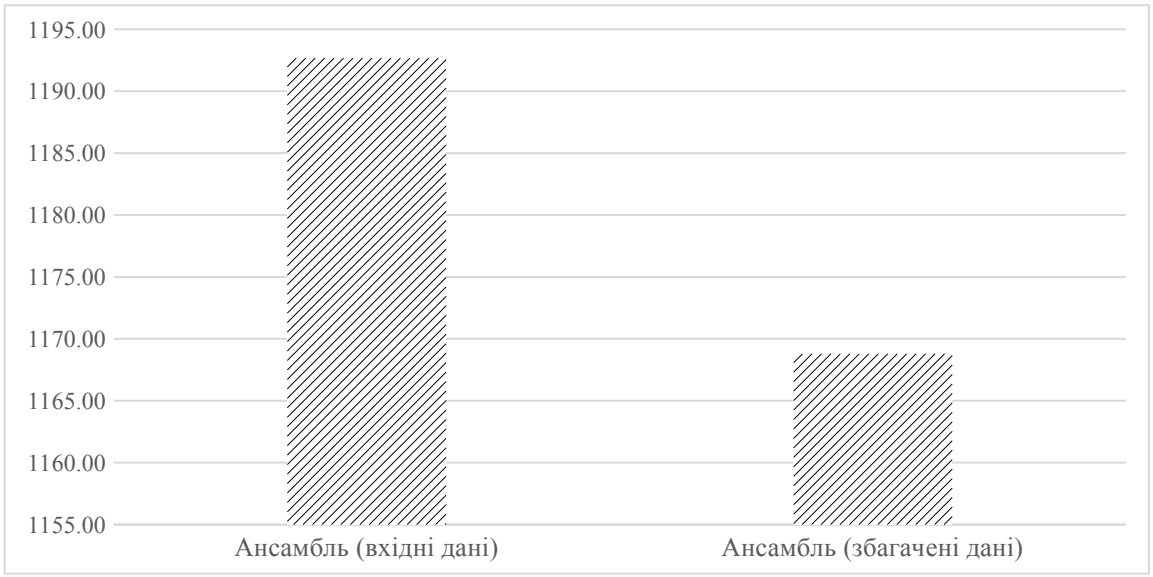


Рисунок 3.3 — Середньоквадратична похибка (RMSE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних “Дорогоцінні камені»

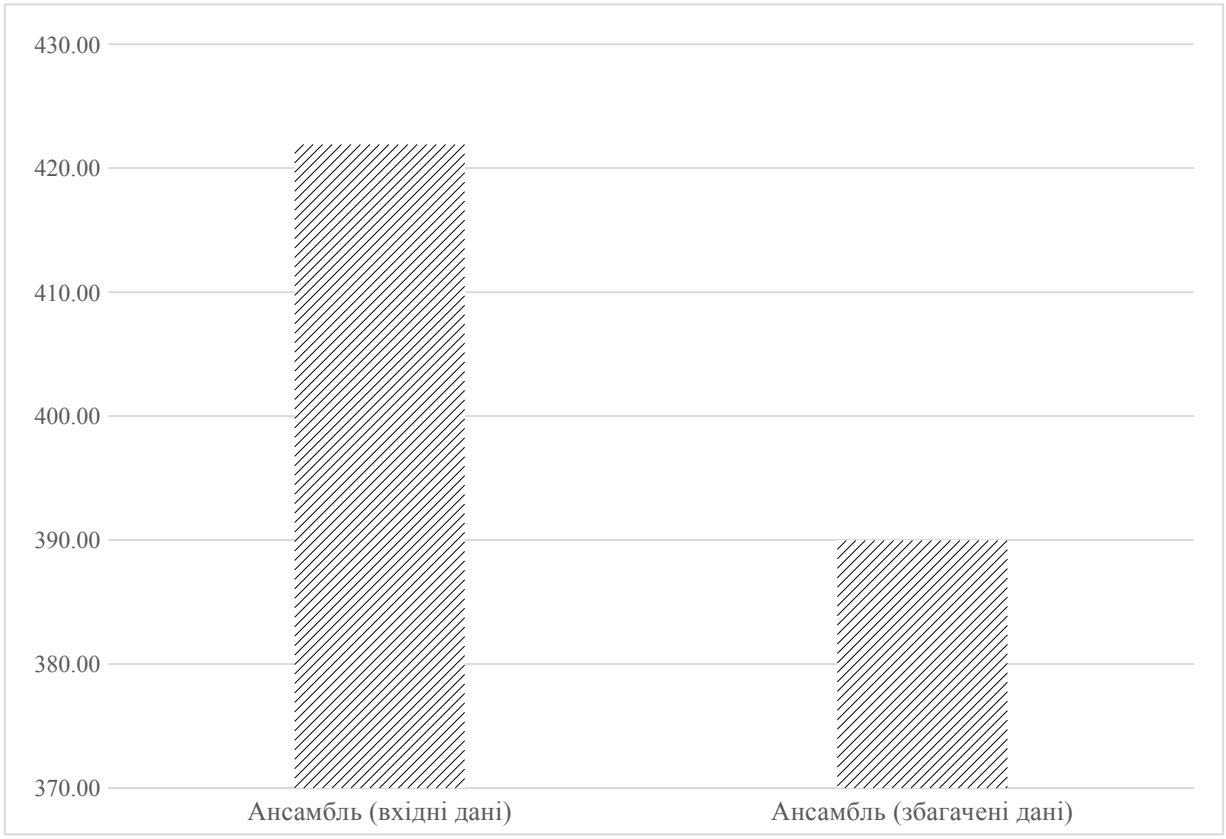


Рисунок 3.4 — Середня абсолютна похибка (MAE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних “Дорогоцінні камені»

3.2.1.3. Результати для набору даних «Маркетингові показники»

Третій набір даних, що містить маркетингові показники та інформацію про взаємодію клієнтів з кампаніями , підтвердив загальну ефективність запропонованого методу. Ключові результати для цього набору даних представлено в табл. 3.3.

Таблиця 3.3 — Порівняння результатів моделей на наборі даних «iFood Marketing»

Архітектура	Метод кластеризації	Фінальний алгоритм (ACM)	RMSE	MAE	R2
Контрольні моделі (базовий масив даних)					
Одиночна	Відсутня	ExtraTree	6588.26	4930.06	0.900
Ансамбль	Відсутня	Ансамбль(5)	6522.51	4894.28	0.902
Запропонований метод (масив зі збагаченими ознаками)					
Одиночна	GMM(k=6)	GB	6435.20	4915.60	0.904
Ансамбль	GMM(k=6)	Ансамбль (1)	**6324.73**	**4715.39**	**0.908**
Ансамбль	GMM(k=6)	Ансамбль (5)	6365.83	4842.45	0.906
Ансамбль	K-Means(k=4)	Ансамбль (1)	6405.43	4972.82	0.905

Аналіз результатів. Для цього набору даних запропонований метод також демонструє стабільне покращення точності, хоча і з меншим відривом, ніж на попередніх датасетах, що вказує на різну чутливість даних до запропонованої

інженерії ознак. Найкращий ансамбль на даних зі збагаченим словником ознак (з конфігурацією GMM(k=6) та розміром ансамблю 1) досяг показника **RMSE** 6324.73, що на **3.0%** краще за контрольний ансамбль (6522.51).

Покращення за метрикою **MAE** для цього ж ансамблю є ще більш вираженим і складає **3.7%** (з 4894.28 до 4715.39). Це підтверджує тенденцію, що запропонований метод ефективно зменшує середню абсолютну помилку.

Патерн поведінки одиничних моделей тут схожий на той, що спостерігався на першому наборі даних («Дорогоцінні камені»). Збагачення ознак дозволило покращити **RMSE** на **2.3%** (з 6588.26 до 6435.20), але майже не вплинуло на **MAE** (зменшення на 0.3% у найкращої одиночної моделі на даних зі збагаченим словником ознак, що знаходиться в межах статистичної похибки). Це знову ж таки підсилює гіпотезу про те, що основна перевага запропонованого методу розкривається при поєднанні збагачення словника ознак з подальшим ансамблюванням, яке ефективно використовує ці нові, складні характеристики. Високі результати, отримані з різними конфігураціями кластеризації на всіх трьох наборах даних, свідчать про робастність та узагальнюючу здатність запропонованого методу.

Для наочної візуалізації переваги запропонованого методу на рис. 3.5 та рис. 3.6 продемонстровано порівняння значень метрик RMSE та MAE для найкращих ансамблевих моделей.

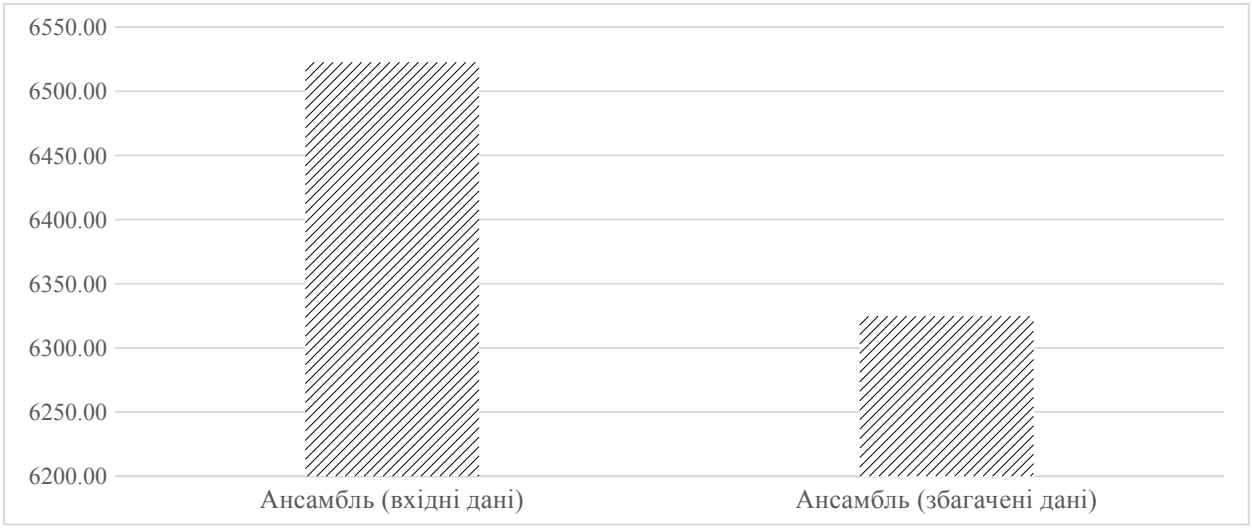


Рисунок 3.5 — Середньоквадратична похибка (RMSE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних «Маркетингові показники»

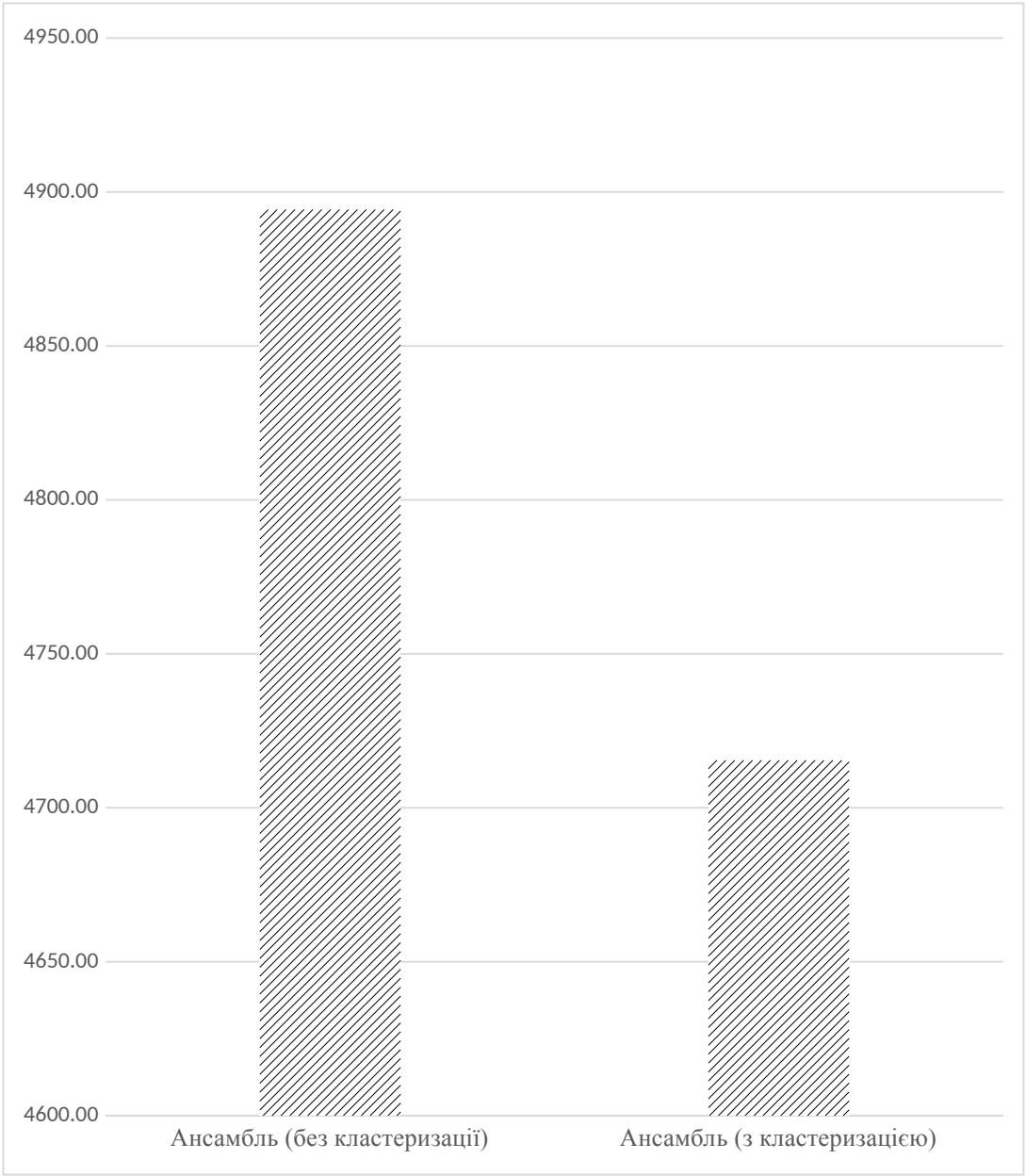


Рисунок 3.6 — Середня абсолютна похибка (MAE) серед ансамблів побудованих на вхідних даних і на збагачених даних для набору даних «Маркетингові показники»

3.2.2. Обговорення результатів та подальші напрями досліджень

Проведені експерименти та аналіз отриманих результатів дозволяють зробити кілька узагальнюючих висновків щодо ефективності та особливостей запропонованого методу, а також визначити перспективні напрямки для її подальшого розвитку.

3.2.2.1. Синтез та інтерпретація основних результатів

Експериментальні дані, отримані на трьох різнорідних наборах даних, послідовно підтверджують ефективність запропонованого метод інженерії ознак на основі попередньої кластеризації дозволяє суттєво підвищити точність фінальних моделей машинного навчання. Приріст якості, хоч і варіюється залежно від специфіки даних (від **3.0%** до **27.9%** за метрикою RMSE для ансамблів), стабільно спостерігається у всіх розглянутих випадках.

Ключовим спостереженням є те, що найбільший синергетичний ефект досягається саме при поєднанні двох підходів: збагачення словника ознак масиву вхідних даних новими мета-ознаками та подальшого застосування ансамблевих методів (рециркуляції). Хоча збагачення словника ознак і для одиночних моделей у більшості випадків дає певний приріст точності, саме ансамбль, навчений на даних зі збагаченим словником ознак, здатен найефективніше використати цю нову, складну інформацію. Це можна пояснити тим, що фінальна модель, особливо якщо вона базується на ансамблях дерев прийняття рішень (як XGBoost або Random Forest), перетворюється на потужний мета-алгоритм, що виконує інтелектуальний аналіз і обробку даних

. Вона навчається не лише на вихідних ознаках, а й на «рекомендаціях» від ансамблю спеціалізованих моделей для сегментів та класифікатора кластерів, ефективно визначаючи, прогнозам яких «локальних експертів» варто довіряти для кожного конкретного екземпляра даних.

Різна величина покращення на різних наборах даних, ймовірно, пов’язана з виразністю внутрішньої кластерної структури. На наборі даних «Прогноз вартості поїздок «, де, очевидно, існують чітко виражені сегменти (наприклад, поїздки в різний

час доби або на різні відстані), явне виділення цих сегментів дало найбільший приріст. Для інших наборів даних, де кластери можуть більше перекриватися, ефект є меншим, але все одно статистично значущим.

3.2.2.2. Обмеження методу та подальші напрями досліджень

Незважаючи на продемонстровану ефективність, запропонований підхід має певні обмеження, які, водночас, відкривають шляхи для подальших досліджень:

1. **Обчислювальна складність.** Запропонований метод є обчислювально витратним. Вона вимагає послідовного навчання великої кількості моделей: для кожного кластера, для класифікатора кластерів, а потім для фінальної моделі та її ансамблів. Це може стати обмеженням при роботі з дуже великими наборами даних. **Майбутні дослідження** можуть бути спрямовані на оптимізацію цього процесу, наприклад, через використання більш простих АСМ для побудови моделей для сегментів або через розробку методів, що дозволяють оновлювати моделі інкрементально, без повного перенавчання.

2. **Вибір оптимальної конфігурації кластеризації.** Результати показують, що вибір алгоритму та параметрів кластеризації впливає на кінцеву точність. У даній роботі оптимальна конфігурація знаходилася шляхом вичерпного перебору. **Перспективним напрямком** є розробка стратегій для автоматичного або адаптивного вибору оптимальних параметрів кластеризації на основі характеристик самого набору даних, що дозволить уникнути повного перебору та значно прискорить процес та дасть розвиток застосуванню інтелектуального аналізу даних.

3. **Можливість застосування до задач класифікації.** Запропоновані методи були перевірені лише для проблеми регресії. Адаптація методу для задач класифікації потребуватиме використання інших метрик якості (наприклад, F1-score, AUC-ROC) та, можливо, врахування проблеми незбалансованих класів, яка може по-різному проявлятися в різних сегментах. **Подальший розвиток** цього дослідження включатиме перевірку підходу на задачах класифікації.

4. **Статичність кластерної структури.** Поточна методологія припускає, що структура кластерів, виявлена на тренувальній вибірці, є статичною. У реальних системах, що працюють з потоками даних, ця структура може змінюватися з часом (дрейф концепції). **Важливим напрямком майбутньої роботи** є розробка динамічних версій методу, які б дозволяли адаптувати не лише фінальні моделі, а й саму кластерну структуру в режимі реального часу.

3.3. Удосконалений метод рециркуляції

3.3.1. Отримані результати

Для кожного вибраного набору даних було проведено моделювання запропонованої удосконаленого методу, використовуючи параметри алгоритмів синтезу моделей, визначених під час етапу параметричної оптимізації та параметрів багатошарової структури, можливі значення були визначені пошуковим простором для побудови запропонованої модельної структури та її оцінки за допомогою обчислення визначених метрик на тестувальному масиві даних. Для повної перевірки можливих комбінацій багатошарової структури було натреновано 1537 моделей для кожного обраного масиву даних, для кожної з моделей були обчислені вибрані метрики. Окремо були обчислені метрики для кожного рівня рециркуляції для можливості вибору рівня, що показує найкращий показник цільової метрики. Цільовою метрикою під час експерименту було вибрано середньоквадратичну похибку (RMSE). Кожна таблиця, наведена нижче, використовує таку структуру:

1. Значення метрик, обчислених на тестувальному масиві даних, використовує кожну з базових моделей (моделей початкового рівня).

2. Значення метрик, обчислених на тестувальному масиві даних, використовуючи моделі, побудовані на 1 рівні поєднання результатів найкращих моделей, наведено значення для найкращої конфігурації. Дані показники будуть показувати результати використання наявного підходу простого ансамблю моделей

3. Значення метрик, обчислених на тестувальному масиві даних, використовують багатошарову структуру, запропоновану в гіпотезі.

3.3.1.1. Результати для набору даних «Маркетингові показники»

У табл. 3.4 наведені результати експерименту, проведені на основі масиву вхідних даних «iFood Marketing».

Таблиця 3.4 — Результати використання моделей на тестувальному наборі даних для масиву вхідних даних «Маркетингові показники»

Тип моделі	Алгоритм	RMSE	MAE	R2
Базова	Gradient Boosting	6547,85	4970,89	0,901
	XGBoost	6705,34	5128,14	0,896
	LGBM	6650,44	5186,61	0,898
	AdaBoost	7638,37	6097,84	0,865
	Decision Tree	8404,28	6636,45	0,837
	Random Forest	6997,45	5241,40	0,887
	ExtraTree	6544,76	4980,31	0,901
	Ridge	7469,88	5844,33	0,871
	Lasso	7410,47	5792,83	0,873
	Elastic Net	7620,35	5919,98	0,866
	MLP	6425,02	5032,47	0,905
Простий ансамбль	N=5 MLP	6425,02	5032,47	0,905
Удосконалена рециркуляція	N5-M4-R7 ElasticNet	**6302,43**	**4800,13**	**0,908**
	N3-M4-R7 ElasticNet	6311,91	4811,51	0,908
	N5-M4-R6 Elastic Net	6326,20	4836,18	0,908

З даних, наведених у табл. 3.4, видно, що серед ізольованих базових моделей найкращу прогнозну здатність продемонструвала нейронна мережа MLP (RMSE = 6425.02). Однак застосування розробленої архітектури багатошарової рециркуляції дозволило системі вийти за межі можливостей одиночних алгоритмів: середньоквадратична похибка (RMSE) зменшилася на **1.9%**, а середня абсолютна похибка (MAE) — на 4.6% (з 5032.47 до 4800.13). Конфігурація простого ансамблю не дала приросту точності порівняно з базовою моделлю MLP, що підкреслює необхідність саме глибоких ієрархічних структур для складних датасетів.

Отримані результати виявляють ключову архітектурну закономірність запропонованого методу — еволюцію типу оптимального алгоритму залежно від глибини шару. На нульовому (базовому) рівні алгоритми працюють із початковим простором ознак, де існують складні нелінійні залежності між маркетинговими стимулами та цільовою змінною. Саме тому на цьому етапі домінують складні нелінійні екстрактори ознак — багатошаровий перцептрон (MLP) або ансамблі дерев рішень. Лінійні моделі (Ridge, Lasso) на базовому шарі показують найгірші результати.

Проте на вищих шарах рециркуляції (аж до оптимальної глибини R=7) статистична природа вхідних даних суттєво змінюється. Вхідними ознаками для моделей глибоких шарів стають прогнози, згенеровані попередніми моделями. Ці мета-прогнози мають надзвичайно високу кореляцію між собою, що породжує проблему мультиколінеарності. Складні алгоритми на основі дерев рішень у таких умовах швидко перенавчаються, хибно інтерпретуючи шум у мета-ознаках.

Саме тут розкривається перевага гетерогенного підходу: синтезатор автоматично віддає перевагу лінійним моделям з регуляризацією (ElasticNet). Завдяки L1/L2 штрафам, ElasticNet ефективно згладжує висококорельовані мета-ознаки та фільтрує найменш інформативні прогнози «слабких» локальних моделей, що є прикладом інтелектуального аналізу даних. Цей факт підтверджує, що основна частина обчислювально складної нелінійної роботи виконується на перших етапах

конвеєра, тоді як завданням глибоких шарів залишається зважена, регуляризована агрегація. На рис. 3.7 продемонстровано значення середньоквадратичної похибки для різних типів моделей. Для кожного типу було вибрано три моделі з найменшими значеннями даної метрики.

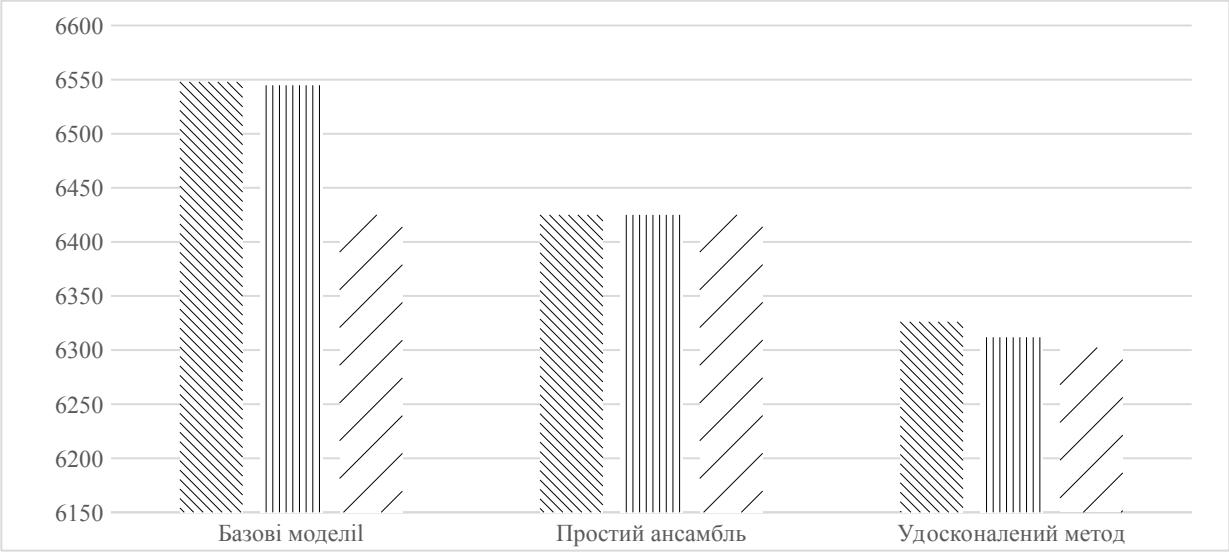


Рисунок 3.7 — Середньоквадратичне відхилення серед моделей різних типів для масиву вхідних даних «Маркетингові показники» (зліва на право) — базові моделі, простий ансамбль, удосконалений метод

3.3.1.2. Результати для набору даних «Прогноз вартості на поїздки»

У табл. 3.5 наведені результати експерименту, проведені на основі масиву вхідних даних « Прогноз вартості на поїздки «

Таблиця 3.5 — Результати використання моделей на тестувальному наборі даних для масиву вхідних даних «Прогноз вартості на поїздки»

Тип моделі	Алгоритм	RMSE	MAE	R2
Базова	Gradient Boosting	10,919	5,637	0,949
	XGBoost	14,424	6,186	0,911
	LGBM	**10,517**	5,316	**0,953**
	AdaBoost	17,047	12,061	0,876

	Decision Tree	16,967	10,132	0,877
	Random Forest	11,579	6,202	0,943
	ExtraTree	10,628	5,855	0,952
	Ridge	17,058	9,913	0,876
	Lasso	17,177	9,843	0,874
	Elastic Net	17,195	9,860	0,874
	MLP	11,790	**4,711**	0,941
Простий ансамбль	N=4 Gradient Boosting	9.695	5,491	0,96
Ансамбль за удосконаленим алгоритмом	**N2-M1-R7 AdaBoost**	**8,154**	5,616	**0,972**
	N2-M1-R5 Gradient Boosting	8,205	**4,689**	0,971
	N2-M3-R6 LGBM	8,491	4,981	0,969

З результатів, наведених у табл. 3.5, встановлено, що серед базових моделей найменшу цільову похибку (RMSE) має алгоритм LGBM (10.517). Використання простого ансамблю дозволило покращити цей показник до **9.695**. Водночас застосування запропонованої багатошарової структури рециркуляції забезпечує зменшення середньоквадратичної похибки до **8.154**. Це свідчить про зниження похибки на **15,9 %** порівняно з простим ансамблем та на **22,4 %** порівняно з найкращою базовою моделлю.

Отримані результати підтверджують доцільність адаптивного вибору параметрів структури ансамблю. Оптимальними значеннями ширини виявилися $N = 2$ та $M = 1$. Це означає, що на початковому етапі алгоритм використовує дві моделі, а на наступних прихованих шарах передає для подальшої обробки результати лише

однієї найточнішої моделі. Така конфігурація дозволяє системі автоматично обмежувати обчислювальну надлишковість та відкидати найменш інформативні ознаки у процесі рециркуляції.

Крім того, експеримент демонструє, що на різних рівнях багатошарової структури найвищу ефективність показують різні алгоритми машинного навчання. Наприклад, якщо на базовому рівні найменшу похибку мав алгоритм LGBM, то для ролі фінальної моделі на найглибшому шарі ($R=7$) оптимальним виявився AdaBoost. Ця зміна ефективності пояснюється тим, що зі зростанням глибини рециркуляції змінюються статистичні властивості масиву даних. Відповідно, використання різнотипних алгоритмів на різних шарах є обґрунтованим, оскільки єдиний алгоритм не може бути універсальним для всіх етапів перетворення ознак.

Для наочної візуалізації цих результатів на рис. 3.8 продемонстровано порівняння значень середньоквадратичної похибки для найкращих моделей кожної з трьох розглянутих архітектур.

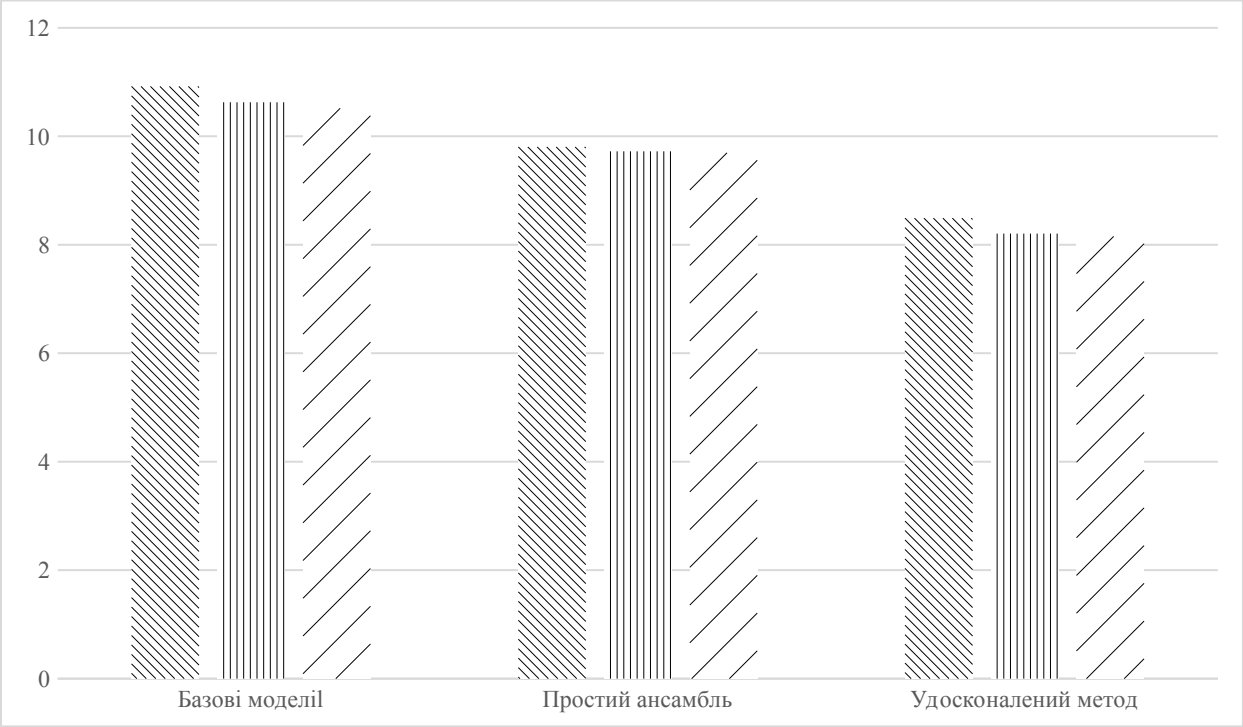


Рисунок 3.8 — Середньоквадратичне відхилення результатів моделювання для масиву вхідних даних «Прогноз вартості на поїздки» (зліва на право) — базові моделі, простий ансамбль, удосконалений метод

3.3.1.3. Результати для набору даних «Дорогоцінні камені»

У табл. 3.6 наведені результати експерименту, проведені на основі масиву вхідних даних «Дорогоцінні камені».

Таблиця 3.6 — Результати використання моделей на тестувальному наборі даних для масиву вхідних даних « Дорогоцінні камені «

Тип моделі	Алгоритм	RMSE	MAE	R2
Базова	Gradient Boosting	1260,71	441,24	0,851
	XGBoost	1322,25	431,04	0,836
	LGBM	1258,66	456,04	0,851
	AdaBoost	1700,79	879,20	0,728
	Decision Tree	1452,70	524,42	0,802
	Random Forest	**1244,48**	427,61	0,854
	ExtraTree	1299,25	**399,36**	0,841
	Ridge	1888,82	1014,85	0,665
	Lasso	1883,36	1010,51	0,667
	Elastic Net	1839,91	1004,36	0,682
	MLP	1253,99	489,33	0,852
Простий ансамбль	N=3 LGBM	1120,23	405,19	0,882
Ансамбль за удосконаленим	N2-M2-R7 Elastic Net	**1072,40**	427,33	**0,892**
	N2-M3-R3 Elastic Net	1090,06	432,24	0,888

алгоритмом	N2-M2-R3 LGBM	1091,12	448,62	0,888

З даних, наведених у табл. 3.6, встановлено, що серед базових одиночних моделей найменшу цільову похибку (RMSE) продемонстрував алгоритм Random Forest (1244.48). Формування простого ансамблю дозволило покращити цей показник до 1120.23. Застосування розробленої багатошарової структури рециркуляції забезпечило значення середньоквадратичної похибки на рівні 1072.40. Це вказує на зниження похибки **на 4.3 %** порівняно з простим ансамблем та на **13.8 %** порівняно з найкращою базовою моделлю.

Отримані результати ілюструють вплив структурних параметрів ансамблю. Оптимальними значеннями ширини на даному масиві виявилися $N=2$ та $M=2$. Така конфігурація означає, що система передає на кожен наступний прихований шар прогнози лише двох моделей з найменшою похибкою, відсікаючи інші результати.

Також результати на цьому наборі даних мають спільну ознаку з експериментом на масиві «Маркетингові показники»: на фінальному рівні рециркуляції ($R=7$) найменшу похибку показала регуляризаційна модель Elastic Net. Наявність такої подібності на двох структурно різних наборах даних (один містить значну кількість категоріальних ознак, інший складається переважно з числових) дає підстави припустити наявність загальної властивості багатошарової обробки.

При переході до більш глибоких шарів рециркуляції вхідні дані (прогнози попередніх моделей) набувають високого ступеня кореляції між собою. За таких умов алгоритми з вбудованою лінійною регуляризацією (як Elastic Net) демонструють вищу стабільність, оскільки вони здатні математично згладжувати вплив мультиколінеарності. Це пояснює їхню перевагу над складнішими нелінійними алгоритмами (наприклад, деревами рішень) на пізніх етапах побудови ансамблю.

Для наочної візуалізації динаміки зниження похибки на рис. 3.9 продемонстровано порівняння значень RMSE для моделей кожної з трьох розглянутих архітектур.

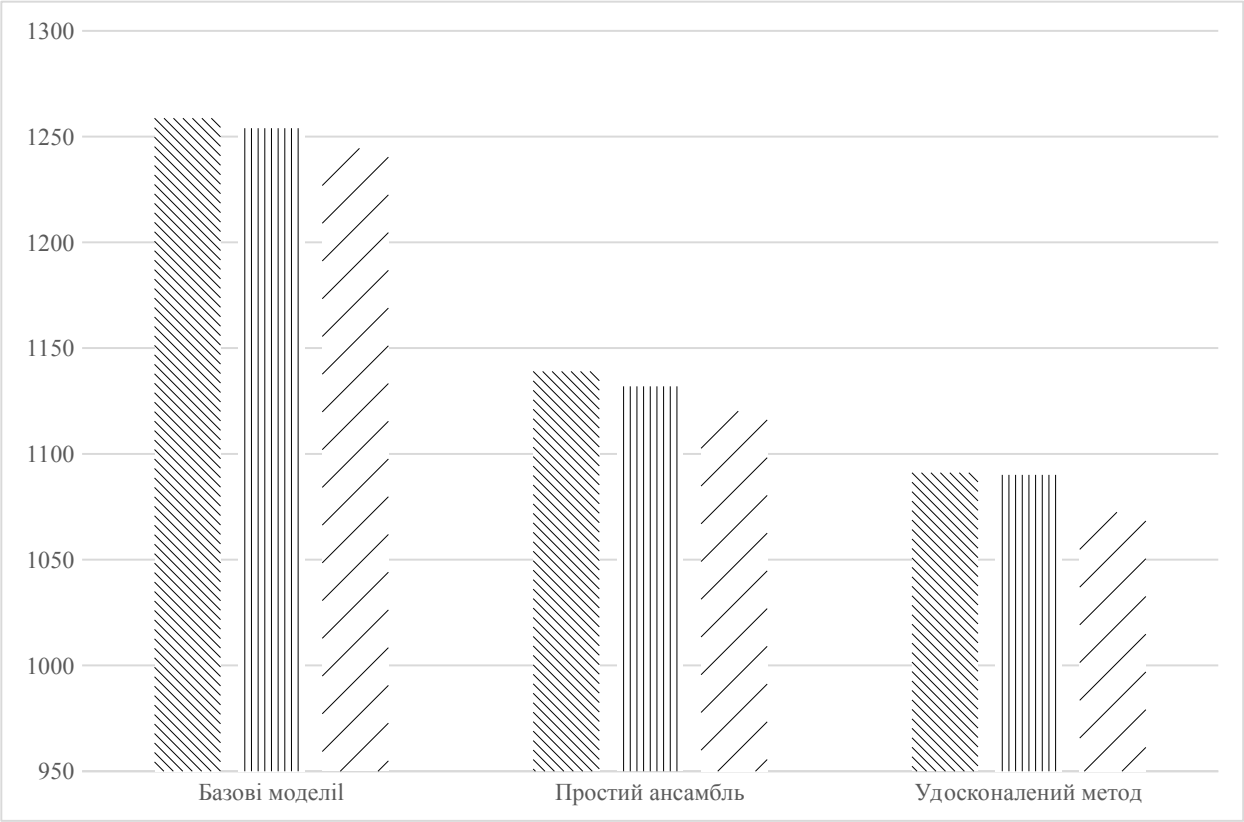


Рисунок 3.9 — Середньоквадратичне відхилення результатів моделювання для масиву вхідних даних «Дорогоцінні камені» (зліва на право) — базові моделі, простий ансамбль, удосконалений метод

3.3.2. Обговорення результатів та подальші напрями досліджень

Для узагальнення результатів експериментального дослідження багатошарової рециркуляції, у табл. 3.7 наведено зведені показники відносного зменшення похибки порівняно з класичними ансамблями.

Таблиця 3.7 — Результати використання удосконаленого методу

Масив вхідних даних	Зменшення середньоквадратичної похибки (відносно простого ансамблю)
Маркетингові показники	1,9 %
Прогноз вартості на поїздки	15,9 %
Дорогоцінні камені	4,2 %

Загалом результати підтверджують інженерну доцільність удосконаленого методу рециркуляції. Головним висновком є експериментальне доведення ефективності використання різноманітного (гетерогенного) пулу алгоритмів на різних рівнях структури. Надання системі доступу до різнотипних методів машинного навчання дозволяє компенсувати математичні обмеження окремих алгоритмів (наприклад, схильність дерев рішень до перенавчання на глибоких шарах) і забезпечує стабільність фінального прогнозу.

Водночас, запропоноване удосконалення процесів синтезу багатошарової структури має певні інженерні обмеження. Для їх усунення подальші дослідження можуть бути спрямовані на вирішення таких завдань:

1. Оптимізація обчислювальної складності. Процес конструювання багатошарової структури вимагає послідовного тренування пулу моделей на кожному рівні. Це висуває підвищені вимоги до апаратного забезпечення (обсягу оперативної пам’яті для зберігання проміжних мета-ознак та обчислювальних потужностей). Це може стати обмеженням у системах, що вимагають адаптації моделі в режимі реального часу. Перспективним напрямком є скорочення кількості кандидатів АСМ на глибоких шарах або використання алгоритмів поступового навчання.

2. Розширення класів задач (Класифікація). Запропоновані методи були верифіковані на задачах визначення регресійної залежності. Перенесення методу на задачі класифікації (наприклад, визначення статусів системи) вимагатиме використання відповідних цільових метрик (F1-score, AUC-ROC) та імплементації механізмів обробки незбалансованих класів. Крім того, подальшого вивчення потребує вплив методів кодування категоріальних ознак на ефективність різних алгоритмів у глибоких шарах рециркуляції.

3.4. Висновки до розділу 3

1. Експериментальні дослідження, проведені на трьох структурно відмінних відкритих наборах даних, підтвердили доцільність розроблених методів. Доведено,

що використання різноманітного набору алгоритмів у багатошарових структурах підвищує загальну ефективність прогнозних моделей моніторингового агента порівняно з однотипними ансамблями.

2. Експериментально підтверджено результативність методу підвищення однорідності вхідних даних. Створення масивів із високою внутрішньою однорідністю шляхом кластеризації з подальшою генерацією мета-ознак (прогнозів спеціалізованих локальних моделей та класифікатора кластерів) дозволило зменшити середньоквадратичну похибку (RMSE) на величину від 3.0 % до 27.9 % (залежно від топології набору даних). Встановлено, що вибір алгоритму кластеризації впливає на кінцевий результат, що обґрунтовує доцільність використання автоматизованого перебору стратегій сегментації.

3. Доведено дієздатність концепції удосконаленого методу рециркуляції. Залучення різнотипних алгоритмів синтезу моделей (АСМ) на кожному ітеративному шарі забезпечило додаткове зниження похибки прогнозування на 1.9 % … 15.9 % порівняно з класичними статичними ансамблями.

4. Виявлено архітектурну закономірність: алгоритми синтезу демонструють різну прогнозну здатність залежно від глибини. Експериментально підтверджено, що лінійні регуляризаційні алгоритми (зокрема Elastic Net) показують високу ефективність як агрегатори на вищих (глибоких) шарах рециркуляції, оскільки вони зменшують вплив мультиколінеарності мета-ознак. Це дозволяє оптимізувати обчислювальну логіку фінальних етапів синтезу.

5. Отримані результати підтверджують можливість використання розроблених методів для побудови автономних агентних моделей. Водночас виявлена залежність точності від структурних параметрів та багатокрокова обчислювальна складність конвеєра формують інженерні вимоги: системі необхідні декларативні засоби для автоматизованого проектування таких структур та забезпечення їхнього детермінованого виконання (відтворюваності), програмна реалізація яких розглядається у наступному розділі.

6.

РОЗДІЛ 4
ЗАСТОСУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

4.1. Метод проектування моніторингового програмного агента

Для практичної реалізації методів адаптивного синтезу моделей (зокрема, підвищення однорідності даних та багатошарової рециркуляції) спроектовано програмний комплекс. Інженерне завдання цього етапу полягало у формуванні об’єктно-орієнтованої архітектури. Такий підхід забезпечує інтеграцію розрізнених алгоритмів машинного навчання в єдину автоматизовану систему моніторингу.

4.1.1. Моделювання предметної області та ролей взаємодії

Проектування системи базується на формалізації предметної області моніторингового агента. Ключовими сутностями (entities) домену визначено базові моделі, кластери, структурні шари рециркуляції та конвеєри обробки даних (data pipelines). Ці абстракції дозволяють формалізувати етапи трансформації даних від входу до фінального прогнозу. Управління визначеними сутностями вимагає розмежування прав доступу користувачів протягом життєвого циклу програмного забезпечення. Взаємодію трьох ключових акторів із системою відображає діаграма прецедентів, що наведена на рис. 4.1.

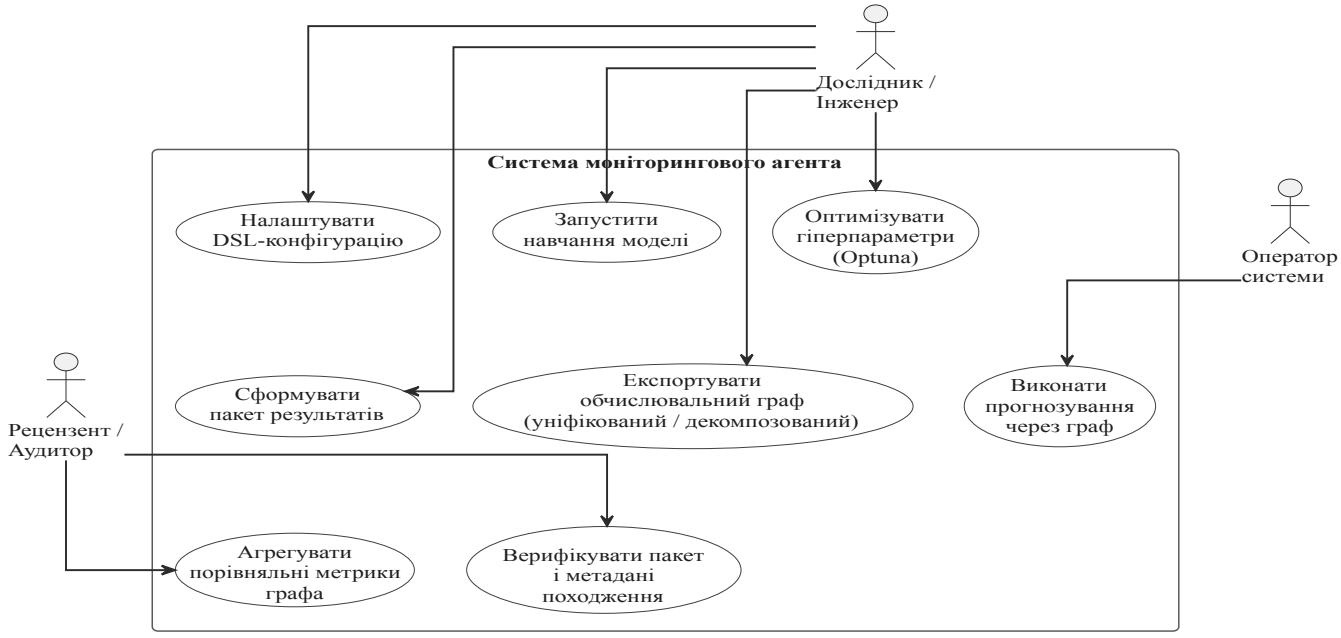


Рис. 4.1 — Діаграма прецедентів моніторингового програмного агента

Відповідно до рис. 4.1, архітектурна логіка розподіляє процеси на фазу дослідження (*research*) та фазу експлуатації (*production*). Ролі акторів формалізовано наступним чином:

- **«Дослідник/Інженер»**: відповідає за етап налаштування. Актор формує декларативну специфікацію, ініціює навчання базових моделей, керує оптимізацією гіперпараметрів та експортує фінальний обчислювальний граф (*computational graph*).
- **«Оператор системи»**: працює виключно на етапі експлуатації. Функціонал актора обмежено застосуванням вже сформованого графа для генерації прогнозів на нових даних, що абстрагує його від внутрішньої логіки навчання.
- **«Рецензент/Аудитор»**: виконує контролюючу функцію. Роль передбачає доступ до агрегованих порівняльних метрик, перевірку пакета результатів (*artifact bundle*) та аналіз метаданих походження (*provenance data*). Це забезпечує відтворюваність та прозорість експериментів.

4.1.2. Формування вимог до програмного забезпечення

Аналіз предметної області дозволив сформувати базові вимоги до програмної реалізації агента. Для забезпечення роботи багатошарових алгоритмів програмне забезпечення має підтримувати:

1. **Уніфікацію компонентів:** алгоритми кластеризації, базові моделі та агрегатори повинні мати спільний інтерфейс для взаємодії між собою.
2. **Модульність рівнів:** архітектура має підтримувати динамічне додавання або вилучення етапів обробки (наприклад, нових шарів рециркуляції). Дана вимога виключає необхідність модифікації базового вихідного коду системи.
3. **Збереження конвеєра:** необхідність фіксації всієї послідовності обробки даних (від входу до фінального прогнозу) для подальшого використання. Реалізація цих інженерних вимог природним чином спирається на можливості об’єктно-орієнтованого програмування (ООП).

4.1.3. Компонентна архітектура системи

З метою забезпечення масштабованості та ізоляції обчислювальної логіки, архітектуру системи побудовано за модульним принципом. Діаграму компонентної взаємодії програмного комплексу відображено на **рис. 4.2**.

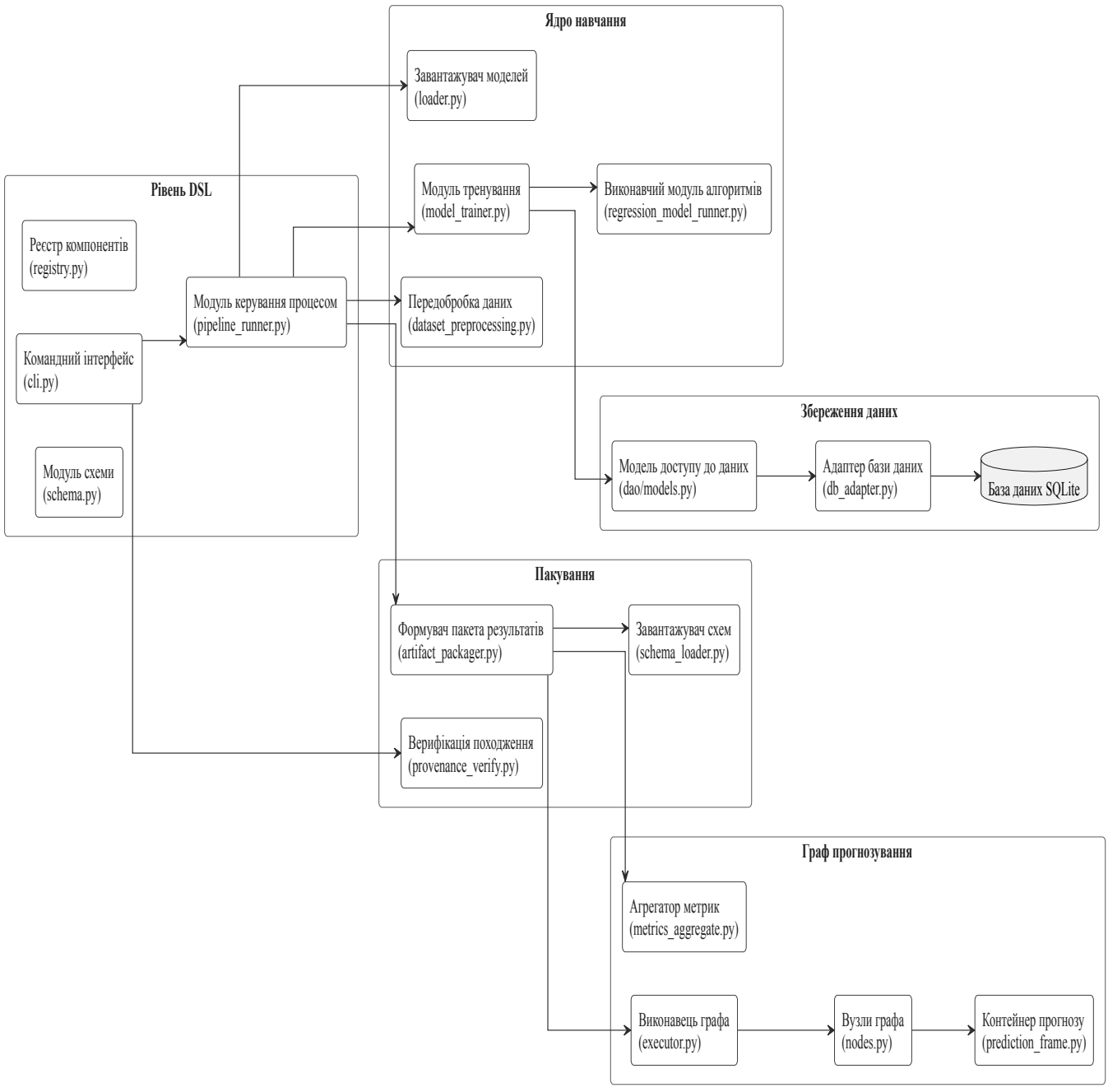


Рис. 4.2 — Діаграма компонентної взаємодії системи

Архітектура складається з п’яти взаємопов’язаних рівнів (підсистем):

1. Рівень предметно-орієнтованої мови (*Domain-Specific Language, DSL*): слугує точкою входу до системи через інтерфейс командного рядка (*CLI*). Підсистема виконує парсинг декларативних налаштувань, валідацію конфігураційної схеми та ініціює загальне управління процесом навчання**.**

2. Ядро навчання (*Training Core*): реалізує алгоритми попередньої обробки даних та логіку алгоритмів машинного навчання. Підсистема безпосередньо керує виконавчими модулями алгоритмів (*model runners*) на основі команд, отриманих від модуля тренування.

3. Підсистема збереження даних (*Data Persistence*): забезпечує збереження проміжних та фінальних результатів. Обчислені метрики, параметри кластеризації та шляхи до навчених моделей зберігаються у локальній реляційній базі даних (SQLite) за допомогою спеціалізованих адаптерів та об'єктів доступу до даних (*Data Access Objects, DAO*).

4. Граф прогнозування (*Inference Graph*): активується на етапі експлуатації моделі. Підсистема завантажує попередньо збережені вузли обчислювального графа та виконує їхню послідовну ініціалізацію для генерації нових прогнозів.

5. Підсистема пакування (*Packaging*): формує фінальний пакет результатів. Цей компонент об'єднує серіалізовані моделі, опис топології графа та метадані, необхідні для верифікації походження результатів (*data provenance*).

4.1.4. Об'єктно-орієнтована архітектура класів

Важливим етапом проектування є розробка логічної структури класів для представлення складних ансамблів як єдиного обчислювального конвеєра. Використання об'єктно-орієнтованого підходу дозволило стандартизувати програмні компоненти агента. Наприклад, окремі моделі та агрегатори шарів рециркуляції реалізовані як об'єкти з єдиним інтерфейсом, що спрощує передачу даних між різними рівнями структури. На рис. 4.3 зображено об'єктно-орієнтовану архітектуру класів

ядра навчання та графа прогнозування.

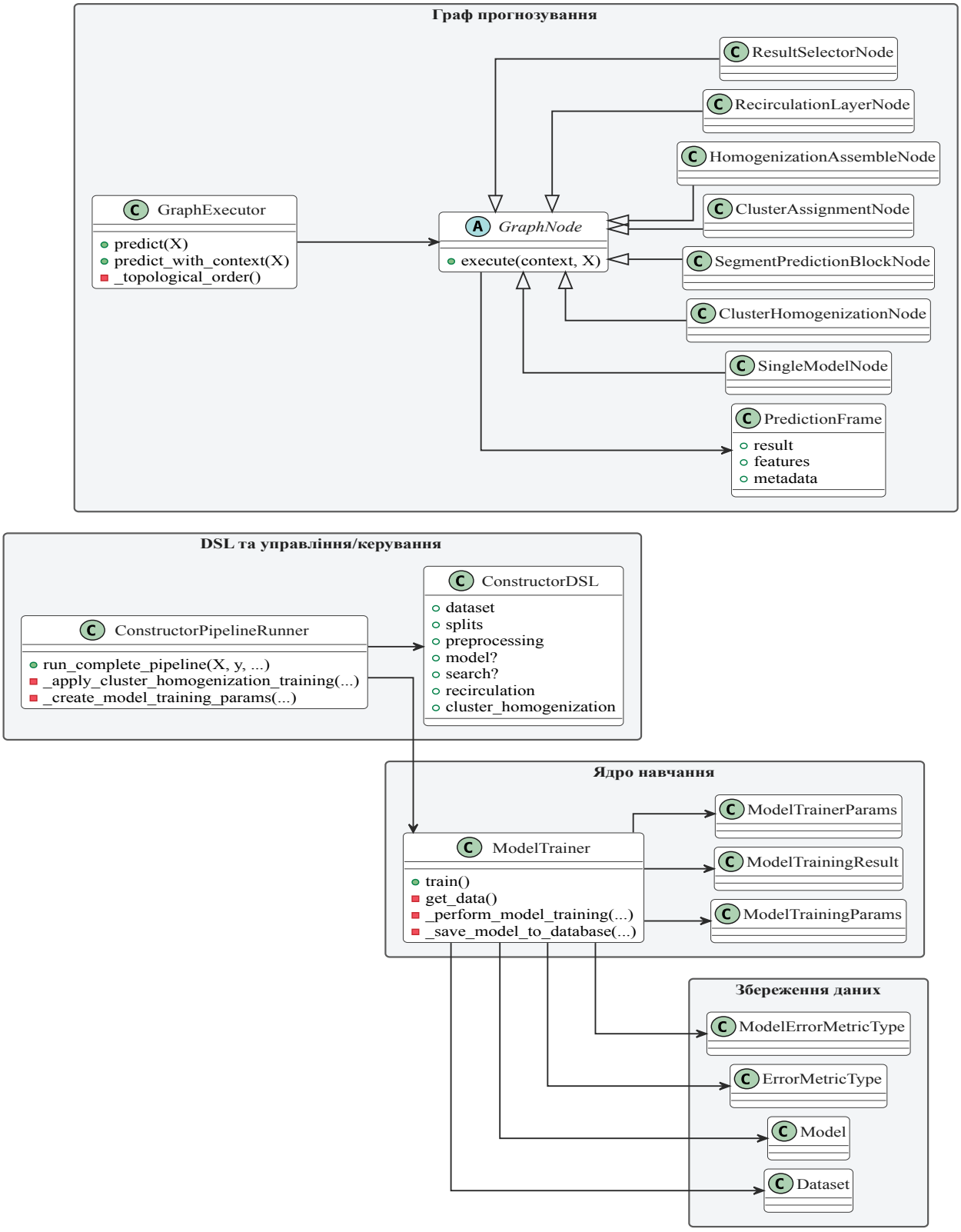


Рис. 4.3 — Логічна архітектура класів системи

Основним патерном проектування графа прогнозування є використання базового абстрактного класу GraphNode. Від нього успадковуються класи, що відповідають за виконання конкретних операцій:

- SingleModelNode — виконання атомарної базової моделі;
- ClusterAssignmentNode та SegmentPredictionBlockNode — класи, що реалізують логіку методу підвищення однорідності вхідних даних (віднесення до кластера та генерація прогнозів відповідно);
- RecirculationLayerNode — клас, що відповідає за агрегацію прогнозів на нових рівнях ієрархії;
- ResultSelectorNode — термінальний клас зваженого обчислення фінального прогнозу.

Для забезпечення уніфікованого обміну даними між вузлами введено структуру контейнера PredictionFrame. Вона інкапсулює не лише безпосередній прогноз (result), а й розширені ознаки (features) та метадані (metadata). Така структура є необхідною для передачі контексту між компонентами графу**.**

Управління зазначеним процесом здійснює клас GraphExecutor. Він програмно реалізує алгоритми топологічного сортування вузлів і виконання обчислень з урахуванням контексту попередніх кроків, що забезпечує детермінованість послідовності дій.

Таким чином, розроблена об’єктно-орієнтована та компонентна архітектура дозволяє перетворити складні багатокрокові математичні операції на програмно керований процес. Механізми безпосереднього управління цим процесом через декларативний підхід розглядаються у наступному підрозділі.

4.2. Архітектура декларативного опису конфігурації

Основою запропонованого підходу є декларативний опис конфігурації, реалізований у форматі структурованої JSON-схеми. Цей формат забезпечує формалізований та однозначний опис архітектури створюваної модельної структури для програмного агента. Використання такого підходу чітко відокремлює етап

логічного проектування від програмної реалізації, що підвищує прозорість обчислень та знижує ймовірність помилок.

Схема включає наступні структурні секції, які формалізують кожен етап конструювання:

1. **dataset** та **splits**: Визначення джерела даних та правил його розділення.
2. **preprocessing**: Конфігурація етапів передобробки для різних типів ознак.
3. **model**: Визначення базового (однорівневого) АСМ.
4. **cluster_homogenization**: Опис процесу підвищення однорідності даних.
5. **recirculation**: Конфігурація багатошарових ансамблів.
6. **search**: Налаштування гіперпараметричної оптимізації.
7. **artifacts**: Керування збереженням результатів та артефактів.

Процес конструювання модельної структури визначається послідовно. У секції dataset вказується шлях до набору даних (path), назва цільової змінної (target) та тип задачі (task_type: регресія чи класифікація). У секції splits описується стратегія формування тренувальної та тестової вибірок. Передбачено використання випадкового поділу (random) для стандартних випадків, або стратифікованого (stratified) для задач класифікації зі збалансованим збереженням пропорцій класів. Параметр test_size дозволяє керувати обсягом даних для валідації.

Наступним кроком є конфігурація етапів передобробки даних у секції preprocessing. Вона містить окремі гілки для числових та категоріальних ознак. Для числових ознак задаються методи заповнення пропусків (*NaN*), наприклад, mean (середнє значення) чи median (медіана, стійка до викидів). Визначається тип масштабування (scaler): standard (стандартизація), minmax (нормалізація до діапазону) чи robust (використання квантилів). Для даних з асиметричним розподілом передбачено застосування алгоритмів корекції (*skewness*), таких як перетворення Бокса-Кокса або Єо-Джонсона.

Для категоріальних ознак визначаються стратегії обробки пропусків (наприклад, most_frequent або виділення в нову категорію new_category) та метод кодування (encoder), наприклад, унітарне кодування (*one-hot encoding*). Інкапсуляція всієї логіки передобробки в єдиний конфігураційний файл забезпечує відтворюваність процесу, на відміну від імперативних підходів, де трансформації даних жорстко закодовані у скриптах. Наявність цих опцій у схемі дозволяє адаптувати цикл обробки під конкретну модельну структуру (наприклад, враховуючи високу чутливість лінійних моделей до масштабу ознак, порівняно з алгоритмами на основі дерев рішень).

У секції model задається базовий алгоритм синтезу моделей та його статичні параметри. Ця конфігурація використовується для навчання простої однорівневої моделі, яка виступає базовою лінією (*baseline*) для порівняння з більш складними архітектурами. Вона застосовується за замовчуванням, якщо не ініційовано секції recirculation або cluster_homogenization.

Секція cluster_homogenization дозволяє декларативно описати процес підвищення однорідності даних. Розробник визначає метод початкової кластеризації (method — наприклад, kmeans або gmm), кількість кластерів (cluster_count) та стратегію генерації нових ознак. Специфікація підтримує гнучке управління через параметри: include_segment_predictions (додавання прогнозів від спеціалізованих моделей) та include_cluster_id (додавання прогнозованої мітки кластера).

Для побудови багатошарових ансамблів призначена секція recirculation. Загальні параметри (first_layer_max_models, next_layer_max_models) дозволяють задавати кількість базових моделей для першого та наступних шарів. Для детального керування передбачено секцію *layers*, яка конфігурує кожен шар індивідуально: визначається кількість моделей (*max_models*), набір АСМ (*algorithms*) та стратегія агрегації прогнозів (gating) — просте (*average*) або зважене (*weighted*) усереднення. Цей механізм виступає декларативною реалізацією концепції стекованої генералізації (*stacked generalization*)**.**

Пошук оптимальних гіперпараметрів та збереження результатів визначаються секціями search та artifacts. Секція search дозволяє активувати оптимізацію за попередньо визначеними просторами пошуку, які можна перевизначити через параметр space. Секція artifacts відповідає за збереження результатів: вказується вихідна директорія (output_dir), статус збереження навчених моделей та параметри експорту знімка бази даних експериментів (db_export) для забезпечення відтворюваності.

4.2.1. **Метод гіперпараметричної оптимізації**.

Процес пошуку гіперпараметрів автоматизовано за допомогою фреймворку Optuna [77]. Вибір даного рішення зумовлений наявністю ефективних алгоритмів вибірки (зокрема, Tree-structured Parzen Estimator, TPE) та можливістю інтеграції у багатоетапні конвеєри навчання. Гіперпараметрична оптимізація (ГПО) ініціюється виключно за умови явного визначення ненульової кількості ітерацій (n_trials).

Підхід є повністю детермінованим: фіксоване глобальне початкове значення (random_state) передається до **внутрішніх об'єктів** Optuna, що гарантує **машинну тотожність результатів** при повторних запусках.

Структура налаштувань дозволяє застосовувати ГПО з різними лімітами ітерацій (hpo_n_trials) до окремих компонентів моделі:

1. **Глобальна оптимізація:** застосовується до основної однорівневої моделі (секція search).
2. **Локальна оптимізація:** визначення окремого обчислювального бюджету для моделей сегментів та класифікатора (секція cluster_homogenization).
3. **Пошарова оптимізація:** розподіл ресурсів на оптимізацію індивідуальних шарів ансамблю (секція recirculation), що дозволяє інтенсивніше налаштовувати ключові (наприклад, початкові) рівні структури.

4.2.2. Процес навчання, керований структурованою конфігурацією.

Спеціалізований програмний комплекс перетворює декларативний опис на навчену модельну структуру та генерує відтворюваний пакет результатів. Цей процес поділено на чотири послідовні етапи**:**

1. **Валідація та синтаксичний аналіз конфігурації:** Програмний комплекс виконує дворівневу перевірку вхідного JSON-файлу:

1. **Синтаксична валідація:** перевірка коректності типів даних та діапазонів значень (наприклад, параметр n_trials $\in [1,200]$).

2. Семантична валідація: аналіз логічних залежностей. Зокрема, система перевіряє, що кількість заявлених алгоритмів для шару рециркуляції є достатньою для відбору max_models, та що ваги для стратегії агрегації (*weighted gating*) визначені коректно. Такий підхід реалізує архітектурний шаблон «швидкого відмовлення» (*fail-fast*). Виявлення логічних помилок до ініціалізації ресурсоємних обчислень запобігає нераціональному використанню апаратних потужностей та є необхідною умовою для функціонування автономних агентів.

2. **Підготовка даних:** Здійснюється завантаження та поділ вибірки. На основі секції preprocessing формується об’єкт трансформації. Математичні параметри передобробки (середні значення, стандартні відхилення) обчислюються (*fit*) виключно на тренувальній вибірці, після чого отримані правила застосовуються (*transform*) до тестової вибірки, що унеможливлює витік даних (*data leakage*).

3. **Виконання навчання:** Якщо ініційовано cluster_homogenization, виконується кластеризація тренувальних даних із подальшим навчанням локальних моделей для кожного сегмента. За наявності секції recirculation ітеративно будуються шари ансамблю: на кожному етапі викликається синтезатор моделей, який за потреби запускає процес ГПО.

4. **Генерація результатів та метаданих:** Система агрегує навчені моделі, серіалізовані об’єкти передобробки, звіти про метрики якості та знайдені гіперпараметри, інтегруючи їх у фінальний пакет (*bundle*).

Взаємодію програмних підсистем під час виконання налаштованої конфігурації відображено на діаграмі послідовностей (рис. 4.4).

.

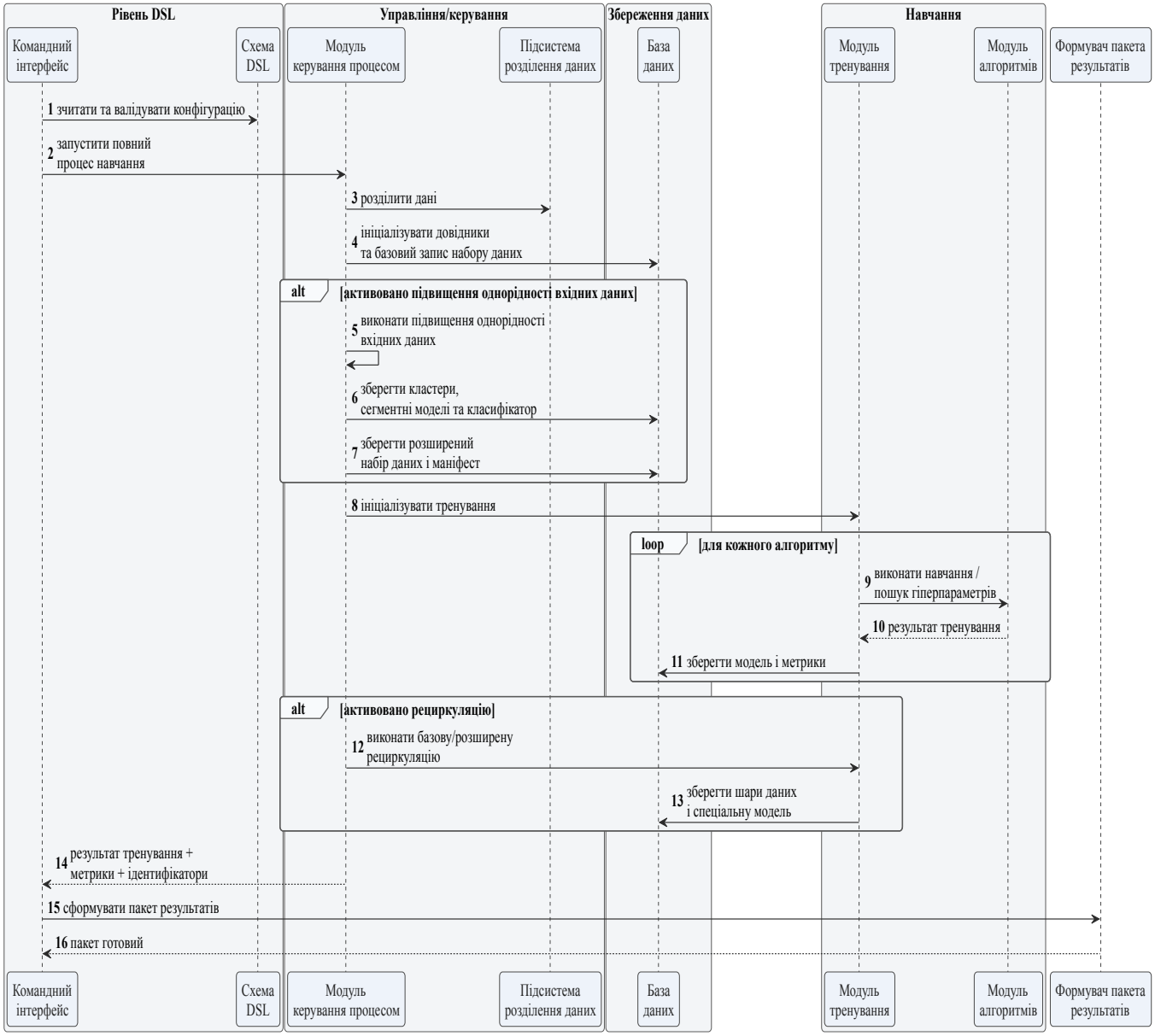


Рис. 4.4 – Діаграма послідовностей процесу управління навчанням моніторингового агента

Відповідно рис. 4.4, процес керується центральним модулем управління, який отримує перевірену конфігурацію від командного інтерфейсу та виконує роль менеджера процесу. Ключовою особливістю реалізованої логіки є динамічне

розгалуження процесу обчислень. Залежно від налаштувань, заданих користувачем у файлі конфігурації, система активує відповідні гілки виконання:

3. Якщо конфігурація передбачає підвищення однорідності, система призупиняє базове тренування, ініціює процеси кластеризації, створює локальні сегментні моделі та формує розширений набір даних, після чого повертає управління основному потоку.

4. Етап базового навчання реалізовано ітеративно: модуль тренування викликає алгоритми, оцінює їхні метрики та виконує ранжування.

5. За умови активації рециркуляції, після завершення базового навчання стартує додатковий цикл формування багатошарового ансамблю, який використовує раніше збережені прогнози як нові вхідні ознаки.

Важливим архітектурним рішенням, відображеним на діаграмі, є поступове збереження стану. Взаємодія з базою даних відбувається не в кінці всього процесу, а транзакційно — після завершення кожного логічного блоку (навчання окремої моделі, формування кластера чи шару). Це гарантує збереження проміжних результатів і стійкість системи до можливих збоїв під час тривалих обчислень. Завершується конвеєр передачею управління модулю пакування, який збирає всі збережені артефакти у фінальний відтворюваний пакет. Для наочної демонстрації цього процесу

на рис. 4.5 наведено діаграму активності. Навчання від збереження даних

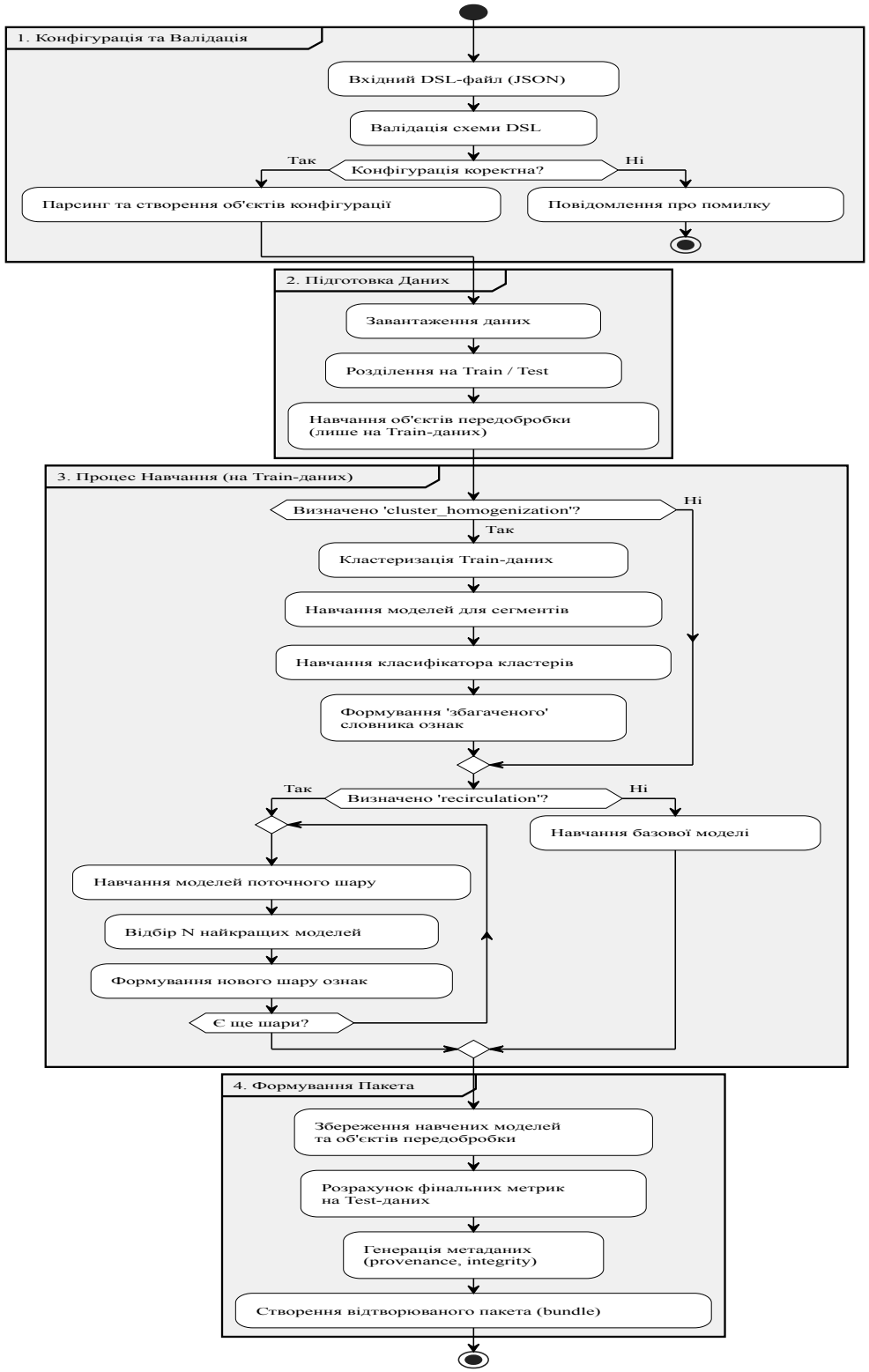


Рисунок 4.5— Діаграма активності процесу навчання на основі DSL-конфігурації

Програмний комплекс спроектовано як набір модулів, де кожен етап – від читання конфігурації до навчання моделей та пакування результатів – реалізовано як окремий компонент. Такий модульний підхід спрощує підтримку та подальше розширення системи новими алгоритмами чи функціональними можливостями. Наприклад, модуль валідації конфігурації може бути використаний автономно для перевірки JSON-схем без запуску процесу навчання. Аналогічно, модуль пакування артефактів може бути інтегрований в інші системи для збереження результатів.

Діаграма подана на рис. 4.5 ілюструє логічну послідовність ключових фаз: від початкової валідації декларативної конфігурації, через етапи підготовки даних та умовні розгалуження (залежно від наявності секцій cluster_homogenization та recirculation), до фінального етапу генерування пакета результатів.

4.2.3. Формування відтворюваного пакета.

Результатом роботи програмного комплексу є створення відтворюваного пакета (*bundle*) — архіву, що містить повну інформацію для аудиту, відтворення та експлуатації навченої модельної структури. Таке пакування вирішує проблему розгортання моделей (*model deployment*) у агентному середовищі. Пакет включає наступні компоненти:

1. **resolved_config.json:** фінальна конфігурація, яка відображає всі параметри, включаючи знайдені в ході ГПО. Цей файл слугує еталонним записом стану модельної структури.
2. **artifacts.json:** Мапа артефактів, що містить відносні шляхи до всіх збережених файлів (навчених моделей, об’єктів для передобробки, маніфестів для cluster_homogenization тощо) всередині пакета.
3. **Серіалізовані моделі та об'єкти передобробки:** бінарні файли навчених моделей (наприклад, у форматі .joblib) для завантаження в оперативну пам'ять без перенавчання.

4. **metrics.json:** структурований звіт із метриками якості, розрахованими на тренувальній та тестовій вибірках.

5. **provenance.json:** метадані походження (*data provenance*), що містять інформацію про версії бібліотек та початкові значення ініціалізації (random_state).

6. **integrity.json:** Файл з контрольними сумами (SHA256) усіх файлів у пакеті, що дозволяє перевірити їхню цілісність та незмінність, гарантуючи, що пакет не було модифіковано після створення.

7. **dag.json (опціонально):** декларативний опис модельної структури у вигляді спрямованого ациклічного графа (DAG) для виконання у цільовому середовищі.

4.2.4. Результати демонстрації та валідації підходу

Для демонстрації гнучкості та практичної цінності розробленої конфігураційної схеми було проведено серію тематичних досліджень (case studies) на наборі даних про вартість поїздок на таксі. Метою експериментів була демонстрація можливостей конфігурування та аналізу впливу архітектурних рішень на кінцевий результат.

У рамках дослідження було протестовано дев'ять різних конфігурацій, кожна з яких була описана за допомогою окремого конфігураційного файлу. Ці експерименти дозволили оцінити вплив таких аспектів, як глибина рециркуляції, кількість моделей на кожному шарі, стратегії агрегації, використання підвищення однорідності даних та застосування ГПО. Результати для кожної конфігурації представлено в табл. 1, в якому представлені значення кореня з середньоквадратичної похибки (RMSE) для різних конфігурацій модельної структури

Таблиця 4.1 — Порівняння ефективності різних архітектур, налаштованих за допомогою запропонованої специфікації

#	Опис архітектури (через конфігурацію)	Ключова конфігурація	RMSE
1	Рециркуляція (1 шар)	recirculation: {layers: 1}	11.06
2	Рециркуляція (1 шар) з ГПО	layers: [{hpo_n_trials: 40}]	12.40
	Рециркуляція (2 шари)	recirculation: {layers: 2}	10.62

3			
4	Рециркуляція (7 шарів, проста)	recirculation: {layers: 7, max_models: 1}	8.97
5	Рециркуляція (7 шарів, гнучка ширина)	layers: [{max_models: 2}, {max_models: 3}, ...]	9.99
6	Рециркуляція (7 шарів, зі змішаним gating)	layers: [{gating: average}, {gating: weighted}, ...]	10.88
7	**Підвищення однорідності + Рециркуляція (1 шар)**	homogenization: true + recirculation: {layers: 1}	**7.09**
8	Підвищення однорідності + Рециркуляція (5 шарів, без ID кластера)	homogenization: {include_cluster_id: false}	7.45
9	Підвищення однорідності + Рециркуляція (5 шарів, з ID кластера)	homogenization: {include_cluster_id: true}	7.16

Отримані дані дозволяють зробити наступні висновки щодо управління архітектурними рішеннями:

1. **Дослідження глибини рециркуляції (порівняння #1, #3, #4):** лінійне збільшення кількості шарів не завжди мінімізує похибку; оптимальною виявилася конфігурація з 7 шарами при наявності однієї моделі на рівень.

2. **Дослідження гнучкості конфігурації шарів (порівняння #4, #5, #6):** специфікація підтвердила здатність тестувати складні гетерогенні ансамблі з різною кількістю моделей та стратегіями агрегації (*gating*)..

3. **Оцінка впливу ГПО (порівняння #1 та #2):** автоматична оптимізація без належного обмеження простору пошуку не гарантує безумовного покращення, що підкреслює потребу у верифікації гіпотез.

4. **Оцінка комбінованих архітектур (порівняння #7, #8, #9):** інтеграція процесів підвищення однорідності даних та рециркуляції забезпечила найнижчий рівень похибки (RMSE = 7.09).

Для валідації критерію відтворюваності експеримент для складної архітектури виконано повторно з ідентичним random_state. Результати запусків збіглися з точністю до 1×10^{-9} що підтверджує машинний детермінізм запропонованого підходу.

4.2.5. Обговорення отриманих результатів.

Результати демонстраційних запусків свідчать, що розроблений декларативний опис є ефективним інструментом проектування складних архітектур. Замість написання імперативного коду, розробник оперує параметрами у конфігураційному файлі, що оптимізує процес дослідження впливу глибини рециркуляції та методів сегментації простору ознак

Даний підхід займає проміжну нішу між універсальними платформами MLOps та системами AutoML На відміну від інструментів, як-от MLflow [74], які вимагають написання коду для реалізації кастомної логіки, запропонована схема формалізує ансамблі декларативно. Водночас, на відміну від систем AutoML[47], які переважно функціонують за принципом «чорної скриньки», розроблений підхід зберігає контроль над архітектурою з боку розробника, фокусуючись на відтворюваності, а не на сліпому переборі.

Генерація самодостатнього пакета результатів із метаданими походження (*provenance*) вирішує фундаментальну проблему багатокомпонентних систем машинного навчання — залежність результату від неявного локального контексту

Практична значимість отриманих результатів полягає у створенні програмного забезпечення для роботи зі складними модельними архітектурами. Запропонований підхід:

1. Мінімізація часових витрат на експериментування та налаштування складних ансамблів.

2. **Забезпечення машинного детермінізму** результатів, що є вимогою для використання моделей в автономних моніторингових системах

3. Інтеграція етапів дослідження та розгортання: згенерований пакет слугує структурною основою для створення портативних графів обчислень (DAG), готових до виконання у цільових середовищах, що детально розглядається у наступному підрозділі..

4.3. DAG-орієнтоване подання процесів конструювання алгоритмів синтезу моделей

4.3.1. Обґрунтування підходу та вимоги до виконання

Як було показано у попередніх підрозділах, багатошарові ансамблі та методи, що виконують попередню кластеризацію дозволяють підвищити точність моніторингового агента. Однак практичне розгортання таких структур потребує вирішення проблеми відтворюваності обчислень. Традиційні формати серіалізації (наприклад, ONNX) та існуючі MLOps-рішення не здатні повною мірою забезпечити прозорість та детермінізм виконання складних ансамблевих архітектур (детальний аналіз наведено у підрозділі 1.3).

Для вирішення цієї проблеми та виконання вимог до автономних моніторингових агентів, у даній роботі набув подальшого розвитку метод багатошарового синтезу агентних моделей за рахунок використання у процесі проектування алгоритмів синтезу моделей спрямованого ациклічного графу (*Directed Acyclic Graph, DAG*). Це дозволяє забезпечити багатоетапність процесу удосконалення структури моделі шляхом обробки сигналів на виході та використання моделей, побудованих у різних середовищах.

Завданням даного етапу є формалізація DAG-орієнтованого підходу, що дозволяє представити складний метод конструювання АСМ у вигляді єдиного, самодостатнього та портативного графу, гарантуючи детермінованість його виконання.

Для реалізації цього підходу розроблено структуру DAG та визначено логіку роботи його компонентів. Показано, як процеси багатошарової рециркуляції та підвищення однорідності даних декомпозуються на послідовність логічних етапів у графі. Паралельно створено програмний рушій для інтерпретації графу та інструменти верифікації проміжних результатів, що підвищує аудитоздатність усього процесу.

4.3.2. Компонентна структура обчислювального графа

Запропонований метод полягає у поданні процесу конструювання алгоритмів синтезу моделей (АСМ) як спрямованого ациклічного графу (DAG). Таке математичне подання об'єднує окремі моделі, етапи кластеризації та агрегації у єдину детерміновану систему. Це забезпечує багатоетапність обробки даних: граф послідовно передає сигнали від локальних моделей до шарів рециркуляції, гарантуючи коректне виконання усього конвеєра. У цьому підрозділі описано загальну структуру графу, ролі та логіку роботи його основних компонентів (вузлів), а також формат обміну даними між ними.

4.3.2.1. Загальна структура графу

Граф виконання (*DAG*) подається у декларативному описі, що складається з двох ключових елементів: nodes (список усіх вузлів-операцій) та entry (ідентифікатор фінального вузла, результат якого є кінцевим результатом роботи графа).

Кожен вузол має унікальний ідентифікатор (id), тип (type), що визначає його функцію, та список вхідних залежностей (inputs), який вказує на вузли-джерела даних. Така структура дозволяє рушію виконання автоматично визначати порядок обчислень на основі алгоритму топологічного сортування. Вся необхідна для роботи вузла конфігурація (шляхи до збережених моделей, параметри агрегації) зберігається безпосередньо в його описі, що робить граф повністю незалежним від зовнішнього середовища.

4.3.2.2. Опис основних вузлів графу: декомпозиційний підхід

Для забезпечення прозорості процесу реалізовано декомпозиційний підхід: складні трансформації (наприклад, кластеризація зі збагаченням) розбиваються на послідовність атомарних кроків. Архітектура графу включає наступні типи вузлів::

4. **Вузол input:** Точка входу графу. Приймає вихідний набір даних (наприклад, Pandas DataFrame) та передає його наступним вузлам без трансформацій.
5. **Вузол single_model:** Базовий компонент, який застосовує серіалізовану модель (разом з її об'єктом передобробки) до вхідного набору даних для генерації прогнозів. Конфігурація вузла містить фіксовані посилання на файли моделі (model_path) та конвеєра передобробки (input_pipeline_path), що гарантує використання заздалегідь підготовлених артефактів.
6. **Вузли підвищення однорідності даних:** Процес сегментації реалізується трьома взаємодіючими вузлами::
    1. **segment_predictions:** Для кожного з k кластерів у графі створюється окремий вузол segment_predictions. Його завдання – застосувати цільову модель, навчену для відповідного сегмента, до вхідних даних і згенерувати нову ознаку (вектор прогнозів цієї моделі). Відповідно, для k = 3 кластерів граф міститиме три паралельні вузли (seg_pred_0, seg_pred_1, seg_pred_2).
    2. **cluster_assignment:** використовує серіалізований класифікатор для генерації прогнозованої мітки кластера для нових екземплярів.
    3. **cluster_homogenization_assemble:** вузол-агрегатор, який об'єднує вихідні ознаки, прогнози від segment_predictions та мітки від cluster_assignment у збагачений словник ознак згідно з маніфестом. Даний процес проілюстровано на рис. 4.6.

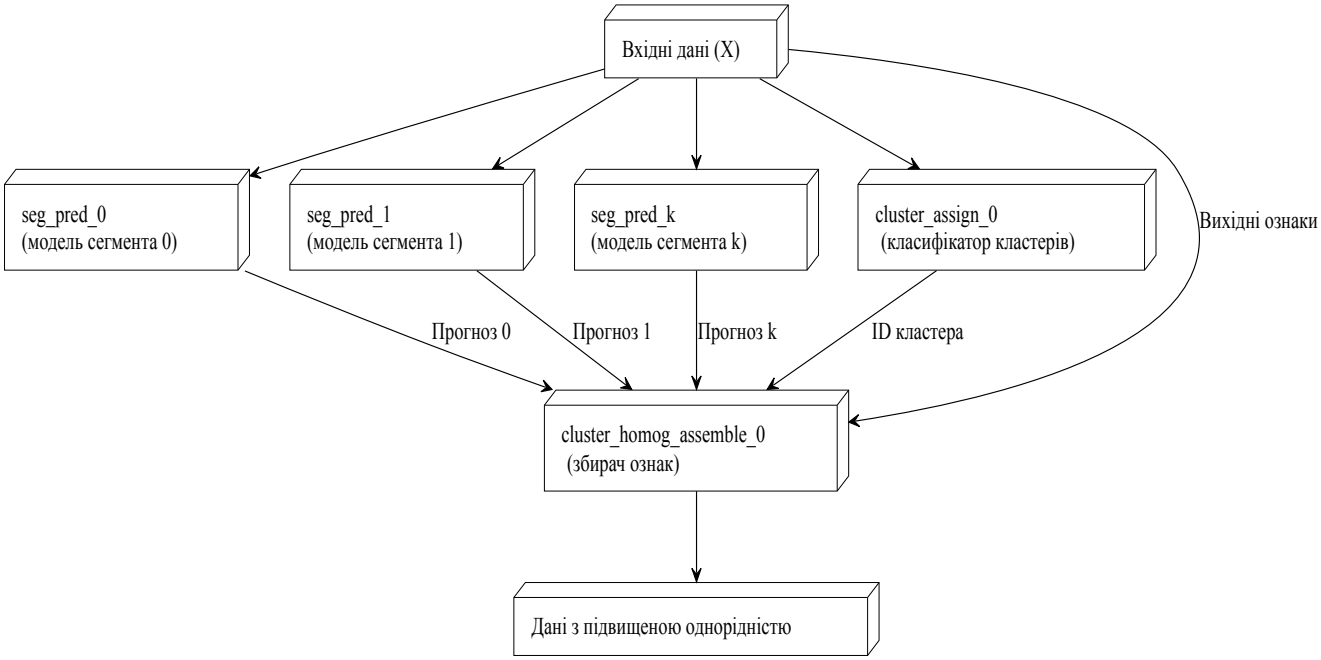


Рисунок 4.6 — Діаграма діяльності процесу підвищення однорідності даних

7. **Вузол recirculation_layer: Вузол recirculation_layer:** Центральний елемент багатошарових ансамблів. Агрегує прогнози від моделей поточного шару та формує розширений набір ознак для наступного. Параметр gating визначає стратегію передачі даних: просте усереднення (average), зважене усереднення (weighted) або передача індивідуальних прогнозів як нових ознак. Потік даних між шарами рециркуляції відображено на рис. 4.7.

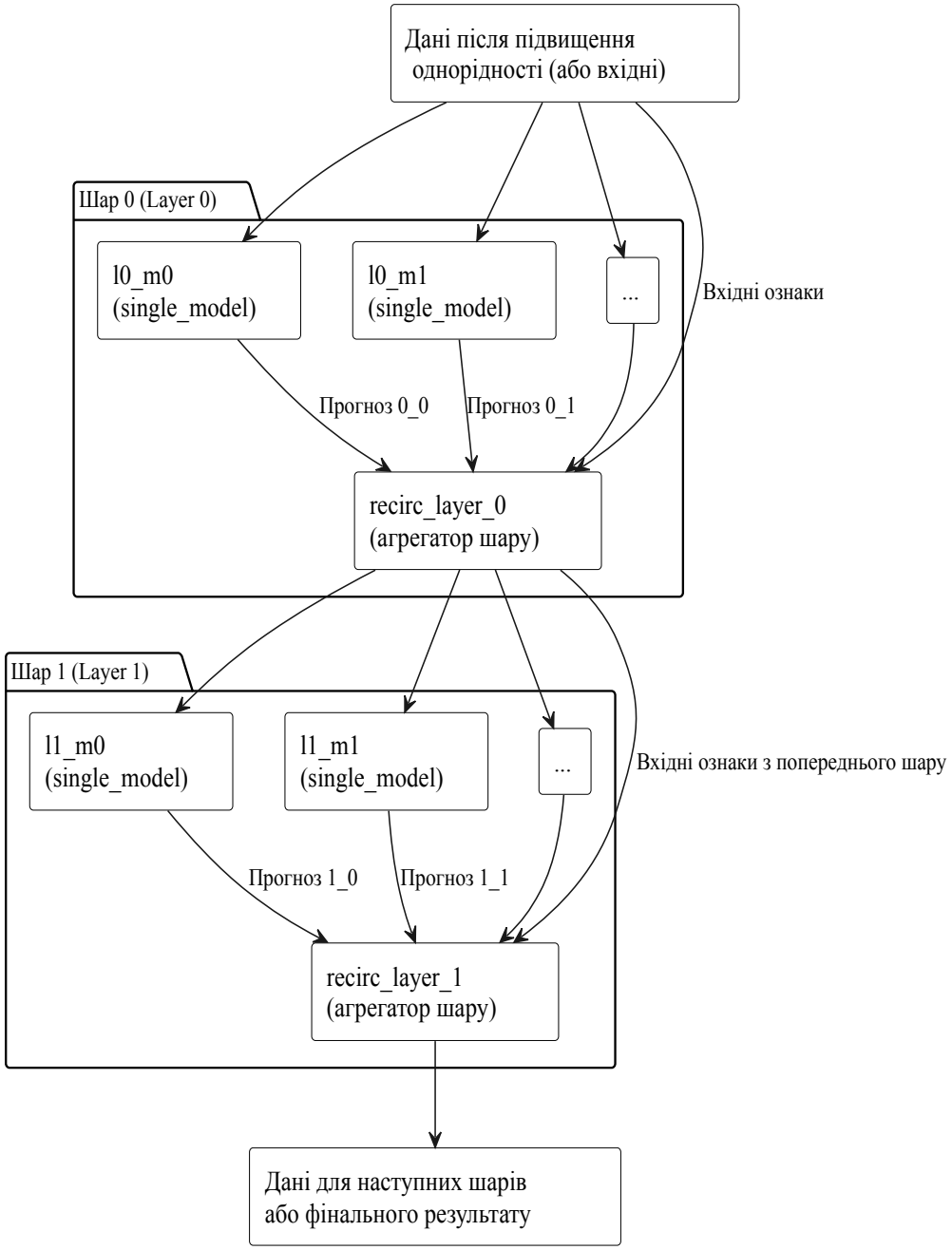


Рисунок 4.7 — Діаграма діяльності процесу рециркуляції

Комбінація перелічених вузлів дозволяє декларативно описувати багатоетапні методи АСМ. Загальну топологію графу, що включає фазу підвищення однорідності та шари рециркуляції, наведено на рис. 4.8.

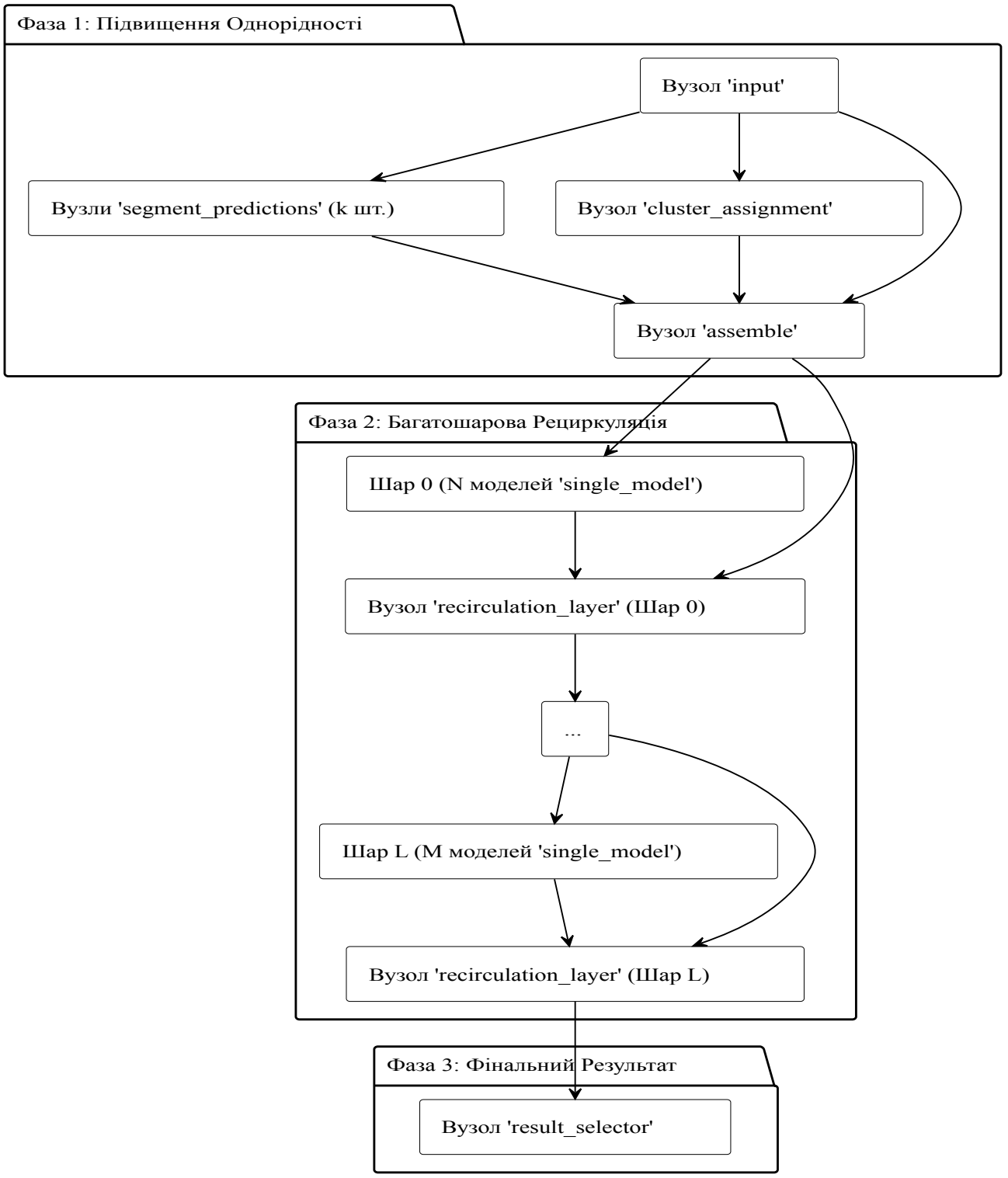


Рисунок 4.8 — Загальна діаграма діяльності обробки DAG

4.3.2.3. Формат обміну даними

Для стандартизації взаємодії між вузлами розроблено єдиний формат — PredictionFrame. Ця структура даних інкапсулює результати роботи вузла: вектор

прогнозів (result), згенеровані ознаки (features) та метадані (metadata щодо типу вузла і шляху до моделі). Використання єдиного контракту обміну даними усуває необхідність конвертації форматів між шарами та спрощує логіку рушія виконання.

4.3.3. Механізм виконання обчислювального графа

Даний підрозділ описує алгоритм інтерпретації DAG програмним рушієм та механізми забезпечення детермінізму результатів.

4.3.3.1. Алгоритм виконання графу

Процес застосування моделі (*inference*) розпочинається з ініціалізації виконавця графа (GraphExecutor). Алгоритм перевіряє цілісність топології DAG (відсутність циклів, коректність посилань) та виконує топологічне сортування. Це формує лінійну чергу, де жоден вузол не запускається до завершення роботи його попередників (рис. 4.9).

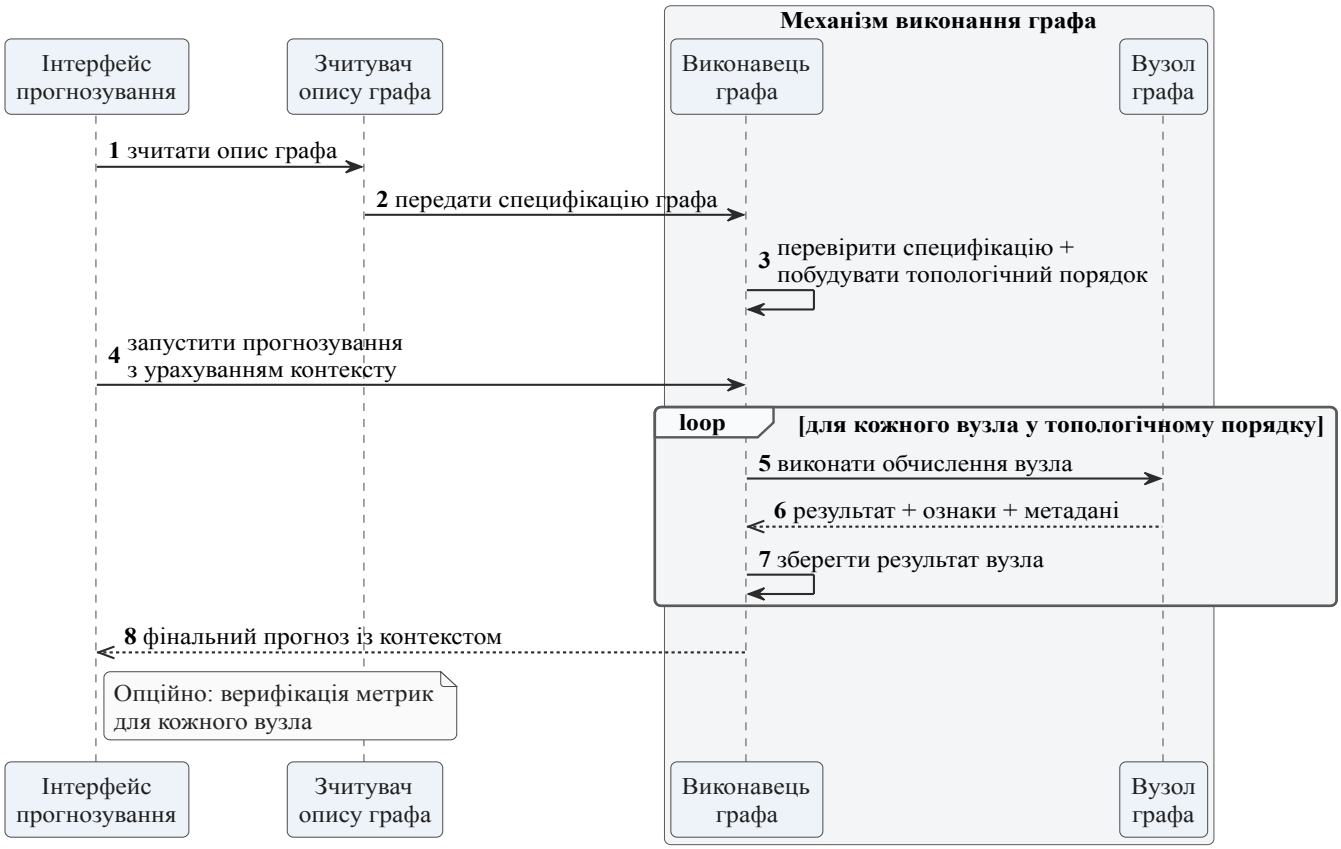


Рис. 4.9 — Діаграма послідовностей виконання обчислювального графа

Програмна реалізація багатошарових ансамблів потребує контролю за розподілом пам'яті та станом системи. Оскільки структура може містити взаємозалежних вузлів, система здійснює синхронну передачу даних між ними. Попередній обхід графа та формування черги обчислень запобігають виникненню взаємних блокувань (*deadlocks*). Такий контроль забезпечує машинну тотожність результатів при повторних запусках агента.

Обмін даними здійснюється за допомогою стандартизованої структури контейнера PredictionFrame. Під час обробки графа результати локальних моделей не передаються як неструктуровані числові масиви. Кожен вузол інкапсулює свої прогнози та згенеровані нові ознаки (наприклад, ідентифікатори кластерів) у цей уніфікований об'єкт.

Вузол агрегації (наприклад, recirculation_layer) приймає масив об'єктів PredictionFrame від попередніх шарів, розпаковує їх згідно з конфігурацією та формує розширену матрицю ознак. Цей архітектурний підхід знижує зв'язність (*coupling*) компонентів: вузли взаємодіють через спільний інтерфейс, що мінімізує потребу у конвертації форматів даних між кроками.

4.3.3.2. Механізми забезпечення детермінізму та валідації

Запропонована архітектура реалізує механізми детермінізму для отримання ідентичних прогнозів на однакових вхідних даних. Це досягається шляхом комбінації топологічного порядку виконання та використання фіксованих версій серіалізованих артефактів. Початкові значення генераторів псевдовипадкових чисел (*seeds*) фіксуються ще на етапі навчання, що унеможливлює випадкову поведінку моделей під час експлуатації [69].

Можливість аналізу обчислень реалізовано через такі функції:

1. Верифікація (verify): обчислення метрик якості (RMSE, MAE) для виходу будь-якого вузла графа. Це дозволяє порівнювати точність прогнозів на проміжних шарах рециркуляції.

2. Трасування (*trace*): збереження проміжних результатів кожного вузла в окремі файли для подальшого аудиту.

3. Експорт субграфів: виконання ізольованих частин графа для оцінки впливу конкретного компонента на загальний результат.

4.3.4. Валідація відтворюваності та прозорості обчислень

Для демонстрації розглянемо приклад графу, що поєднує підвищення однорідності (3 кластери) та багатошарову рециркуляцію глибиною у 5 шарів. Граф цієї структури візуалізовано на рис. 4.10.

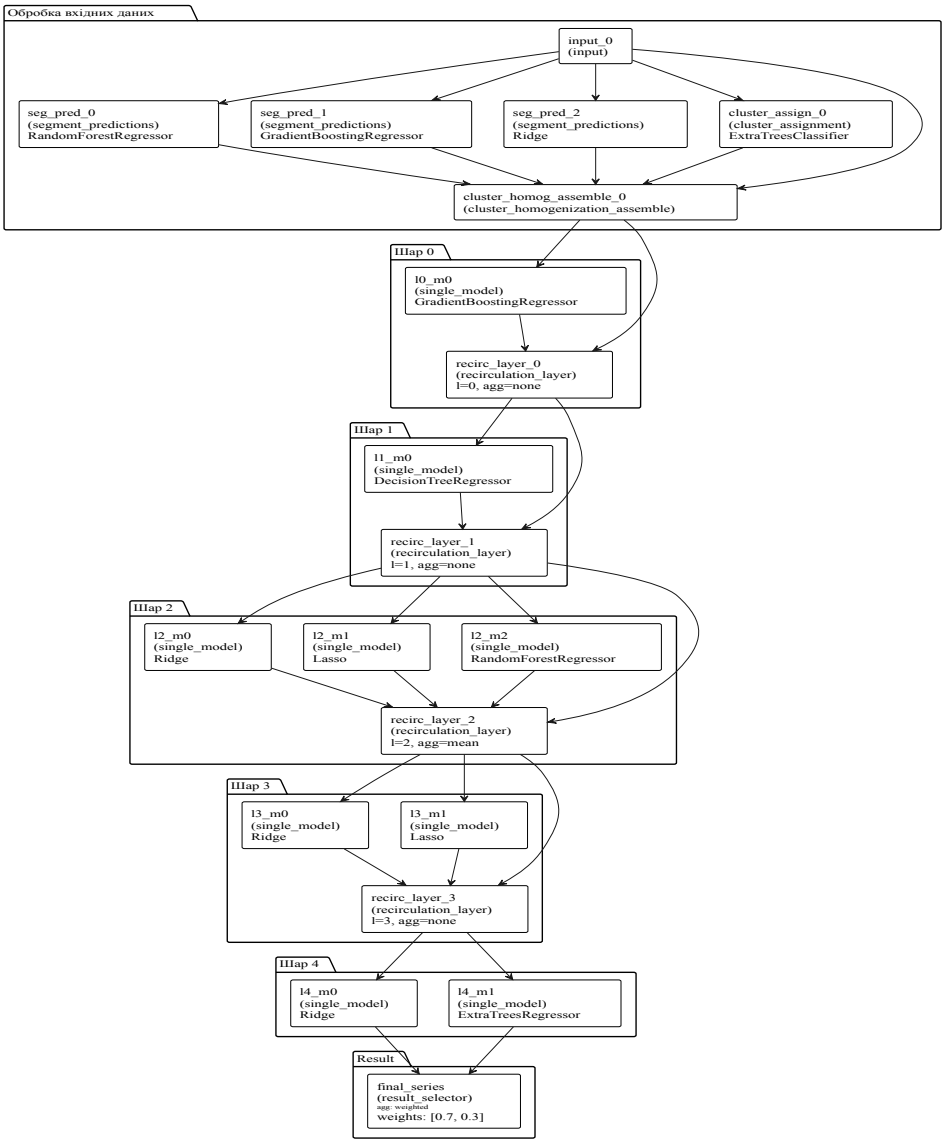


Рисунок 4.10 — Структура DAG, що був використаний для демонстрації архітектури

Обчислювальний процес декомпозовано на логічні блоки:

1. **Підвищення однорідності:** блок включає паралельне виконання трьох вузлів segment_predictions (по одному для кожної локальної моделі сегмента) та вузла cluster_assignment.

2. **Агрегація ознак:** результати локальних моделей та вихідні дані об'єднуються у вузлі cluster_homogenization_assemble. Збагачений набір ознак стає входом для першого шару рециркуляції ("Шар 0").

3. **Рециркуляція:** вихід кожного проміжного шару (recirc_layer_k) стає входом для моделей наступного рівня.

4. **Фіналізація:** результат формується вузлом result_selector, який застосовує зважене усереднення до прогнозів моделей останнього шару.

Ця топологія описується у файлі dag.json. Вузол cluster_homogenization_assemble містить поле feature_order, яке визначає порядок ознак у збагаченому наборі. Вузол recirc_layer_2 використовує параметр gating_mode: "average" (просте усереднення), тоді як фінальний вузол result_selector містить aggregation: "weighted" із набором вагових коефіцієнтів (weights).

4.3.4.1. Результати верифікації вузлів графу

Використання Функція верифікації застосовується для аналізу ефективності окремих компонентів графа. У табл. 4.2 наведено результати для описаної вище архітектури. Для кожного проміжного вузла розраховано метрики якості та визначено його ранг (де 1 — найвища точність).

Таблиця 4.2 — Результати верифікації вузлів DAG на тестовій вибірці

Ідентифікатор вузла	Рециркуляція (шар)	Алгоритм	RMSE	MAE	R2	Сумарний ранг
l2_m0	2	Ridge	7.10	4.67	0.978	10
l1_m0	1	Decision Tree	7.09	4.71	0.978	16
l2_m1	2	Lasso	7.42	4.67	0.976	30

Продовження таблиці 4.2

recirc_layer_2	2	Рециркуляція (середнє)	7.32	4.78	0.977	31
l4_m1	4	ExtraTree	7.27	4.78	0.977	35
recirc_layer_0	0	Рециркуляція	9.57	4.93	0.961	43

Аналіз даних свідчить, що точність прогнозу не завжди монотонно зростає з глибиною рециркуляції. Проміжна модель l2_m0 (алгоритм Ridge на 2-му шарі) має один із найкращих результатів та низький сумарний ранг. Відповідно, верифікація дозволяє знаходити оптимальні проміжні конфігурації всередині ансамблю.

Використання цих даних дає змогу спростити архітектуру графа шляхом вилучення надлишкових шарів без суттєвої втрати точності. Порівняння отриманих метрик із результатами етапу навчання (Розділ 3) підтвердило машинну ідентичність обчислень та їх відтворюваність.

4.3.5. Обговорення результатів та подальші напрями досліджень

Аналіз розробленого методу дозволяє виділити його ключові властивості: портативність, прозорість та гнучкість управління архітектурою.

Метод забезпечує портативність та відтворюваність. Представлення всього процесу виконання у вигляді єдиного *dag.json* файлу разом з необхідними моделями створює самодостатній "пакет". Такий пакет можна легко переносити між різними середовищами, гарантуючи отримання однакових результатів. Це вирішує задачу розриву між дослідженням та розгортанням, де результати часто залежать від налаштувань локального середовища, версій бібліотек чи навіть операційної системи.

Прозорість та аудит. Декомпозиція операцій на послідовність типізованих вузлів дозволяє відстежувати потік даних (*data flow*). Файл dag.json виступає єдиним джерелом істини (*single source of truth*) для процесу виконання, а функції трасування та верифікації використовуються для аудиту багатошарових ансамблів.

Практична значущість отриманих результатів полягає у створенні інструментарію для роботи зі складними модельними архітектурами. Запропонований

метод спрощує їхній аудит [82] та створює забезпечує детерміновану поведінку моделей моніторингового агента у розподілених системах.

Незважаючи на продемонстровані переваги, запропонований підхід має певні обмеження та напрямки для подальших досліджень. Поточна реалізація орієнтована на офлайн-виконання і не оптимізована для обробки даних у режимі реального часу (стрімінгу), що вимагало б розробки механізмів управління станом всередині графу. Перспективним напрямком є також розробка інтерактивних інструментів візуалізації, які б дозволяли в реальному часі аналізувати проміжні результати та потоки даних всередині графу. Майбутні дослідження також можуть бути спрямовані на розширення набору доступних вузлів для підтримки нових типів моделей та алгоритмів агрегації.

4.4. Висновки до розділу 4

1. **Спроектовано об'єктно-орієнтовану архітектуру** моніторингового агента для практичної реалізації методів адаптивного синтезу. Розроблено декларативний опис конфігурації (у форматі JSON-схеми), який формалізує та об'єднує в єдиний процес усі етапи конструювання: від передобробки даних до інтеграції багатоетапних архітектур (багатошарова рециркуляція, підвищення однорідності). Застосування цього підходу мінімізує часові витрати на проектування та автоматизує синтез моделей без використання імперативного програмування.

2. **Впроваджено наскрізний механізм управління** життєвим циклом моделі, який формалізує перехід від декларативного опису через етап автоматизованого навчання до генерації спрямованого ациклічного графа (*DAG*). Представлення експлуатаційного процесу у вигляді графа з декомпозицією складних операцій (кластеризація, рециркуляція, агрегація) на типізовані вузли забезпечує прозорість та верифікованість обчислень. Це дозволяє проводити ізольований аудит кожної проміжної операції всередині багатошарового ансамблю.

3. **Створено метод генерації відтворюваних пакетів результатів** (*bundle*). Пакет інкапсулює компоненти для розгортання та аудиту: серіалізовані моделі,

конфігурації, метрики якості та метадані походження (*provenance*). Експериментально підтверджено, що фіксований топологічний порядок обчислень у графі спільно з артефактами пакета гарантують машинний детермінізм результатів. Це забезпечує портативність архітектури та перенесення конструйованих АСМ між різними середовищами без деградації точності.

4. **Доведено, що розроблене програмне забезпечення вирішує інженерну проблему** розриву між етапами наукового дослідження та практичного розгортання моделей. Забезпечення відтворюваності, цілісності та детермінованості обчислень формує основу для практичного застосування багатошарових ансамблів у складі автономних моніторингових агентів у розподілених мультиагентних системах.

ВИСНОВКИ

У дисертаційній роботі розв'язано науково-прикладне завдання підвищення ефективності функціонування моніторингового програмного агента в інформаційній технології інтелектуального моніторингу шляхом розробки методів побудови та засобів програмної реалізації багатошарових агентних моделей.

Виконані теоретичні та експериментальні дослідження дозволили зробити такі висновки:

1. Проведено аналіз існуючих підходів до побудови моделей у системах моніторингу. Встановлено, що класичні статичні ансамблі та системи автоматизованого машинного навчання (AutoML) мають обмеження при роботі в умовах динамічних змін статистичних властивостей даних. Це обумовлює доцільність створення та використання спеціалізованих методів адаптивного синтезу багатошарових моделей для автономних агентів.

2. Розроблено метод підвищення точності агентних моделей за рахунок зростання однорідності точок спостереження. Експериментально підтверджено, що попередня кластеризація простору ознак із подальшою інтеграцією прогнозів локальних моделей та міток класифікатора кластерів як нових мета-ознак допомагає моделям краще розпізнавати режими функціонування об'єкта. Застосування цього підходу дозволило знизити середньоквадратичну похибку прогнозування (RMSE) на величину до 27,9 % на відкритих тестових наборах даних.

3. Удосконалено метод синтезу багатошарових моделей (рециркуляції) шляхом залучення різнотипних алгоритмів машинного навчання на різних структурних шарах. Це дає можливість поєднувати властивості різних класів алгоритмів під час перетворення простору ознак. Зокрема, показано, що алгоритми лінійної регуляризації можуть використовуватися для зменшення впливу мультиколінеарності на глибоких шарах ансамблю. Експериментальне застосування удосконаленого методу забезпечило додаткове зниження похибки на 1,9...15,9 % порівняно з простими ансамблями.

4. Набув подальшого розвитку метод багатошарового синтезу агентних моделей за рахунок використання спрямованого ациклічного графу (DAG) у процесі проектування алгоритмів синтезу. Доведено, що подання процесу у вигляді графа забезпечує необхідну багатоетапність конструювання моделей. Практично це досягається завдяки тому, що кожен крок обробки стає окремим вузлом: це дає можливість незалежно перевіряти сигнали на виході проміжних етапів (наприклад, оцінювати похибку окремого шару рециркуляції) та об'єднувати в єдиний конвеєр моделі, побудовані в різних програмних середовищах. Даний метод усуває проблему втрати логіки взаємодії алгоритмів при їх збереженні та гарантує машинний детермінізм обчислень.

5. Побудовано інформаційну технологію проектування програмного забезпечення моніторингових програмних агентів. Застосування декларативного підходу дозволило формалізувати життєвий цикл моделі у вигляді єдиного конфігураційного опису та керувати процесом автоматизованого синтезу без написання імперативного коду. Використання розробленого програмного забезпечення дозволяє генерувати самодостатні відтворювані пакети результатів, що вирішує інженерну проблему перенесення навчених багатошарових ансамблів у середовище експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 10. Model persistence. *scikit-learn*. URL: https://scikit-learn/stable/model_persistence.html (дата звернення: 19.10.2025).

2. Breiman L. Random Forests. *Machine Learning*. Вип. 45, № 1. С. 5–32. DOI:10.1023/A:1010933404324.

3. Chavarriaga E., Jurado F., Rodríguez F. D. An approach to build JSON-based Domain Specific Languages solutions for web applications. *Journal of Computer Languages*. Вип. 75, 06.2023. С. 101203. DOI:10.1016/j.cola.2023.101203.

4. Cheng Z., Wu Y., Li Y. та ін. A Comprehensive Review of Explainable Artificial Intelligence (XAI) in Computer Vision. *Sensors*. Вип. 25, № 13. С. 4166. DOI:10.3390/s25134166.

5. Dempster A. P., Laird N. M., Rubin D. B. Maximum Likelihood from Incomplete Data Via the *EM* Algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*. Вип. 39, № 1. С. 1–22. DOI:10.1111/j.2517-6161.1977.tb01600.x.

6. Geurts P., Ernst D., Wehenkel L. Extremely randomized trees. *Machine Learning*. Вип. 63, № 1. С. 3–42. DOI:10.1007/s10994-006-6226-1.

7. Giner-Miguelez J., Gómez A., Cabot J. A domain-specific language for describing machine learning datasets. *Journal of Computer Languages*. Вип. 76, 08.2023. С. 101209. DOI:10.1016/j.cola.2023.101209.

8. Greco S., Vacchetti B., Apiletti D. та ін. DriftLens: A Concept Drift Detection Tool. OpenProceedings.org, 2024. DOI:10.48786/EDBT.2024.75. 2024.

9. Grinsztajn L., Oyallon E., Varoquaux G. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*. Вип. 35, 06.12.2022. С. 507–520.

10. Guazzelli A., Zeller M., Lin W. C. та ін. PMML: An open standard for sharing models. *R Journal*. Вип. 1, № 1. С. 60–65. DOI:10.32614/rj-2009-010.

11. Hutson M. Artificial intelligence faces reproducibility crisis. *Science*. Вип. 359, № 6377. С. 725–726. DOI:10.1126/science.359.6377.725.

12. Jennings N. R. On agent-based software engineering. *Artificial Intelligence*. Вип. 117, № 2. С. 277–296. DOI:10.1016/S0004-3702(99)00107-1.

13. Katzir L., Elidan G., El-Yaniv R. Net-DNF: Effective Deep Modeling of Tabular Data. *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=73WTGs96kho (дата звернення: 10.11.2024).

14. MacQueen J. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, 1967. С. 281–298. Також доступний за посиланням: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fifth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992 (дата звернення: 06.04.2026).

15. Murtagh F., Contreras P. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*. Вип. 2, № 1. С. 86–97. DOI:10.1002/widm.53.

16. Ostapiuk V., Holub S. Model Synthesis Algorithms for a Monitoring Software Agent. *Mathematical Modeling and Simulation of Systems*. ред. Volodymyr Kazymyr, Anatoliy Morozov, Alexander Palagin, Serhiy Shkarlet, Nikolai Stoianov, Dmitri Vinnikov, Mark Zheleznyak. Cham : Springer Nature Switzerland, 2024. С. 113–129. DOI:10.1007/978-3-031-67348-1_9.

17. Park A., Jung S. Y., Yune I. та ін. Applying Robotic Process Automation to Monitor Business Processes in Hospital Information Systems: Mixed Method Approach. *JMIR Medical Informatics*. Вип. 13, 07.03.2025. С. e59801. DOI:10.2196/59801.

18. Patrício L., Varela L., Silveira Z. Proposal for a Sustainable Model for Integrating Robotic Process Automation and Machine Learning in Failure Prediction and

Operational Efficiency in Predictive Maintenance. *Applied Sciences*. Вип. 15, № 2. С. 854. DOI:10.3390/app15020854.

19. Quaranta L., Azevedo K., Calefato F. та ін. A multivocal literature review on the benefits and limitations of industry-leading AutoML tools. *Information and Software Technology*. Вип. 178, 02.2025. С. 107608. DOI:10.1016/j.infsof.2024.107608.

20. Quinlan J. R. Induction of decision trees. *Machine Learning*. Вип. 1, № 1. С. 81–106. DOI:10.1007/BF00116251.

21. Reynolds D. Gaussian Mixture Models. *Encyclopedia of Biometrics*. Springer, Boston, MA, 2009. С. 659–663. DOI:10.1007/978-0-387-73003-5_196.

22. Sun Y., Song Q., Gui X. та ін. AutoML in The Wild: Obstacles, Workarounds, and Expectations. *Proceedings of the 2023 CHI Conference on Human Factors in Computing SystemsCHI '23: CHI Conference on Human Factors in Computing Systems*. Hamburg Germany : ACM, 2023. DOI:10.1145/3544548.3581082. С. 1–15.

23. Vernotte A., Cretin A., Legeard B. та ін. A domain-specific language to design false data injection tests for air traffic control systems. *International Journal on Software Tools for Technology Transfer*. Вип. 24, № 2. С. 127–158. DOI:10.1007/s10009-021-00604-4.

24. Zhou Z.-H., Feng J. Deep forest. *National Science Review*. Вип. 6, № 1. С. 74–86. DOI:10.1093/nsr/nwy108.

25. Zou H., Hastie T. Regularization and Variable Selection Via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*. Вип. 67, № 2. С. 301–320. DOI:10.1111/j.1467-9868.2005.00503.x.

26. Голуб С.В., Колос П.О. Застосування стратегії оптимальності при виборі алгоритмів синтезу моделей у системах багаторівневого соціоекологічного моніторингу. Vol. 4, 2010. P. 127–134.

27. Голуб С.В, Остапюк В. Підвищення однорідності вхідних даних у методах машинного навчання ансамблів моделей. *Актуальні проблеми*

автоматизації та інформаційних технологій. Vol. 29, 2025. P. 131–155. DOI:10.15421/432513.

28. Голуб С. Багаторівневе моделювання в технологіях моніторингу оточуючого середовища. *Черкаси: Вид. від. ЧНУ імені Богдана Хмельницького*. 2007. P. 220.

29. Колос П.О, Голуб С.В Умови конструювання алгоритмів синтезу моделей у системах багато- рівневого перетворення інформації. Вісник Східноукраїнського національного університету імені Володимира Даля, 2009. P. 325–329.

30. Остапюк В., Голуб С.В Багатошарові архітектури машинного навчання як інструмент аналізу складних даних сенсорів у робототехнічних системах. *Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2025). Вісімнадцята міжнародна науково-практична конференція*. Київ, 2025P. 362–364.

31. Остапюк В., Голуб С. Дослідницький застосунок для побудови моніторингових агентів. *Інформаційні технології та комп'ютерне моделювання*. Івано-Франківськ, 2023P. 101–103.

32. Остапюк В., Голуб С. Інструменти для моделювання прогнозних алгоритмів. *Modern problems of science, education and society. Proceedings of the 3rd International scientific and practical conference. SPC "Sci-conf.com.ua"*. Kyiv, 2023P. 21–27.

33. Остапюк В., Голуб С. Концептуальні підходи до адаптивного синтезу моделей для задач інтелектуального моніторингу. *Інформаційні технології та комп'ютерне моделювання. Матеріали Міжнародної науково-практичної конференції Інформаційні технології та комп'ютерне моделювання*. Івано-Франківськ, 2025 P. 142–143.

34. Остапюк В., Голуб С. Методи штучного інтелекту для вирішення задачі виявлення ключових слів публікацій у соціальних мережах. *Інформація, комунікація, суспільство 2024*. P. 138–140.

35. Остапюк В., Голуб С. Методи штучного інтелекту для вирішення задачі виявлення рівня аварійності обладнання у робототехнічних системах. *ІНТЕГРОВАНІ ІНТЕЛЕКТУАЛЬНІ РОБОТОТЕХНІЧНІ КОМПЛЕКСИ (ІІРТК 2024)*. Київ, 2024Р. 398–400.

36. Остапюк В., Голуб С. Новітні підходи до кластерного призначення даних на основі інтерпретації моделей машинного навчання. *Сучасні інформаційні технології та системи штучного інтелекту. Матеріали 1-ї Міжнародної науково-практичної конференції. Частина 1 Сучасні інформаційні технології та системи штучного інтелекту*. Харків-Яремче, 2025Р. 134–137.

37. Остапюк В., Голуб С. Роль методів інтерпретації машинного навчання, зокрема SHAP, в аналізі та сегментації користувачів соціальних мереж. *XIV МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «ІНФОРМАЦІЯ, КОМУНІКАЦІЯ, СУСПІЛЬСТВО 2025» ПРИСВЯЧЕНА ПАМ'ЯТІ ПРОФЕСОРА АНДРІЯ ПЕЛЕЩИШИНА*. Львів, 2025ISBN 978-966-994-052-0. Р. 83–84.

38. Остапюк В. Штучний інтелект, як ресурс посилення цивілізаційних спроможностей. Розвиток наукової думки: актуальні питання, досягнення та інновації. *Молодий вчений*. Хмельницький-Одеса, 2023Р. 86–89.

39. Abdel-Basset M., Abdel-Fatah L., Sangaiah A. K. Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review. *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. ed. A. K. Sangaiah., M. Sheng., Z. Zhang. Academic Press, 2018. P. 185–231. DOI:https://doi.org/10.1016/B978-0-12-813314-9.00010-4.

40. Borisov V., Leemann T., Seßler K. et al. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*. Vol. 35, Issue 6. P. 7499–7519. DOI:10.1109/TNNLS.2022.3229161.

41. Cecaj A., Lippi M., Mamei M. et al. Comparing Deep Learning and Statistical Methods in Forecasting Crowd Distribution from Aggregated Mobile Phone Data. *Applied Sciences*. Vol. Volume 10, 27.09.2020. DOI:10.3390/app10186580.

42. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*DOI:10.1145/2939672.2939785. P. 785–794.

43. Dags — Airflow 3.1.0 Documentation. URL: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html (accessed 19/10/2025).

44. Dynamic Dag Generation — Airflow 3.1.7 Documentation. URL: https://airflow.apache.org/docs/apache-airflow/stable/howto/dynamic-dag-generation.html (accessed 17/02/2026).

45. Ellerm A., Gahegan M., Adams B. LivePublication: The Science Workflow Creates and Updates the Publication. *2023 IEEE 19th International Conference on e-Science (e-Science)2023 IEEE 19th International Conference on e-Science (e-Science)*. Limassol, Cyprus : IEEE, 2023. DOI:10.1109/e-Science58273.2023.10254857. P. 1–10.

46. Farris F. A. The Gini Index and Measures of Inequality. *The American Mathematical Monthly*. Vol. 117, Issue 10. P. 851–864. DOI:10.4169/000298910x523344.

47. Feurer M., Klein A., Eggensperger K. et al. Efficient and Robust Automated Machine Learning. *Neural Information Processing Systems*. URL: https://www.semanticscholar.org/paper/Efficient-and-Robust-Automated-Machine-Learning-Feurer-Klein/775a4e375cc79b53b94e37fa3eedff481823e4a6 (accessed 20/09/2025).

48. Gomes H. M., Gunasekara N., Sun Y. Machine Learning on the Fly: A Hands-On Tutorial for Streaming Data. *2025 IEEE 41st International Conference on Data Engineering (ICDE)2025 IEEE 41st International Conference on Data Engineering (ICDE)*. Hong Kong, Hong Kong : IEEE, 2025. DOI:10.1109/ICDE65448.2025.00342. P. 4513–4516.

49. Hastie T., Tibshirani R., Wainwright M. Statistical Learning with Sparsity: The Lasso and Generalizations. Chapman & Hall/CRC, 2015. 367 p. ISBN 978-1-4987-1216-3.

50. Hinder F., Vaquet V., Hammer B. One or two things we know about concept drift—a survey on monitoring in evolving environments. Part A: detecting concept drift. *Frontiers in Artificial Intelligence*. Vol. Volume 7-2024, 2024. DOI:10.3389/frai.2024.1330257.

51. Hochreiter S., Schmidhuber J. Long Short-term Memory. *Neural computation*. Vol. 9, 01.12.1997. P. 1735–1780. DOI:10.1162/neco.1997.9.8.1735.

52. Hoerl A. E., Kennard R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*. Vol. 42, Issue 1. P. 80–86. DOI:10.2307/1271436.

53. Holub S. V., Ostapiuk V. V. A DECLARATIVE APPROACH TO THE DESIGN AND REPRODUCIBLE LEARNING OF COMPLEX MODEL STRUCTURES FOR MONITORING SOFTWARE AGENTS. *Ukrainian Journal of Information Technology*. Vol. 7, Issue 2. P. 1–8. DOI:10.23939/ujit2025.02.001.

54. Holub S. V., Tolbatov D. V. Удосконалення методу синтезу багатошарових моделей моніторингового програмного агента. *Actual problems of automation and information technology*. Vol. 27, Issue 0. DOI:10.15421/432306.

55. Holub S. V., Ostapiuk V. V. DAG-oriented representation of model synthesis algorithm construction processes by monitoring software agents. *Mathematical machines and systems*. Vol. 1, 2026. P. 73–86. DOI:10.34121/1028-9763-2026-1-73-86.

56. Holub S. V., Ostapiuk V. V. Machine learning of multilayer models of a monitoring software agent. *Mathematical machines and systems*. Vol. 2, 2025. P. 76–96. DOI:10.34121/1028-9763-2025-2-76-95.

57. Jain A. K. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*. ("Award winning papers from the 19th International Conference on Pattern

Recognition (ICPR)" Series)Vol. 31, Issue 8. P. 651–666. DOI:10.1016/j.patrec.2009.09.011.

58. Ke G., Meng Q., Finley T. et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (accessed 09/11/2024).

59. Lim B., Arık S. Ö., Loeff N. et al. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*. Vol. 37, Issue 4. P. 1748–1764. DOI:10.1016/j.ijforecast.2021.03.012.

60. Nugroho K. S., Megantara R. A., Alzami F. et al. Workload-Aware Performance Evaluation of Sequential and Parallel DAG-Based Machine Learning Orchestration on Single-Node Systems. *Sinkron*. Vol. 10, Issue 1. P. 725–740. DOI:10.33395/sinkron.v10i1.15788.

61. Olivares K. G., Challu C., Marcjasz G. et al. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting*. Vol. 39, Issue 2. P. 884–900. DOI:10.1016/j.ijforecast.2022.03.001.

62. ONNX Concepts - ONNX 1.20.0 documentation. URL: https://onnx.ai/onnx/intro/concepts.html (accessed 19/10/2025).

63. Paszke A., Gross S., Massa F. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*Curran Associates, Inc., 2019. URL: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (accessed 12/10/2025).

64. Patel J. Self-Healing Mechanisms in Software Development- A Machine Learning Method. *INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING & APPLIED SCIENCES*. Vol. 6, Issue 3. P. 48–54. DOI:10.55083/irjeas.2018.v06i03014.

65. Qiu Q., Liu H. Numerical Embedding of Categorical Features in Tabular Data: A Survey. *2023 International Conference on Machine Learning and Cybernetics (ICMLC)2023 International Conference on Machine Learning and Cybernetics (ICMLC)*. Adelaide, Australia : IEEE, 2023. DOI:10.1109/ICMLC58545.2023.10327921. P. 446–451.

66. Ravindra Reddy Madireddy. Self-Healing RPA Systems: A Reliability-Centric Architecture for Financial Enterprises. *International Journal of Computational and Experimental Science and Engineering*. Vol. 12, Issue 1. DOI:10.22399/ijcesen.4622.

67. Rocklin M. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. *Python in Science Conference*. Austin, Texas, 2015DOI:10.25080/Majora-7b98e3ed-013. P. 126–132.

68. Rodriguez S., Thangarajah J., Davey A. Design Patterns for Explainable Agents (XAg). *Adaptive Agents and Multi-Agent Systems*URL: https://api.semanticscholar.org/CorpusID:269527950

69. Sculley D., Holt G., Golovin D. et al. Hidden Technical Debt in Machine Learning Systems. *Advances in Neural Information Processing Systems*Curran Associates, Inc., 2015. URL: https://papers.nips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html (accessed 25/10/2025).

70. Shashank Pasupuleti. Robotic Process Automation for Enhancing Workflow Automation in Multi-System Environments. 05.06.2024. DOI:10.5281/ZENODO.14598763.

71. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*. Vol. 404, 01.03.2020. P. 132306. DOI:10.1016/j.physd.2019.132306.

72. The Kubeflow Authors. Kubeflow Pipelines. URL: https://github.com/kubeflow/pipelines 2025.

73. Wolpert D. H. Stacked generalization. *Neural Networks*. Vol. 5, Issue 2. P. 241–259. DOI:10.1016/S0893-6080(05)80023-1.

74. Zaharia M. A., Chen A., Davidson A. et al. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.* Vol. 41, 2018. P. 39–45.

75. Zaharia M., Xin R. S., Wendell P. et al. Apache Spark: a unified engine for big data processing. *Commun. ACM*. Vol. 59, Issue 11. P. 56–65. DOI:10.1145/2934664.

76. Abadi M., Barham P., Chen J. et al. TensorFlow: A system for large-scale machine learning. arXiv, 2016. DOI:10.48550/arXiv.1605.08695.

77. Akiba T., Sano S., Yanase T. et al. Optuna: A Next-generation Hyperparameter Optimization Framework. arXiv, 2019. URL: http://arxiv.org/abs/1907.10902 (accessed 10/11/2024).

78. Arik S. O., Pfister T. TabNet: Attentive Interpretable Tabular Learning. arXiv, 2020. URL: http://arxiv.org/abs/1908.07442 (accessed 10/11/2024).

79. Challu C., Olivares K. G., Oreshkin B. N. et al. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. arXiv, 2022. DOI:10.48550/arXiv.2201.12886.

80. Chang S., Zhang Y., Han W. et al. Dilated Recurrent Neural Networks. arXiv, 2017. DOI:10.48550/arXiv.1710.02224.

81. Chung J., Gulcehre C., Cho K. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv, 2014. DOI:10.48550/arXiv.1412.3555.

82. Doshi-Velez F., Kim B. Towards A Rigorous Science of Interpretable Machine Learning. arXiv, 2017. DOI:10.48550/arXiv.1702.08608.

83. Duan T., Avati A., Ding D. Y. et al. NGBoost: Natural Gradient Boosting for Probabilistic Prediction. arXiv, 2020. DOI:10.48550/arXiv.1910.03225.

84. Hansen N. The CMA Evolution Strategy: A Tutorial. arXiv, 2023. URL: http://arxiv.org/abs/1604.00772 (accessed 10/11/2024).

85. Lundberg S., Lee S.-I. A Unified Approach to Interpreting Model Predictions. arXiv, 2017. DOI:10.48550/arXiv.1705.07874.

86. Luo J., Xu S. NCART: Neural Classification and Regression Tree for Tabular Data. arXiv, 2024. URL: http://arxiv.org/abs/2307.12198 (accessed 10/11/2024).

87. Luong M.-T., Pham H., Manning C. D. Effective Approaches to Attention-based Neural Machine Translation. arXiv, 2015. DOI:10.48550/arXiv.1508.04025.

88. Marcos-Mercadé J., Lopez-Novoa U., Aranguren M. E. An Empirical Evaluation of Modern MLOps Frameworks. arXiv, 2026. DOI:10.48550/ARXIV.2601.20415.

89. Martin I. C., Mukherjee S., Baimagambetov A. et al. Evolving Machine Learning in Non-Stationary Environments: A Unified Survey of Drift, Forgetting, and Adaptation. arXiv, 2026. DOI:10.48550/arXiv.2505.17902.

90. Oord A. van den, Dieleman S., Zen H. et al. WaveNet: A Generative Model for Raw Audio. arXiv, 2016. DOI:10.48550/arXiv.1609.03499.

91. Oreshkin B. N., Carpov D., Chapados N. et al. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv, 2020. DOI:10.48550/arXiv.1905.10437.

92. Prokhorenkova L., Gusev G., Vorobev A. et al. CatBoost: unbiased boosting with categorical features. arXiv, 2019. DOI:10.48550/arXiv.1706.09516.

93. Sserujongi R., Ogenrwot D., Niwamanya N. et al. Design and Evaluation of a Scalable Data Pipeline for AI-Driven Air Quality Monitoring in Low-Resource Settings. arXiv, 2025. DOI:10.48550/ARXIV.2508.14451.

94. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need. arXiv, 2017. DOI:10.48550/arXiv.1706.03762.

95. Watanabe S. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. arXiv, 2023. URL: http://arxiv.org/abs/2304.11127 (accessed 10/11/2024).

96. Wu H., Xu J., Wang J. et al. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. arXiv, 2022. DOI:10.48550/arXiv.2106.13008.

97. Yang Y., Morillo I. G., Hospedales T. M. Deep Neural Decision Trees. arXiv, 2018. URL: http://arxiv.org/abs/1806.06988 (accessed 10/11/2024).

98. Zhang H., Si S., Hsieh C.-J. GPU-acceleration for Large-scale Tree Boosting. arXiv, 2017. DOI:10.48550/arXiv.1706.08359.

99. Zhou H., Zhang S., Peng J. et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. arXiv, 2021. DOI:10.48550/arXiv.2012.07436.

100. Zhou T., Ma Z., Wen Q. et al. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. arXiv, 2022. DOI:10.48550/arXiv.2201.12740.

ДОДАТКИ

ДОДАТОК А

Акти та довідки впровадження результатів дисертаційного дослідження

ЗАТВЕРДЖУЮ
Директор
ТОВ«ОСТО-ІНВЕСТ»
Шуплина Ірина Михайлівна
«ОСТО-ІНВЕСТ»
42562019

Акт № 1 від 30.09.2025
використання наукових результатів дисертаційного дослідження Остапюка Володимира Вікторовича,поданої на здобуття вченого звання «Доктор філософії» за спеціальністю «Інженерія програмного забезпечення»

Складений комісією у складі інженерів: голова комісії Походенко Ігор Васильович, члени комісії: Гнатюк Анатолій Володимирович, Коломієць Микола Васильович

У період з 10 серпня до 30 вересня 2025 року комісія розглянула наукові результати дисертації Остапюка Володимира Вікторовича, виконаної на кафедрі програмного забезпечення автоматизованих систем Черкаського державного технологічного університету.

Комісія встановила, що результати дисертаційної роботи В.В. Остапюка використані у діяльності ТОВ «ОСТО – ІНВЕСТ» у вигляді:

1. Технічної пропозиції щодо впровадження нового методу синтезу моделей
2. Технічної пропозиції щодо удосконалення процесів формування структур модельних шарів

За результатами спільної роботи створена інформаційна система інтелектуального моніторингу та аналізу даних, що реалізована у вигляді автоматизованої моделі підтримки прийняття рішень.

Зазначений інструментарій застосовується для оптимізації планування та оперативного управління матеріальними ресурсами при виконанні будівельних робіт. Це дозволило мінімізувати витрат через комплексне врахування якісних показників, логістичних витрат та діючих бюджетних обмежень у єдиному розрахунковому середовищі.

Голова комісії _____ Походенко І.В.

Члени комісії

Гнатюк А.В. _____

Коломієць М.В. _____

ЗАТВЕРДЖУЮ
Директор ТОВ РМ
КОНСАЛТИНГ ГРУП

Ф.О. Шебалін

Акт
впровадження результатів дисертаційних досліджень
Остапюка Володимира Вікторовича
що подається на здобуття вченого звання «Доктор філософії» за спеціальністю
121- Інженерія програмного забезпечення

Ми, що нижче підписалися, аналітик з бізнес-процесів (ФОП) Пугач Ірина Михайлівна, діловод (фахівець з інформаційно-аналітичного супроводу публічних закупівель) Мороз Наталія Іванівна, склали цей акт про те, що результати дисертаційних досліджень В.В. Остапюка:

- новий метод багатошарового синтезу агентних моделей та його реалізації при проектуванні програмного забезпечення інформаційної технології інтелектуального моніторингу;
- процеси підвищення різноманітності шарів моделі при побудові програмних агентів:
- використання спрямованого ациклічного графу (DAG) у процесі проєктування алгоритмів синтезу моделей та їх програмної реалізації.

експериментально впроваджено у діяльність ТОВ «РМ КОНСАЛТИНГ ГРУП» при наданні консультаційних послуг у сфері публічних закупівель.

На основі запропонованих методів та процесів реалізовано механізм оцінювання та попереднього відбору тендерних процедур за технічними, фінансовими та організаційними критеріями, що дозволило підвищити оперативність аналізу тендерної документації та покращити процес прийняття рішень щодо участі у закупівлях.

Акт впровадження обговорено і схвалено на робочій нараді ТОВ РМ КОНСАЛТИНГ ГРУП (протокол № 02/03/26 від «02» березня 2026 року.)

Акт складено 02.03.2026 року

Аналітик з бізнес-процесів (ФОП) _____І.М. Пугач

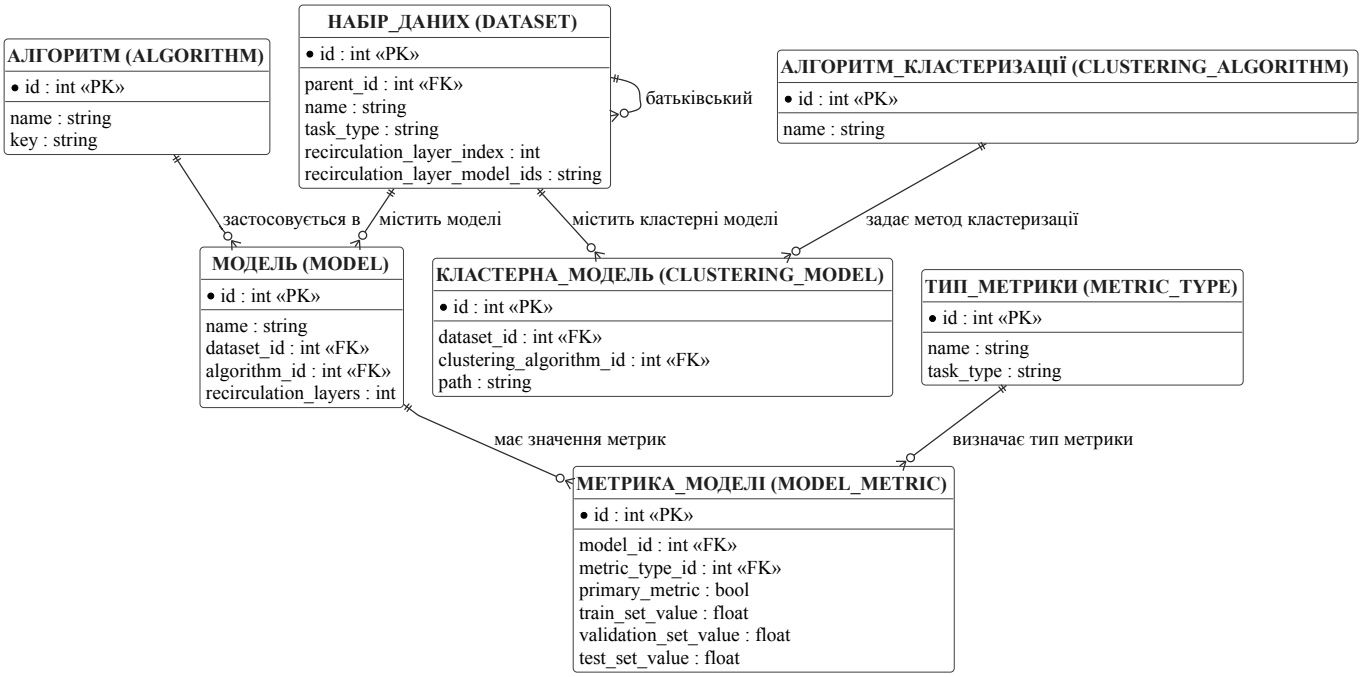
Діловод (фахівець з інформаційно-аналітичного супроводу публічних закупівель) _____Н.І. Мороз

ДОДАТОК Б

Діаграма активності повного процесу експерименту

ДОДАТОК В

Ключові сутності збереження результатів



ДОДАТОК Г

Приклад DSL JSON файлу


```
{
  "dataset": {
    "path": "../data/taxi.csv",
    "target": "Trip_Price",
    "task_type": "regression"
  },
  "evaluation": {
    "metrics": [
      "rmse",
      "mae",
      "mape",
      "r2"
    ]
  },
  "model": {
    "name": "ridge"
  },
  "target": {
    "scaler": null,
    "skewness": null
  },
  "preprocessing": {
    "numerical": {
      "scaler": "robust",
      "nan": "mean"
    },
    "categorical": {
      "encoding": "onehot",
      "nan": "most_frequent"
    }
  },
  "cluster_homogenization": {
    "enabled": true,
    "method": "kmeans",
    "cluster_count": 3,
    "include_cluster_id": true,
    "include_membership": false,
    "include_segment_predictions": true,
    "only_homogenized_features": false,
    "segment_prediction_prefix": "seg_pred_",
    "canonical_naming": true,
    "segment_algorithms": [
      "xgboost",
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
      "lasso",
      "lgbm"
    ]
  },
  "recirculation": {
    "enabled": true,
    "enhanced": true,
    "only_best_model_output": false,
    "layers": [
      {
        "algorithms": [
          "xgboost",
```

```
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
      "lasso",
      "lgbm"
    ],
    "max_models": 1,
    "gating": "none"
  },
  {
    "algorithms": [
      "xgboost",
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
      "lasso",
      "lgbm"
    ],
    "max_models": 1,
    "gating": "none"
  },
  {
    "algorithms": [
      "xgboost",
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
      "lasso",
      "lgbm"
    ],
    "max_models": 3,
    "gating": "average"
  },
  {
    "algorithms": [
      "xgboost",
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
      "lasso",
      "lgbm"
    ],
    "max_models": 2,
    "gating": "none"
  },
  {
    "algorithms": [
      "xgboost",
      "random_forest",
      "ridge",
      "gb_boost",
      "decision_tree",
      "extra_tree",
      "elastic_net",
```

```
          "lasso",
          "lgbm"
        ],
        "max_models": 2,
        "gating": "weighted",
        "weights": [
          0.7,
          0.3
        ]
      }
    ],
    "canonical_recirculation_naming": true
  }
}
```

ДОДАТОК Е

Приклад експортованого DAG файлу


```
{
  "entry": "final_series",
  "nodes": [
    {
      "id": "input_0",
      "type": "input",
      "config": {
        "typed": true,
        "base_dir": "."
      },
      "inputs": []
    },
    {
      "id": "seg_pred_0",
      "type": "segment_predictions",
      "inputs": [
        "input_0"
      ],
      "config": {
        "canonical": true,
        "output_feature": "Trip_Price",
        "manifest_inline": {
          "segments": [
            {
              "index": 0,
              "model_path":
"3647d721451743d395352c894db7d2e3dsl_model_clusters_cluster_0_random_forest_random_forest.jobl
ib",
              "input_pipeline_path":
"76677fb41eb54846b3404e579d23a103dsl_model_clusters_cluster_0_random_forest.joblib",
              "class_name": "RandomForestRegressor"
            }
          ],
          "include_segment_predictions": true,
          "segment_prediction_prefix": "seg_pred_",
          "canonical_naming": true
        },
        "base_dir": "."
      }
    },
    {
      "id": "seg_pred_1",
      "type": "segment_predictions",
      "inputs": [
        "input_0"
      ],
      "config": {
        "canonical": true,
        "output_feature": "Trip_Price",
        "manifest_inline": {
          "segments": [
            {
              "index": 1,
              "model_path":
"7bc95cc9ccc0409e8224dc93da937b3fdsl_model_clusters_cluster_1_gb_boost_gb_boost.joblib",
              "input_pipeline_path":
"254fae0e268d423e99f5ff2945a54aa9dsl_model_clusters_cluster_1_gb_boost.joblib",
              "class_name": "GradientBoostingRegressor"
            }
          ],
          "include_segment_predictions": true,
```

```
          "segment_prediction_prefix": "seg_pred_",
          "canonical_naming": true
        },
        "base_dir": "."
      }
    },
    {
      "id": "seg_pred_2",
      "type": "segment_predictions",
      "inputs": [
        "input_0"
      ],
      "config": {
        "canonical": true,
        "output_feature": "Trip_Price",
        "manifest_inline": {
          "segments": [
            {
              "index": 2,
              "model_path":
"f50b829fa6c544b980e84d06f09a564bdsl_model_clusters_cluster_2_ridge_ridge.joblib",
              "input_pipeline_path":
"bfdd1a81a2214870ada85c78d59cd73ddsl_model_clusters_cluster_2_ridge.joblib",
              "class_name": "Ridge"
            }
          ],
          "include_segment_predictions": true,
          "segment_prediction_prefix": "seg_pred_",
          "canonical_naming": true
        },
        "base_dir": "."
      }
    },
    {
      "id": "cluster_assign_0",
      "type": "cluster_assignment",
      "config": {
        "canonical": true,
        "output_feature": "Trip_Price",
        "manifest_inline": {
          "include_cluster_id": true,
          "classifier_model_path":
"04b7ef2557f540548a6772971b9ca1d9dsl_model_clusters_cluster_classifier_extra_tree_extra_tree.j
oblib",
          "classifier_input_pipeline_path":
"cf9dedcd22fb4f45bc7ed89a2e93a07fdsl_model_clusters_cluster_classifier_extra_tree.joblib",
          "classifier_feature_name": "cluster_id_pred",
          "canonical_naming": true,
          "class_name": "ExtraTreesClassifier"
        },
        "base_dir": "."
      },
      "inputs": [
        "input_0"
      ]
    },
    {
      "id": "cluster_homog_assemble_0",
      "type": "cluster_homogenization_assemble",
      "inputs": [
        "input_0",
        "cluster_assign_0",
        "seg_pred_0",
        "seg_pred_1",
        "seg_pred_2"
      ],
```

```
      "config": {
        "canonical": true,
        "output_feature": "Trip_Price",
        "manifest_inline": {
          "feature_order": [
            "Trip_Distance_km",
            "Time_of_Day",
            "Day_of_Week",
            "Passenger_Count",
            "Traffic_Conditions",
            "Weather",
            "Base_Fare",
            "Per_Km_Rate",
            "Per_Minute_Rate",
            "Trip_Duration_Minutes",
            "Trip_Price_cluster_0_prediction",
            "Trip_Price_cluster_1_prediction",
            "Trip_Price_cluster_2_prediction",
            "predicted_cluster"
          ],
          "include_cluster_id": true,
          "segment_prediction_prefix": "seg_pred_",
          "classifier_feature_name": "cluster_id_pred",
          "canonical_naming": true
        },
        "base_dir": "."
      }
    },
    {
      "id": "l0_m0",
      "type": "single_model",
      "inputs": [
        "cluster_homog_assemble_0"
      ],
      "config": {
        "model_path":
"7c0cbfe3e85740f9a97ea475e6894810dsl_model_layer_0_gb_boost_gb_boost.joblib",
        "input_pipeline_path":
"6e89dfaf3daa4c439916a8dff898b63bdsl_model_layer_0_gb_boost.joblib",
        "aggregate_aliases": [
          "recirc_layer_0_model_0_prediction_layer_0"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"5d6aeeb073774205aaf1ad424a435f64dsl_model_layer_0_gb_boost_target.joblib",
        "class_name": "GradientBoostingRegressor"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "recirc_layer_0",
      "type": "recirculation_layer",
      "inputs": [
        "cluster_homog_assemble_0",
        "l0_m0"
      ],
      "config": {
        "layer_index": 0,
        "aggregation": "concat",
        "expose": "aggregate_only",
        "aggregate_aliases": [
          "recirc_layer_0_model_0_prediction_layer_0"
        ],
        "model_prediction_aliases": [
          "Trip_Price_dsl_model_layer_0_gb_boost_prediction_layer_0"
```

```
        ],
        "average_prediction_alias": null,
        "output_feature": "Trip_Price",
        "canonical": true,
        "gating_mode": "none",
        "only_best_model_output": false,
        "base_dir": "."
      }
    },
    {
      "id": "l1_m0",
      "type": "single_model",
      "inputs": [
        "recirc_layer_0"
      ],
      "config": {
        "model_path":
"c01dd9ab45964ab5b23164f97142c972dsl_model_layer_1_decision_tree_decision_tree.joblib",
        "input_pipeline_path":
"5a7f331961b7446faabb39417a7c02b5dsl_model_layer_1_decision_tree.joblib",
        "aggregate_aliases": [
          "recirc_layer_1_model_0_prediction_layer_1"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"682251103a9f4e76809ea48907f18952dsl_model_layer_1_decision_tree_target.joblib",
        "class_name": "DecisionTreeRegressor"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "recirc_layer_1",
      "type": "recirculation_layer",
      "inputs": [
        "recirc_layer_0",
        "l1_m0"
      ],
      "config": {
        "layer_index": 1,
        "aggregation": "concat",
        "expose": "aggregate_only",
        "aggregate_aliases": [
          "recirc_layer_1_model_0_prediction_layer_1"
        ],
        "model_prediction_aliases": [
          "Trip_Price_dsl_model_layer_1_decision_tree_prediction_layer_1"
        ],
        "average_prediction_alias": null,
        "output_feature": "Trip_Price",
        "canonical": true,
        "gating_mode": "none",
        "only_best_model_output": false,
        "base_dir": "."
      }
    },
    {
      "id": "l2_m0",
      "type": "single_model",
      "inputs": [
        "recirc_layer_1"
      ],
      "config": {
        "model_path": "5604f826bbce49a9a96949cc42bf800edsl_model_layer_2_ridge_ridge.joblib",
        "input_pipeline_path":
"84d05cad837c466bb77c27d488049611dsl_model_layer_2_ridge.joblib",
```

```
        "aggregate_aliases": [
          "recirc_layer_2_mean_prediction_layer_2"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"9cf2f996b94d493497004e09851159d1dsl_model_layer_2_ridge_target.joblib",
        "class_name": "Ridge"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "l2_m1",
      "type": "single_model",
      "inputs": [
        "recirc_layer_1"
      ],
      "config": {
        "model_path": "30f2b46190cd468bbf1758a81b5fc2c3dsl_model_layer_2_lasso_lasso.joblib",
        "input_pipeline_path":
"c33357644d7c42f98cfe7d2cf7d895eadsl_model_layer_2_lasso.joblib",
        "aggregate_aliases": [
          "recirc_layer_2_mean_prediction_layer_2"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"3e21ac050e7b47a38451ab4d95396b37dsl_model_layer_2_lasso_target.joblib",
        "class_name": "Lasso"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "l2_m2",
      "type": "single_model",
      "inputs": [
        "recirc_layer_1"
      ],
      "config": {
        "model_path":
"5eaba70fb03a43b6baa9965069c71ea8dsl_model_layer_2_random_forest_random_forest.joblib",
        "input_pipeline_path":
"a84630514314413a96e6a0171ba2f5fcdsl_model_layer_2_random_forest.joblib",
        "aggregate_aliases": [
          "recirc_layer_2_mean_prediction_layer_2"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"1f54211521d6426480d4ed2eb7f4e7a9dsl_model_layer_2_random_forest_target.joblib",
        "class_name": "RandomForestRegressor"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "recirc_layer_2",
      "type": "recirculation_layer",
      "inputs": [
        "recirc_layer_1",
        "l2_m0",
        "l2_m1",
        "l2_m2"
      ],
      "config": {
        "layer_index": 2,
        "aggregation": "mean",
```

```
        "expose": "aggregate_only",
        "aggregate_aliases": [
          "recirc_layer_2_mean_prediction_layer_2"
        ],
        "model_prediction_aliases": [
          "Trip_Price_dsl_model_layer_2_ridge_prediction_layer_2",
          "Trip_Price_dsl_model_layer_2_lasso_prediction_layer_2",
          "Trip_Price_dsl_model_layer_2_random_forest_prediction_layer_2"
        ],
        "average_prediction_alias": "Trip_Price_average_prediction_layer_2",
        "output_feature": "Trip_Price",
        "canonical": true,
        "gating_mode": "average",
        "only_best_model_output": false,
        "base_dir": "."
      }
    },
    {
      "id": "l3_m0",
      "type": "single_model",
      "inputs": [
        "recirc_layer_2"
      ],
      "config": {
        "model_path": "de5b3cec4f7e42d7b8295e98fd4b98eadsl_model_layer_3_ridge_ridge.joblib",
        "input_pipeline_path":
"999651de02e3435db32fd91103c43805dsl_model_layer_3_ridge.joblib",
        "aggregate_aliases": [
          "recirc_layer_3_model_0_prediction_layer_3",
          "recirc_layer_3_model_1_prediction_layer_3"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"d55c12e25e504fe78dca0af5cf0afe8ddsl_model_layer_3_ridge_target.joblib",
        "class_name": "Ridge"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "l3_m1",
      "type": "single_model",
      "inputs": [
        "recirc_layer_2"
      ],
      "config": {
        "model_path": "d724a327bd95466392ac8e335afe80abdsl_model_layer_3_lasso_lasso.joblib",
        "input_pipeline_path":
"37e67e417dba4828a55c9c8836f79187dsl_model_layer_3_lasso.joblib",
        "aggregate_aliases": [
          "recirc_layer_3_model_0_prediction_layer_3",
          "recirc_layer_3_model_1_prediction_layer_3"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"6ed8c4e0f9ae4214b5ab7fb49d9e78f7dsl_model_layer_3_lasso_target.joblib",
        "class_name": "Lasso"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "recirc_layer_3",
      "type": "recirculation_layer",
      "inputs": [
        "recirc_layer_2",
```

```
        "l3_m0",
        "l3_m1"
      ],
      "config": {
        "layer_index": 3,
        "aggregation": "concat",
        "expose": "aggregate_only",
        "aggregate_aliases": [
          "recirc_layer_3_model_0_prediction_layer_3",
          "recirc_layer_3_model_1_prediction_layer_3"
        ],
        "model_prediction_aliases": [
          "Trip_Price_dsl_model_layer_3_ridge_prediction_layer_3",
          "Trip_Price_dsl_model_layer_3_lasso_prediction_layer_3"
        ],
        "average_prediction_alias": null,
        "output_feature": "Trip_Price",
        "canonical": true,
        "gating_mode": "none",
        "only_best_model_output": false,
        "base_dir": "."
      }
    },
    {
      "id": "l4_m0",
      "type": "single_model",
      "inputs": [
        "recirc_layer_3"
      ],
      "config": {
        "model_path": "aeb3af71601841c1917f407db6c27d55dsl_model_layer_4_ridge_ridge.joblib",
        "input_pipeline_path":
"13eb34a875884d7fb105b245a6747d12dsl_model_layer_4_ridge.joblib",
        "aggregate_aliases": [
          "recirc_layer_4_mean_prediction_layer_4"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"e4d0a7fbdfaf4cd4937626e74792ebe7dsl_model_layer_4_ridge_target.joblib",
        "class_name": "Ridge"
      },
      "output_feature": "Trip_Price"
    },
    {
      "id": "l4_m1",
      "type": "single_model",
      "inputs": [
        "recirc_layer_3"
      ],
      "config": {
        "model_path":
"053e43d4650d4841ba96a2b46f5313ecdsl_model_layer_4_extra_tree_extra_tree.joblib",
        "input_pipeline_path":
"e9cfde34bbce4a858e1a418e9e79b8d3dsl_model_layer_4_extra_tree.joblib",
        "aggregate_aliases": [
          "recirc_layer_4_mean_prediction_layer_4"
        ],
        "canonical": true,
        "base_dir": ".",
        "target_pipeline_path":
"8cf02d67d46c4065a7ef125c315f39cfdsl_model_layer_4_extra_tree_target.joblib",
        "class_name": "ExtraTreesRegressor"
      },
      "output_feature": "Trip_Price"
    },
```

```
    {
      "id": "final_series",
      "type": "result_selector",
      "inputs": [
        "l4_m0",
        "l4_m1"
      ],
      "config": {
        "canonical": true,
        "aggregation": "weighted",
        "weights": [
          0.7,
          0.3
        ],
        "base_dir": "."
      }
    }
  ],
  "schema_version": "0.1.0",
  "strict": true,
  "typed": true
}
```