ISSN 2707-1898 (print)
ISSN 2786-9172 (online)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2025.02.001
Article received 14.10.2025 p.
Article accepted 30.10.2025 p.
UDC 004.94

Correspondence author
@ V. V. Ostapiuk
v.v.ostapiuk.asp22@chdtu.edu.ua

С. В. Голуб, В. В. Остапюк
Черкаський державний технологічний університет, м. Черкаси, Україна

ДЕКЛАРАТИВНИЙ ПІДХІД ДО ПРОЄКТУВАННЯ ТА ВІДТВОРЮВАНОГО НАВЧАННЯ СКЛАДНИХ МОДЕЛЬНИХ СТРУКТУР ДЛЯ МОНІТОРИНГОВИХ ПРОГРАМНИХ АГЕНТІВ

Розроблення ефективних моніторингових програмних агентів, що є важливими компонентами сучасних мультиагентних систем (МАС), все більше покладається на складні модельні структури, такі як багатошарові ансамблі машинного навчання. Однак зростання складності цих архітектур створює значні виклики стосовно забезпечення надійності, аудитоздатності та, що найважливіше, відтворюваності експериментальних результатів. Для вирішення цього завдання у статті запропоновано декларативний підхід, зосереджений на новоствореній предметно-орієнтованій мові (DSL). Ця мова надає структурований, зрозумілий формат для описання усього процесу побудови моделі: від підготовки даних та гіперпараметричної оптимізації до складної конфігурації багатошарових ансамблів, ураховуючи механізми рециркуляції та підвищення однорідності даних. Розроблено програмну систему, що інтерпретує цю DSL для автоматизації процесу навчання. Ключовою особливістю цього процесу є автоматична генерація самодостатнього, відтворюваного пакета, що містить не лише серіалізовані моделі, а й усі пов'язані конфігурації, метрики продуктивності та детальні дані про походження. Основні результати демонструють, що цей декларативний підхід дає змогу ефективно управляти складними просунутими експериментами, забезпечує цілісність створених моделей та гарантує їхню повну відтворюваність. Також було встановлено, що формалізація експериментальних налаштувань у DSL надає надійну основу для об'єктивного порівняння різних модельних архітектур. Загалом запропонований DSL-орієнтований підхід створює надійну та аудитоздатну основу для розроблення та валідації ефективних програмних агентів, долаючи критичний розрив між алгоритмічними дослідженнями та практичною потребою у надійних системах машинного навчання, що можна розгорнути.

Ключові слова: мультиагентні системи, агентно-орієнтований моніторинг, предметно-орієнтована мова (DSL), відтворюваність моделей, алгоритм синтезу моделей, багатошарові модельні структури.

Вступ / Introduction

Сучасні системи інтелектуального моніторингу часто реалізують на основі архітектури мультиагентних систем (МАС) для створення гнучких та автономних рішень [1]. Ключовим елементом таких систем є моніторинговий програмний агент, ефективність якого визначається якістю його внутрішніх модельних структур, що відповідають за прогнозування, класифікацію та інші інтелектуальні функції. Для досягнення високої точності такі структури часто будують у вигляді складних, багатошарових ансамблів, які комбінують прогнози декількох окремих моделей. Для підвищення їхньої ефективності доведено дієвість методу багатошарової рециркуляції, запропонованого в [2] та удосконаленого в [3], а також досліджено методи підвищення однорідності під час роботи з різнорідними даними.

Постановка завдання дослідження. Дедалі більша складність модельних структур призводить до викликів стосовно надійності, відтворюваності та керованості їх створення та розгортання. Конфігурація такої структури, що охоплює декілька етапів передоброблення, десятки алгоритмів синтезу моделей (АСМ) та складні зв'язки між ними, занадто громіздка для ручного керування та схильна до помилок. Відсутність формалізованого опису істотно ускладнює відтворення, оновлення або перенесення модельної структури, що є одним із ключових завдань на шляху від дослідження до практичного застосування [4] .

Об'єкт дослідження – процес життєвого циклу (проєктування, навчання, валідація, розгортання) складних модельних структур для програмних агентів.

Предмет дослідження – методи та засоби декларативного опису та автоматизованої генерації відтворюваних послідовностей дій для навчання та розгортання моделей.

Метою роботи є розроблення та опис підходу, що ґрунтується на предметно-орієнтованій мові (DSL), який дає змогу спростити проєктування, підвищити надійність та забезпечити повну відтворюваність процесу створення складних модельних структур.

Для досягнення поставленої мети було визначено такі *завдання дослідження*:

Розробити схему DSL, здатної гнучко описувати складні ансамблеві архітектури.

Реалізувати програмний комплекс, що інтерпретує DSL для автоматизації процесу навчання.

Розробити механізм генерації відтворюваного пакета, що містить усі необхідні моделі та метадані.

Продемонструвати роботу запропонованого підходу на прикладі конфігурації складної модельної структури.

Аналіз останніх досліджень та публікацій. Для обґрунтування актуальності та визначення невирішеної частини завдання доцільно розглянути чотири ключові напрями досліджень: проблему відтворюваності в машинному навчанні, наявні системи управління ML-експериментами (MLOps), підходи до автоматизованого машинного навчання (AutoML) та декларативні підходи в цій галузі.

Проблема відтворюваності у наукових дослідженнях з МН. Однією із гострих проблем сучасної науки про дані є відтворюваність результатів. Як зазначено у статті [4], багато висновків із робіт 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 межах концепції MLOps. Такі інструменти, як MLflow [5] та Kubeflow[6], надають потужні засоби автоматизації. MLflow дає змогувідстежувати параметри та метрики, керувати версіями моделей та розгортати їх. Kubeflow, своєю чергою, є комплекснішою платформою для оркестрації багатоетапних процесів навчання у середовищі Kubernetes.

Однак ці системи часто потребують значних інженерних зусиль для налаштування, а логіку експериментів у них описано переважно імперативно, тобто через написання програмного коду. Вони не надають зручних, декларативних засобів для описання саме *складних, кастомних ансамблевих архітектур*, таких як багатошарова рециркуляція чи підвищення однорідності, яка у центрі уваги нашого дослідження.

Автоматизоване машинне навчання (AutoML). Системи AutoML, такі як Auto-sklearn [7] або TPOT, йдуть ще далі в автоматизації, намагаючись самостійно знайти оптимальну модель та її гіперпараметри для заданого набору даних. Вони корисні для швидкого отримання базових рішень.

Незважаючи на потужність, їхнім головним недоліком часто є підхід "чорної скриньки": контроль дослідника над кінцевою архітектурою моделі обмежений. AutoML-системи не призначені для *декларативного проєктування*, де людина свідомо конструює складну модельну структуру (наприклад, багатошаровий ансамбль із певними моделями на кожному шарі), а система лише забезпечує надійне та відтворюване виконання цього проєкту.

Декларативні підходи та DSL у машинному навчанні. Ідея використання предметно-орієнтованих мов (DSL) для спрощення складних завдань не нова. Такі інструменти, як TensorFlow (через Keras) [8] або PyTorch [9], також використовують декларативні елементи для описання архітектури нейронних мереж. Дослідник описує шари мережі та зв'язки між ними, а фреймворк сам піклується про обчислення та оптимізацію.

Відомі декларативні підходи зосереджені переважно на описі *архітектури однієї моделі* (переважно нейронної мережі). Вони не охоплюють увесь життєвий цикл експерименту: від передоброблення до вибору з пулу різнорідних АСМ та автоматичної генерації повного, відтворюваного пакета з усіма артефактами. Отже, існує невирішене завдання у вигляді відсутності легкого, гнучкого та водночас строгого інструменту для декларативного проєктування всього експерименту зі складними, кастомними ансамблевими архітектурами, якої стосується ця робота.

Результати досліджень та їх обговорення / Research results and their discussion

Архітектура предметно-орієнтованої мови (DSL). Основою запропонованого підходу є предметно-орієнтована мова, реалізована у вигляді структурованої JSON-схеми. Такий формат надає розробнику потужний декларативний інструмент для повного та однозначного описання усієї архітектури створюваної модельної структури для програмного агента. Використання DSL чітко відокремлює етап логічного проєктування від програмної реалізації, що підвищує прозорість, гнучкість та знижує ймовірність помилок. Схема DSL містить декілька ключових секцій, що логічно описують кожен етап конструювання модельної структури:

- **dataset** та **splits**: визначення джерела даних та правил його розділення;
- **preprocessing**: конфігурація етапів передоброблення для різних типів ознак;
- **model**: визначення базового (однорівневого) АСМ;
- **cluster_homogenization**: описання процесу підвищення однорідності даних;
- **recirculation**: конфігурація багатошарових ансамблів;

- **search**: налаштування гіперпараметричної оптимізації;
- **artifacts**: керування збереженням результатів та артефактів.

Конструювання модельної структури за допомогою DSL відбувається послідовно, ніби розробник веде діалог із системою, крок за кроком визначаючи всі аспекти майбутньої моделі. Далі розглянемо призначення кожної із цих секцій детальніше.

Все починається із визначення даних та завдання, що вирішується. У секції **dataset** вказують шлях до набору даних (*path*), назву цільової змінної (*target*) та тип задачі (*task_type*) – чи то регресія, чи класифікація. Після цього у секції **splits** описують стратегію розділення даних на тренувальну та тестову вибірки. Тут можна вибрати тип розділення (*type*), наприклад, *random* для стандартних випадків, або *stratified* для задач класифікації із незбалансованими класами, щоб забезпечити збереження пропорцій класів у обох вибірках. Також вказують розмір тестової вибірки (*test_size*), що дає змогу гнучко керувати обсягом даних для навчання та оцінки.

Наступним логічним кроком є детальна конфігурація етапів передоброблення даних, що описується в секції **preprocessing**. Вона має окремі гілки для числових та категоріальних ознак, що відображає специфіку роботи із різнорідними даними. Для числових ознак розробник може задати методи заповнення пропусків (*nan*), наприклад, *mean* (середнім значенням) чи *median*, що є стійкішим до викидів. Визначають також тип масштабування (*scaler*), як-от *standard* (стандартизація), *minmax* (нормалізація до діапазону) чи *robust*, що використовує квантилі та є стійким до викидів. Опціонально, для даних із асиметричним розподілом, можна застосувати алгоритми корекції асиметрії (skewness), такі як перетворення Бокса – Кокса або Єо – Джонсона. Для категоріальних ознак, своєю чергою, визначають стратегії обробки пропусків (наприклад, *most_frequent* або *new_category*) та метод кодування (encoder), наприклад, *one_hot*. Такий деталізований контроль над передобробленням критично важливий, оскільки якість підготовки даних безпосередньо впливає на фінальний результат. Важливо, що вся логіка опрацювання даних тепер описана в єдиному конфігураційному файлі. Це робить процес повністю прозорим та відтворюваним, на відміну від підходів, де кроки передоброблення "зашиті" в програмний код і є неочевидними для зовнішнього аналізу. Такий декларативний опис також дає змогу легко порівнювати різні стратегії передобробки, змінюючи лише кілька параметрів у DSL, замість переписування великих частин коду, що значно прискорює дослідницький процес. Така гнучкість налаштування важлива, оскільки вибір методу обробки пропусків чи масштабування може істотно вплинути на ефективність різних АСМ. Наприклад, лінійні моделі чутливі до масштабу ознак, тоді як для АСМ на основі дерев рішень це не так критично. Наявність цих опцій у DSL дає розробнику змогу свідомо проєктувати повний цикл опрацювання даних, адаптований під конкретну модельну структуру, що є перевагою порівняно із жорсткішими, автоматизованими системами.

Після визначення даних та їх передоброблення у секції **model** задають базовий алгоритм синтезу моделей (АСМ) та його статичні параметри. Ця конфігурація використовується для навчання простої, **однорівневої моделі** і слугує точкою відліку для порівняння складніших архітектур. Її застосовують за замовчуванням, якщо в DSL не визначено секції recirculation або cluster_homogenization.

Ключовою особливістю розробленої DSL є вбудована підтримка складних, багатоетапних архітектур. Секція **cluster_homogenization** дає змогу декларативно описати процес підвищення однорідності даних. Розробник може вибрати метод початкової кластеризації (method), наприклад, kmeans або gmm, задати кількість кластерів (*cluster_count*) та детально визначити, які саме нові ознаки будуть згенеровані. DSL підтримує гнучку конфігурацію цього етапу через поля-перемикачі: *include_segment_predictions* (додає прогнози від спеціалізованих моделей для кожного сегмента) та *include_cluster_id* (додає прогнозовану мітку кластера). Для кожного із цих етапів також можливе здійснення індивідуальної гіперпараметричної оптимізації через поле *hpo_n_trials*.

Для побудови багатошарових ансамблів призначена секція **recirculation**. Вона надає два рівні гнучкості. Для швидкого налаштування можна використовувати загальні параметри, такі як *first_layer_max_models* та *next_layer_max_models*, що дають змогу задавати різну кількість найкращих базових моделей для першого та наступних шарів ансамблю. Для глибшого контролю передбачено секцію **layers**, де можна конфігурувати **кожен шар рециркуляції індивідуально**. У межах кожного шару можна визначити точну кількість моделей (*max_models*), набір використовуваних АСМ (*algorithms*), а також стратегію агрегації прогнозів (*gating*), наприклад, просте (*average*) або зважене (*weighted*) усереднення. Така дворівнева система конфігурації дає змогу як швидко прототипувати, так і детально проєктувати складні гетерогенні ансамблі. Введення таких параметрів, як індивідуальний вибір АСМ та налаштування gating для кожного шару, є ключовою перевагою, що відрізняє цю DSL від простіших підходів до ансамблювання. Це надає можливість реалізовувати складні стратегії, де, наприклад, початкові шари використовують потужні нелінійні моделі для виявлення складних залежностей, а фінальні – прості лінійні моделі для стабільної агрегації. По суті, цей механізм є гнучкою реалізацією ідеї стекованої генералізації (stacked generalization)[10].

Нарешті, **пошук оптимальних гіперпараметрів та збереження результатів** конфігуруються у секціях search та artifacts. Секція *search* дає змогу опціонально активувати гіперпараметричну оптимізацію. Система містить попе-

редньо визначені простори пошуку для кожного із підтримуваних ACM, що спрощує налаштування, але розробник може доповнювати або перевизначати їх через поле space. Секція *artifacts*, своєю чергою, керує збереженням усіх результатів роботи. Вона дає можливість вказати вихідну директорію (*output_dir*), визначити необхідність збереження навчених моделей та налаштувати експорт знімка бази даних експериментів (*db_export*) для забезпечення максимальної відтворюваності.

Метод гіперпараметричної оптимізації. Для автоматизації процесу пошуку оптимальних гіперпараметрів, що важливо для досягнення максимальної точності моделей, у запропонований підхід інтегровано фреймворк **Optuna** [11]. Вибір цього інструменту зумовлений його гнучкістю, ефективними алгоритмами вибору (зокрема, Tree-structured Parzen Estimator, TPE) та можливістю легкого інтегрування у складні процеси навчання. Важливою особливістю реалізації є принцип **"вимкнено за замовчуванням"**. Гіперпараметрична оптимізація (ГПО) активується лише тоді, коли у відповідній секції DSL явно вказано ненульову кількість ітерацій (*n_trials*). Це забезпечує розробнику гнучкість: можна виконувати як швидкі експерименти зі стандартними параметрами для перевірки гіпотез, так і глибші дослідження із автоматичним підбиранням параметрів.

Підхід повністю детермінований: за фіксованого глобального початкового значення (*random_state*), що задають на початку DSL-файлу, результати оптимізації є повністю відтворюваними. Це забезпечується передаванням початкового значення до об'єктів Optuna, що гарантує однакову послідовність випробувань під час кожного запуску із однаковою конфігурацією.

Деталізація налаштувань дає змогу застосовувати ГПО із різною кількістю проб (*hpo_n_trials*) до окремих компонентів модельної структури, що є однією з ключових переваг розробленої DSL:

- **Глобальна оптимізація:** задають в секції search і застосовують до основної, однорівневої моделі.
- **Локальна оптимізація:** у секції *cluster_homogenization* можна визначити окремий бюджет для оптимізації моделей для сегментів та класифікатора кластерів.
- **Пошарова оптимізація:** у секції *recirculation* можна задати як загальний бюджет для всіх шарів, так і індивідуальні бюджети для кожного шару окремо, що дає змогу інтенсивніше оптимізувати, наприклад, перший, найважливіший шар ансамблю.

Така гнучкість дає змогу зосередити обчислювальні ресурси на оптимізації найкритичніших компонентів модельної структури, що важливо для ефективного використання ресурсів.

Процес навчання, керований DSL. Перетворення декларативного опису в DSL на навчену модельну структуру та відтворюваний пакет виконує спеціалізований програмний комплекс. Його робота складається з кількох логічних етапів, що забезпечують надійність та послідовність виконання.

1. Валідація та парсинг DSL. На першому кроці конфігураційний JSON-файл перевіряють на відповідність схемі DSL. Система контролює коректність типів даних, діапазонів значень (наприклад, *n_trials* від 1 до 200), а також логічні залежності між параметрами (наприклад, *cluster_count* є обов'язковим для методу *kmeans*). Крім синтаксичної валідації, система виконує і семантичні перевірки. Наприклад, вона контролює, щоб кількість алгоритмів, вказаних для шару рециркуляції, була достатньою для відбору *max_models* моделей, або щоб ваги для *weighted* gating були коректно визначені. Такий дворівневий контроль гарантує, що до виконання надійде лише логічно узгоджена та синтаксично правильна конфігурація, що підвищує надійність всього процесу та запобігає помилкам під час тривалого виконання Це дає змогу виявити помилки конфігурації на ранньому етапі, до початку ресурсоємних обчислень.

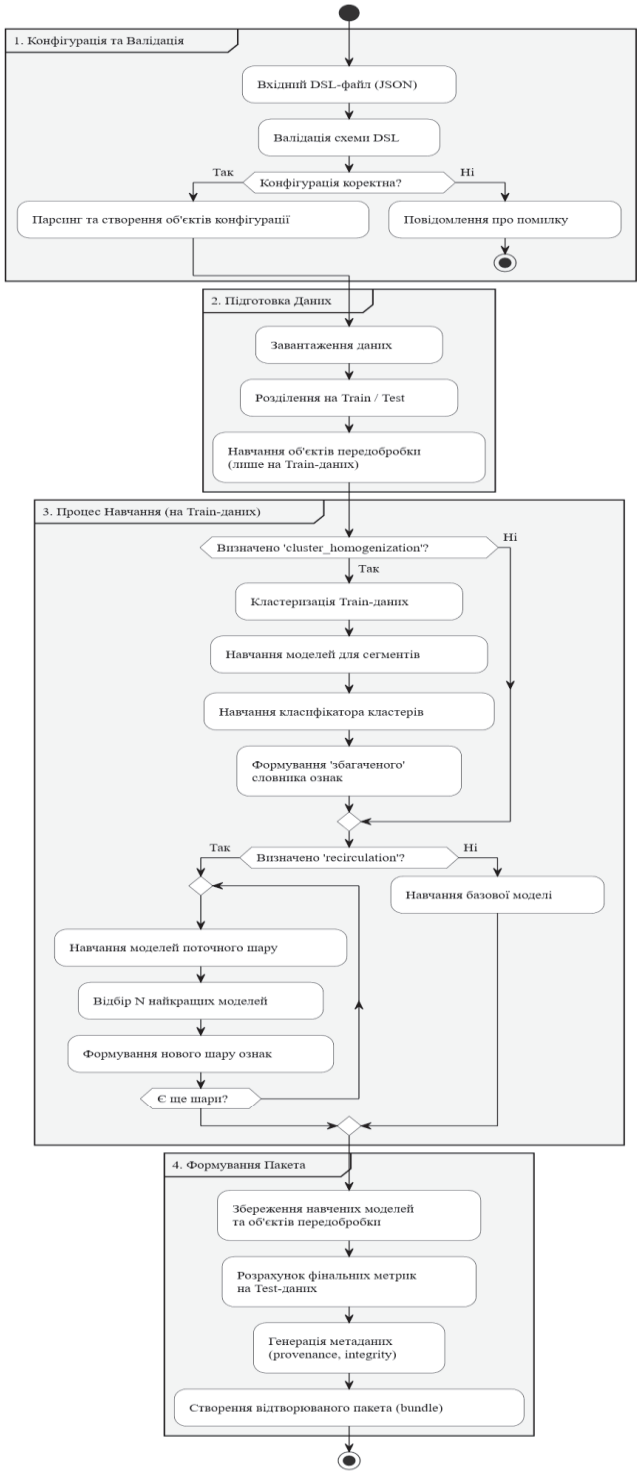
2. Підготовка даних. На основі секцій *dataset*, *splits* та *preprocessing* виконують всі необхідні операції з даними. Спочатку дані завантажують та розділяють на тренувальну та тестову вибірки. Потім, на основі конфігурації в *preprocessing*, будують об'єкт, що містить усю послідовність кроків передоброблення. Важливо, що цей об'єкт "навчається" (наприклад, обчислює середні та стандартні відхилення для стандартизації) лише на тренувальних даних, а потім застосовується як до тренувальної, так і до тестової вибірки, що запобігає витоку даних.

3. Виконання навчання. Програмний комплекс послідовно виконує етапи навчання, визначені в DSL. Якщо вказано секцію cluster_homogenization, спочатку виконується кластеризація тренувальних даних, після чого для кожного отриманого кластера запускається окремий процес навчання спеціалізованої моделі. Якщо визначено recirculation, ітеративно будують шари ансамблю. На кожному етапі, що потребує навчання, викликають синтезатор моделей, який, за потреби, запускає процес ГПО з відповідною кількістю ітерацій.

4. Генерація результатів та метаданих. Після завершення навчання система збирає всі результати: навчені моделі, серіалізовані об'єкти передоброблення, детальні звіти про метрики якості на всіх етапах, оптимальні гіперпараметри, знайдені в ході ГПО, та іншу метаінформацію, яка буде запакована у фінальний відтворюваний пакет.

Програмний комплекс спроєктовано як набір модулів, де кожен етап – від читання DSL до навчання моделей та пакування результатів – реалізовано як окремий компонент. Такий модульний підхід істотно спрощує підтримку та подальше розширення системи новими алгоритмами чи функціональними можливостями. Наприклад, модуль для валідації DSL може бути використаний окремо для швидкої перевірки коректності конфігурацій без запуску повного, ресурсоємного процесу навчання, що є корисним на етапі проєктування. Аналогічно, модуль пакування артефактів може бути інтегрований в інші системи для стандартизації збереження результатів.

Для наочної демонстрації цього процесу на рисунку наведено концептуальну діаграму діяльності. Вона ілюструє логічну послідовність ключових фаз: від початкової валідації декларативної конфігурації, через етапи підготовки даних та ітеративного навчання модельних структур, до фінального етапу формування самодостатнього, відтворюваного пакета з усіма результатами.



Концептуальна діаграма діяльності процесу тренування / Conceptual activity diagram of training flow

Формування відтворюваного пакета. Ключовим результатом роботи програмного комплексу є створення відтворюваного пакета (bundle) – самодостатнього архіву, що містить усю необхідну інформацію для повного відтворення, аудиту або розгортання навченої модельної структури. Цей підхід забезпечує довготривалу цінність виконаної роботи та є основою для надійного розгортання агентів. Пакет містить такі компоненти:

- **resolved_config.json.** Фінальна, валідована та доповнена конфігурація DSL, що точно відображає всі параметри, використані під час навчання, ураховуючи знайдені в ході ГПО. Це єдине "джерело правди" про те, як було створено модельну структуру.
- **artifacts.json.** Мапа артефактів, що містить відносні шляхи до всіх збережених файлів (навчених моделей, об'єктів для передоброблення, маніфестів для cluster_homogenization тощо) всередині пакета.
- **Серіалізовані моделі та об'єкти передобробки.** Фактичні файли навчених моделей (наприклад, у форматі .joblib), що дає змогу завантажити їх без необхідності повторного навчання.
- **metrics.json.** Структурований файл з усіма метриками якості, розрахованими на тренувальній та тестовій вибірках, що дає змогу швидко оцінити ефективність моделі.
- **provenance.json.** Дані про походження, що містять версії ключових бібліотек, початкові значення (random_state), хеші конфігураційних файлів та іншу метаінформацію для забезпечення максимальної прозорості.
- **integrity.json.** Файл із контрольними сумами (SHA256) усіх файлів у пакеті, що дає змогу перевірити їхню цілісність та незмінність, гарантуючи, що пакет не було модифіковано після створення.
- **dag.json (опціонально).** Декларативний опис навченої модельної структури у вигляді спрямованого ациклічного графу (DAG), що готовий для виконання у спеціалізованому середовищі

Така структура пакета гарантує, що будь-який дослідник або розробник може не лише перевірити отримані результати, а й легко розгорнути навчену модельну структуру в іншому середовищі.

Результати демонстрації та валідації підходу. Для демонстрації гнучкості та практичної цінності розробленої DSL було здійснено серію тематичних досліджень (case studies) на наборі даних про вартість поїздок на таксі. Метою цих досліджень було не досягнення максимальної точності, а демонстрація того, як легко DSL дає змогу конфігурувати, відтворювати та систематично досліджувати вплив різних архітектурних рішень на остаточний результат.

Сценарії дослідження та результати. У межах дослідження протестовано дев'ять різних конфігурацій, кожну із яких описано за допомогою окремого DSL-файлу. Ці експерименти дали змогу оцінити вплив таких

аспектів, як глибина рециркуляції, кількість моделей на кожному шарі, стратегії агрегації, використання підвищення однорідності даних та застосування гіперпараметричної оптимізації (ГПО). Результати для кожної конфігурації подано в таблиці, де містяться значення кореня із середньоквадратичної похибки (RMSE) для різних конфігурацій модельної структури

Порівняння ефективності різних архітектур, налаштованих за допомогою DSL / Comparison of the performance of different architectures configured using DSL

№	Опис архітектури (через DSL)	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
3	Рециркуляція (2 шари)	recirculation: {layers: 2}	10.62
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

Інтерпретація результатів. Як видно з таблиці, використання DSL дало змогу не лише легко налаштувати та виконати широкий спектр архітектур, але й виконати їхній детальний порівняльний аналіз. Дані дають підставу зробити кілька важливих спостережень, що демонструють цінність DSL як інструменту для дослідження архітектурних рішень:

- **Дослідження глибини рециркуляції (порівняння 1, 3, 4).** DSL дала змогу легко змінювати кількість шарів рециркуляції. Результати свідчать, що просте збільшення глибини не завжди є оптимальним, а конфігурація із сімома шарами, але лише однією моделлю на кожному, виявилася ефективною.
- **Дослідження гнучкості конфігурації шарів (порівняння 4, 5, 6).** Завдяки детальній конфігурації в секції layers було протестовано архітектури з різною кількістю моделей на кожному шарі та різними стратегіями агрегації (gating). Це демонструє здатність DSL описувати та тестувати складні, гетерогенні ансамблі.
- **Оцінка впливу ГПО (порівняння 1 та 2).** Експеримент з активованою ГПО, налаштований через секцію search, показав, що автоматична оптимізація не гарантує покращення результату, що підкреслює важливість DSL для швидкої перевірки таких практичних гіпотез.
- **Оцінка комбінованих архітектур (порівняння 7, 8, 9).** DSL дала змогу легко поєднати два складні підходи – підвищення однорідності (cluster_homogenization) та подальшу рециркуляцію. Експерименти показують, що саме такі комбіновані архітектури досягають найкращих результатів. Також, змінюючи лише один прапорець (include_cluster_id), вдалося оцінити вплив окремої метаознаки на кінцеву точність.

Підтвердження відтворюваності. Для валідації відтворюваності експеримент для однієї зі складних архітектур було виконано повторно з тим самим фіксованим random_state. Результати обох запусків збіглися з високою точністю (різниця метрик менше ніж 1e-9), що підтверджує повну детермінованість та надійність запропонованого підходу.

Обговорення отриманих результатів. Здійснене дослідження дає змогу оцінити практичні переваги запропонованого DSL-орієнтованого підходу для конструювання та аналізу складних модельних структур.

Інтерпретація результатів. Результати, отримані в ході демонстраційних запусків, свідчать, що розроблена DSL є гнучким інструментом для практичної реалізації та порівняння складних архітектур. Замість того, щоб писати велику кількість програмного коду, розробник може швидко описувати та тестувати різні конфігурації, змінюючи лише параметри в декларативному файлі. Це істотно спрощує дослідження впливу різних архітектурних рішень, таких як глибина рециркуляції, ширина шарів ансамблю або комбінація із методом підвищення однорідності даних. Це позиціонує запропонований підхід у проміжній ніші між універсальними MLOps платформами та AutoML-системами. На відміну від інструментів, як-от MLflow [5], які потребують написання коду для реалізації кастомної логіки, розроблена DSL дає змогу описувати складні ансамблі повністю декларативно. Порівняно з AutoML-системами [7], які часто працюють як “чорна скринька”, наша DSL залишає повний контроль над архітектурою в руках розробника, зосереджуючись на керованому проєктуванні та відтворюваності, а не на автоматичному пошуку.

Ключовою перевагою є забезпечення відтворюваності. Генерація самодостатнього пакета з усіма конфігураціями, моделями, метриками та даними про походження (provenance) гарантує, що результати можуть бути надійно відтворені та перевірені. Це вирішує одне із фундаментальних завдань під час роботи зі складними, багатокомпонентними системами, коли результат може залежати від великої кількості неявних налаштувань.

Наукова новизна. Наукова новизна цієї роботи полягає у тому, що:

1. Розроблено спеціалізовану DSL, яка формалізує та об’єднує в єдиній конфігурації проєктування складних, багатоетапних модельних структур, вико-

ристовуючи такі раніше досліджені методи, як багатошарова рециркуляція та підвищення однорідності даних.

2. Запропоновано підхід до гранулярного керування гіперпараметричною оптимізацією (ГПО), що дає змогу застосовувати її до окремих компонентів складної модельної структури (наприклад, до окремих шарів ансамблю), зберігаючи повну відтворюваність результатів за фіксованого початкового стану.

3. Сформовано метод генерації повністю самодостатнього, відтворюваного пакета, що містить усі необхідні артефакти та метадані для валідації та аудиту модельної структури, забезпечуючи її цілісність та портативність.

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

- **Істотно спрощує та прискорює** процес експериментування та налаштування складних ансамблів, допомагаючи розробникам швидше знаходити оптимальні конфігурації для конкретних задач.
- **Гарантує відтворюваність** результатів, що є фундаментальною вимогою для надійного порівняння різних підходів у наукових дослідженнях.
- **Створює надійний міст між дослідженням та розгортанням**, оскільки згенерований відтворюваний пакет може слугувати основою для створення портативних графів обчислень (DAG), готових до використання в реальних системах, зокрема, у вигляді внутрішніх моделей для надійних програмних агентів.

Висновки/ Conclusions

Метою цієї роботи було розроблення підходу, що дає змогу спростити проєктування, підвищити надійність та забезпечити повну відтворюваність процесу створення складних модельних структур для моніторингових програмних агентів. Для досягнення цієї мети запропоновано та детально описано декларативний, DSL-орієнтований підхід, центральним елементом якого є спеціалізована предметно-орієнтована мова.

Розроблена DSL дає змогу формалізувати та об'єднати в єдиній, зрозумілій конфігурації всі етапи конструювання: від підготовки даних та передоброблення до опису складних, багатоетапних ансамблевих архітектур, таких як багатошарова рециркуляція та підвищення однорідності даних. На основі DSL реалізовано програмний комплекс, що автоматизує процес навчання та забезпечує детермінізм результатів за фіксованого початкового стану, що підтверджено експериментально.

Ключовим результатом роботи є створення методу генерації самодостатніх, відтворюваних пакетів. Ці пакети містять усі необхідні для розгортання та аудиту компоненти: навчені моделі, повні конфігурації, метрики якості та дані про походження. Це забезпечує повну прозорість, цілісність та портативність розроблених рішень. Продемонстровано, що такий підхід є гнучким інструментом для систематичного дослідження та порівняння різних архітектурних рішень.

Отже, запропонований DSL-орієнтований підхід успішно вирішує поставлені завдання. Він надає розробникам потужний інструментарій для роботи зі складними модельними структурами, істотно спрощуючи їх проєктування та гарантуючи відтворюваність. Це, своєю чергою, створює надійну основу для подальших досліджень та є важливим кроком на шляху від теоретичного аналізу складних моделей до їх практичного застосування у вигляді ефективних та надійних програмних агентів у складі мультиагентних систем.

References

1. Wooldridge, M. (2009). An Introduction to MultiAgent Systems. John Wiley & Sons..
2. Голуб, С. (2007). Багаторівневе моделювання в технологіях моніторингу оточуючого середовища. Черкаси: Вид. Від. ЧНУ Імені Богдана Хмельницького, 220.
3. Holub, S. V., Ostapiuk, V. V.(2025). Machine learning of multilayer models of a monitoring software agent. *Mathematical Machines and Systems,* 2, 76–96. https://doi.org/10.34121/1028-9763-2025-2-76-95.
4. Hutson, M. (2018). Artificial intelligence faces reproducibility crisis. *Science*, 359(6377), 725–726. https://doi.org/10.1126/science.359.6377.725.
5. Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41, 39–45.
6. The Kubeflow Authors. (2025). Kubeflow Pipelines. https://github.com/kubeflow/pipelines
7. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015, December 7). Efficient and Robust Automated Machine Learning. *Neural Information Processing Systems*. https://www.semanticscholar.org/paper/Efficient-and-Robust-Automated-Machine-Learning-Feurer-Klein/775a4e375cc79b53b94e37fa3eedff481823e4a6
8. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning (No. arXiv:1605.08695). *arXiv.* https://doi.org/10.48550/ arXiv.1605.08695.
9. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32. https://papers.nips.cc/paper_files/paper/2019/hash/ bdbca288fee7f92f2bfa9f7012727740-Abstract.html
10. Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1.
11. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework (No. arXiv:1907.10902). *arXiv*. http://arxiv. org/abs/1907.10902

S. V. Holub, V. V. Ostapiuk
Cherkasy State Technological University, Cherkasy, Ukraine

A DECLARATIVE APPROACH TO THE DESIGN AND REPRODUCIBLE LEARNING OF COMPLEX MODEL STRUCTURES FOR MONITORING SOFTWARE AGENTS

The development of effective monitoring software agents, essential components of modern multi-agent systems (MAS), increasingly relies on sophisticated model structures such as multi-layer machine learning ensembles. However, the growing complexity of these architectures presents significant challenges in ensuring the reliability, auditability, and, most critically, the reproducibility of experimental results. Addressing this challenge, this paper proposes a declarative approach centered on a newly developed domain-specific language (DSL). As a research method, this language provides a structured, human-readable format for describing the entire model construction process. The DSL specification covers not only data preparation and hyperparameter optimization but also the intricate configuration of multi-layer ensembles, including advanced mechanisms like recirculation and data homogeneity improvement through clustering. A software system was developed to interpret this DSL, thereby automating the complex training process. A key feature of this process is the automatic generation of a self-contained, reproducible bundle. This bundle includes not only the serialized models and preprocessing steps but also all associated configurations, performance metrics, and detailed provenance data, ensuring no implicit dependencies remain. The main results demonstrate that this declarative approach effectively tackles the complexity of managing advanced experiments, ensures the integrity of the created models, and guarantees their full reproducibility. It was also found that formalizing the experimental setup in a DSL provides a robust and objective framework for comparing different, often heterogeneous, model architectures. In conclusion, the proposed DSL-oriented approach creates a reliable and auditable foundation for developing and validating effective software agents. This work bridges the critical gap between algorithmic research and the practical need for trustworthy and deployable machine learning systems in real-world applications.

Keywords: multi-agent systems, agent-oriented monitoring, domain-specific language (DSL), model reproducibility, model synthesis algorithm, multi-layer model structure.

Інформація про авторів:

Голуб Сергій Васильович, д-р техн. наук, професор, завідувач кафедри програмного забезпечення автоматизованих систем.
Email: s.holub@chdtu.edu.ua; https://orcid.org/0000-0002-5523-6120

Остапюк Володимир Вікторович, аспірант, кафедра програмного забезпечення автоматизованих систем.
Email: v.v.ostapiuk.asp22@chdtu.edu.ua; https://orcid.org/0000-0002-5523-6120

Цитування за ДСТУ: Голуб С. В., Остапюк В. В. Декларативний підхід до проєктування та відтворюваного навчання складних модельних структур для моніторингових програмних агентів. *Український журнал інформаційних технологій.* 2025, т. 7, № 2. С. 01–08.

Citation APA: Holub, S. V., & Ostapiuk, V. V. (2025). A declarative approach to the design and reproducible learning of complex model structures for monitoring software agents. *Ukrainian Journal of Information Technology, 7*(2), 01–08. https://doi.org/10.23939/ujit2025.02.001