Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Застосування графових нейронних мереж
для розв'язання задач регресійного аналізу
(тема)

Виконав:
здобувач _2_ року навчання, групи _ПМм-24-1 _
Микита АВСІТІДІЙСЬКИЙ
(Власне ім'я, ПРІЗВИЩЕ)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Надія ГИБКІНА
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту

Завідувач кафедри ПМ _____ Максим СИДОРОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
(підпис)
"_10_" _листопада_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Авсітідійському Микиті Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи _Застосування графових нейронних мереж для розв'язання задач регресійного аналізу
затверджена наказом по університету від 10 листопада 2025 р. № 1028 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 18 грудня 2025 р.

3. Вихідні дані до роботи _математична модель регресії та набори даних

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв'язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	10 – 16 листопада 2025 р.	виконано
2	Вибір та обґрунтування методу	17 – 23 листопада 2025 р.	виконано
3	Розробка алгоритму і програми	24 – 30 листопада 2025 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	01 – 07 грудня 2025 р.	виконано
5	Робота над текстом пояснювальної записки	08 – 17 грудня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	18 грудня 2025 р.	виконано

Дата видачі завдання _10_ листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Надія ГИБКІНА
(підпис) (посада, Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка: 78 с., 1 табл., 6 рис., 1 дод., 30 джерел.

ГЛИБОКЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, ГРАФИ, ГРАФОВІ НЕЙРОННІ МЕРЕЖІ, РЕГРЕСІЙНИЙ АНАЛІЗ, ГРАФОВЕ ПОДАННЯ ДАНИХ.

Об’єкт дослідження – задача регресійного аналізу просторово-табличних даних.

Мета роботи – дослідження ефективності застосування графових нейронних мереж для розв’язання задач регресійного аналізу з урахуванням структурних зв’язків між об’єктами даних.

Методи дослідження – графові нейронні мережі з механізмом уваги, методи побудови графових моделей даних, методи регресійного аналізу.

Кваліфікаційна робота присвячена дослідженню можливостей використання графових нейронних мереж для підвищення точності регресійного аналізу. У роботі розглянуто підхід до представлення табличних і просторових даних у вигляді графа, де вершини відповідають об’єктам, а ребра відображають їх просторову близькість. На основі такого подання реалізовано модель графової нейронної мережі з механізмом уваги для прогнозування неперервної цільової змінної.

ABSTRACT

Introductory note: 78 pages, 1 tables, 6 figures, 1 appendixes, 30 sources.

DEEP LEARNING, MACHINE LEARNING, GRAPHS, GRAPH NEURAL NETWORKS, REGRESSION ANALYSIS, GRAPH DATA REPRESENTATION.

Object of research – the problem of regression analysis of spatial and tabular data.

Purpose of work – to investigate the effectiveness of graph neural networks for solving regression analysis problems taking into account structural relationships between data objects.

Methods of research – graph neural networks with attention mechanism, graph data modeling methods, regression analysis methods.

The qualification work is devoted to the study of the application of graph neural networks to improve the accuracy of regression analysis. The paper considers an approach to representing tabular and spatial data in the form of a graph, where vertices correspond to data objects and edges reflect their spatial proximity. Based on this representation, a graph neural network model with an attention mechanism was implemented to predict a continuous target variable.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку інформаційних технологій та зростання обсягів даних дедалі більшої актуальності набувають задачі їх аналізу та прогнозування. Значна частина даних, що виникають у соціально-економічних, інженерних, транспортних та природничих системах, має складну структуру та містить приховані взаємозв'язки між об'єктами. Однією з ключових задач аналізу таких даних є регресійний аналіз, який спрямований на прогнозування неперервних числових величин на основі наявних ознак.

Традиційні методи регресійного аналізу, зокрема лінійна регресія, багатошарові персептрони та ансамблеві алгоритми, широко застосовуються на практиці та добре досліджені. Проте ці підходи зазвичай розглядають об'єкти як незалежні між собою та не враховують структурних зв'язків, що можуть існувати між ними.

Світові тенденції розвитку машинного та глибинного навчання свідчать про зростання інтересу до методів, здатних безпосередньо працювати зі структурованими даними. Одним із перспективних напрямів є графові нейронні мережі, які поєднують можливості класичних нейронних мереж із формальним описом даних у вигляді графа. Провідні наукові колективи та дослідники активно застосовують такі підходи для аналізу соціальних мереж, молекулярних структур, транспортних систем, геопросторових даних та інших складних об'єктів. Особливе місце серед цих методів займають графові нейронні мережі з механізмом уваги, які дозволяють адаптивно враховувати вплив сусідніх об'єктів на результат прогнозування.

Актуальність даної роботи зумовлена необхідністю підвищення точності регресійного аналізу в задачах, де дані мають як табличну, так і просторову складову. Зокрема, у задачах прогнозування вартості об'єктів нерухомості, економічних показників або соціально-демографічних характеристик суттєвий вплив на результат мають не лише індивідуальні ознаки об'єкта, але й його оточення. Застосування графових нейронних мереж дозволяє формалізувати ці взає-

мозв’язки та враховувати їх у процесі навчання моделі, що робить дослідження даної теми актуальним як з наукової, так і з практичної точки зору.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є дослідження ефективності застосування графових нейронних мереж для розв’язання задач регресійного аналізу з урахуванням структурних зв’язків між об’єктами даних. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі регресійного аналізу з використанням методів машинного навчання;

– проаналізувати принципи роботи графових нейронних мереж і механізму уваги;

– реалізувати графову нейронну модель для задачі регресійного аналізу;

– провести обчислювальний експеримент та порівняти результати розв’язання задачі регресії за допомогою графової моделі з результатами класичних моделей машинного навчання.

Об’єктом дослідження є задача регресійного аналізу просторово-табличних даних.

Предметом дослідження є методи регресійного аналізу на основі графових нейронних мереж з урахуванням структурних зв’язків між об’єктами даних.

Методи дослідження. У кваліфікаційній роботі використовуються графові нейронні мережі з механізмом уваги, методи побудови графових моделей даних, методи регресійного аналізу.

Публікації. Результати, отримані у кваліфікаційній роботі, було представлено на 4-й Міжнародній науково-практичній конференції англійською мовою «Європейські студії. Навчання і викладання у світі технологій» (м. Клуж-Напока, Румунія 12 листопада 2025 р.) [1], на міжнародному конкурсі студентських наукових робіт зі штучного інтелекту (м. Київ травень-листопад 2025 р.) [2] та на Всеукраїнській науково-практичній конференції здобувачів вищої освіти і молодих учених «Комп’ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» (м. Харків, 25 листопада 2025 р.) [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Регресійний аналіз як задача машинного навчання

1.1.1 Постановка задачі регресійного аналізу та методи її розв'язання

Регресійний аналіз є одним із базових напрямів машинного навчання та математичної статистики, метою якого є встановлення функціональної залежності між набором вхідних ознак об'єкта та неперервною цільовою змінною. На відміну від задач класифікації, де результатом є віднесення об'єкта до одного з наперед визначених класів, у задачах регресії необхідно передбачити кількісне значення, що може бути довільним дійсним числом у певному діапазоні.

У загальному випадку задача регресійного аналізу формулюється наступним чином. Нехай задано вибірку спостережень $\{(x_i, y_i)\}_{i=1}^{N}$, де $x_i \in \mathbb{R}^d$ – вектор ознак i-го об'єкта, а $y_i \in \mathbb{R}$ – відповідне значення цільової змінної. Потрібно побудувати модель у вигляді функції $f : \mathbb{R}^d \to \mathbb{R}$, яка для об'єктів вибірки, а також нових, раніше невідомих об'єктів якісно прогнозує значення цільової змінної [12].

Класичним підходом до розв'язання задачі регресійного аналізу є метод найменших квадратів (МНК). У випадку лінійної регресії припускається, що залежність між вхідними ознаками та цільовою змінною має вигляд:

$$\hat{y} = x^T \cdot w + b,$$

де $x \in \mathbb{R}^d$ – вектор ознак об'єкта;

w – вектор параметрів моделі;

b – вільний член.

Для вибірки з n спостережень модель можна записати у матричній формі:

$$\hat{y} = X \cdot w,$$

де $X \in \mathbb{R}^{n \times d}$ – матриця ознак;

$\hat{y} \in \mathbb{R}^n$ – вектор прогнозованих значень.

Метод найменших квадратів полягає у мінімізації сумарної квадратичної похибки між фактичними та прогнозованими значеннями цільової змінної:

$$L(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \| y - X \cdot w \|_2^2 .$$

Розв'язок цієї задачі за умови повного рангу матриці X має аналітичну форму:

$$w = (X^T \cdot X)^{-1} X^T \cdot y .$$

Такий підхід забезпечує оптимальне в сенсі МНК наближення даних лінійною моделлю, однак вимагає обчислення оберненої матриці, що може бути обчислювально складним для великих наборів даних.

У рамках машинного навчання для знаходження параметрів регресійної моделі часто застосовується градієнтний спуск, який дозволяє мінімізувати функцію втрат ітеративно. Для середньоквадратичної похибки оновлення параметрів на k-й ітерації має вигляд:

$$w^{(k+1)} = w^{(k)} - \eta \nabla_w L(w^{(k)}),$$

де $\eta > 0$ – крок навчання.

Градієнтний спуск є універсальним методом оптимізації та використовується не лише для лінійної регресії, а й для навчання складних нелінійних моделей, зокрема нейронних мереж.

Якість побудованої регресійної моделі зазвичай оцінюється за допомогою метрик, що вимірюють відхилення між прогнозованими та фактичними значеннями. Найпоширенішими метриками у задачах регресійного аналізу є середньоквадратична похибка та середня абсолютна похибка. Вибір конкретної метрики залежить від властивостей даних і вимог до моделі, зокрема чутливості до викидів та інтерпретованості результатів.

У рамках машинного навчання регресійний аналіз реалізується за допомогою різноманітних алгоритмів. До класичних методів належать метод найменших квадратів, а також регуляризовані підходи. Більш складні залежності можуть бути змодельовані за допомогою нелінійних методів, таких як багатошарові персептрони або ансамблеві алгоритми, зокрема випадковий ліс. Ці методи здатні апроксимувати складні функціональні залежності, однак зазвичай розглядають об’єкти як незалежні один від одного.

У багатьох прикладних задачах припущення про незалежність об’єктів не виконується. Наприклад, у задачах прогнозування цін на нерухомість, аналізу транспортних потоків або соціально-економічних показників, значення цільової змінної для окремого об’єкта може суттєво залежати від характеристик сусідніх або пов’язаних об’єктів. У таких випадках табличне подання даних не дозволяє повною мірою врахувати взаємозв’язки між спостереженнями. Саме ця обставина зумовлює необхідність розширення класичних підходів до регресійного аналізу та використання моделей, здатних працювати зі структурованими даними.

Подання даних у вигляді графа дає можливість явно зафіксувати відношення між об’єктами, де вершини відповідають окремим спостереженням, а ребра – наявним між ними зв’язкам. У такій постановці задача регресійного аналізу полягає не лише у використанні власних ознак об’єкта, але й у врахуванні інформації з його локального оточення.

1.1.2 Огляд літератури за темою досліджень

Задача регресійного аналізу є однією з базових у машинному навчанні та статистиці і традиційно розв’язується за допомогою класичних методів, що ґрунтуються на табличному поданні даних. До таких підходів належать лінійна та поліноміальна регресія, метод найменших квадратів, а також різні варіанти регуляризованих моделей. Узагальнений огляд класичних і сучасних методів машинного навчання для регресійних задач наведено у роботах [12, 13, 15–18], де детально розглядаються як статистичні основи, так і практичні аспекти застосування моделей.

Подальший розвиток методів регресійного аналізу пов’язаний із використанням глибоких нейронних мереж, зокрема багатошарових персептронів, які здатні апроксимувати складні нелінійні залежності між ознаками та цільовою змінною [9, 10, 14]. Однак такі моделі, як правило, розглядають об’єкти незалежно один від одного та не враховують можливі структурні взаємозв’язки між спостереженнями, що обмежує їх ефективність у задачах із просторовою або мережевою природою даних. Для подолання зазначених обмежень були запропоновані методи, орієнтовані на обробку структурованих даних, зокрема графові нейронні мережі. Ранні підходи до нейронної обробки структур та графів закладені у роботах [19, 21], де вперше сформульовано ідею навчання на графових доменах. Подальший розвиток цих ідей привів до появи спектральних і просторових графових згорткових моделей [25], а також узагальнених архітектур графових нейронних мереж, огляд яких наведено у роботі [4].

Сучасні дослідження демонструють ефективність графових нейронних мереж у широкому спектрі задач, включно з регресійним аналізом. Зокрема, GNN застосовуються для прогнозування фізичних полів і параметрів інженерних систем [5, 7, 8], аналізу часових рядів та багатовимірних залежностей [27], а також для роботи з табличними даними, доповненими графовими зв’язками [29]. Окрему увагу приділено теоретичним властивостям графових моделей, зокрема їх виразній здатності [20] та інтерпретації механізмів згортки на графах

[21]. Важливим напрямом розвитку графових нейронних мереж є використання механізмів уваги, що дозволяють адаптивно зважувати внесок сусідніх вузлів у процес формування представлень. Архітектура Graph Attention Networks запропонована у роботі [6] і надалі активно застосовується у задачах прогнозування неперервних величин, де вплив локального оточення є нерівномірним. Подальші дослідження розширюють можливості GNN для регресійних задач за рахунок методів пояснюваності [28] та аналізу динамічних графів [30].

1.2 Просторові та графові структури даних у прикладних задачах

У багатьох прикладних задачах аналізу даних об'єкти дослідження не є незалежними між собою, а пов'язані просторовими, функціональними або логічними відношеннями. Такі дані називають структурованими, оскільки між окремими об'єктами існують залежності, що не можуть бути повністю описані лише індивідуальними ознаками кожного об'єкта [5].

Нехай задано скінченну множину об'єктів $V = \{v_1, v_1, ..., v_N\}$, де кожному об'єкту v_i відповідає вектор ознак $x_i \in \mathbb{R}^d$. У випадку просторових даних кожному об'єкту додатково відповідає вектор координат $\mathbf{p}_i = (p_i^{(1)}, p_i^{(2)}) \in \mathbb{R}^2$, який визначає його положення у просторі.

Зв'язки між об'єктами можуть визначатися на основі просторової близькості. Для цього вводиться функція відстані $d_{ij} = dist(\mathbf{p}_i, \mathbf{p}_j)$, яка задовольняє властивості метрики. У практичних задачах для евклідового простору використовується евклідова відстань, а для географічних координат – сферичні метрики.

На основі відстаней формується локальне оточення кожного об'єкта за правилом k найближчих сусідів.

Нехай $d_{i\pi_i(1)} \le d_{i\pi_i(2)} \le ... \le d_{i\pi_i(N-1)}$ – впорядкування відстаней від об'єкта v_i до всіх інших об'єктів. Тоді множина k найближчих сусідів визначається як

$$N_k(v_i) = \{v_{\pi_i(1)}, v_{\pi_i(2)}, ..., v_{\pi_i(k)}\}.$$

Для формалізації взаємозв'язків між об'єктами використовується графове подання даних. Дані задаються у вигляді графа $G = (\nu, \varepsilon)$, де ν – множина вершин, а $\varepsilon \subseteq \nu \times \nu$ – множина ребер. Ребро (v_i, v_j) належить множині ε, якщо вершина v_j входить до множини сусідів $N_k(v_i)$.

У багатьох прикладних задачах доцільно використовувати зважені графи. Для цього вводиться функція ваг ребер $w_{ij} = \phi(d_{ij})$, де $\phi(d_{ij})$ – спадна функція відстані, яка відображає ступінь впливу об'єкта v_j на об'єкт v_i. Таким чином, кожному ребру графа відповідає числова міра зв'язку між вершинами.

Графове подання дозволяє об'єднати інформацію про індивідуальні ознаки об'єктів та їх взаємозв'язки.

Формально дані описуються трійкою (G, X, W), де $X \in \mathbb{R}^{N \times d}$ – матриця ознак вершин, а W – вектор ваг ребер.

У задачах регресійного аналізу цільова змінна $y_i \in \mathbb{R}$ може залежати не лише від вектору ознак x_i, але й від ознак сусідніх вершин та структури графа. У загальному вигляді це можна записати як

$$y_i = f(x_i, \{x_j : v_j \in N_k(v_i)\}) + \varepsilon_i,$$

де ε_i – випадкова похибка.

Таким чином, просторові та графові структури даних забезпечують формально обґрунтовану математичну основу для моделювання залежностей між об'єктами у прикладних задачах. Використання графового подання створює передумови для застосування сучасних методів машинного навчання, зокрема графових нейронних мереж, які дозволяють ефективно враховувати як індивідуальні характеристики об'єктів, так і вплив їх локального оточення.

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Регресійний аналіз є інструментом для виявлення кількісних закономірностей у даних та прогнозування значень цільової змінної на основі відомих характеристик об’єктів. Такий підхід дозволяє будувати моделі, що відображають залежність між вхідними ознаками та кількісним результатом. У випадку задачі прогнозування вартості житла досліджувані дані містять як числові, так і категоріальні характеристики об’єктів: географічні координати, вік будівель, кількість кімнат, чисельність населення в районі, рівень доходу домогосподарств тощо. Метою аналізу є встановлення залежностей між цими ознаками та ціною житла, що дозволяє не лише пояснити існуючі дані, але й передбачати вартість нових об’єктів.

Основною особливістю цієї задачі є те, що значення цільової змінної – ринкової вартості житла – є неперервною величиною, яка може істотно змінюватися залежно від поєднання численних факторів. До таких факторів належать як індивідуальні характеристики об’єкта (площа, кількість кімнат, технічний стан), так і контекстні ознаки середовища, у якому він розташований (щільність забудови, наявність інфраструктури, транспортна доступність). Важливим завданням регресійного аналізу є врахування впливу як індивідуальних характеристик об’єкта, так і контекстних ознак середовища, у якому він розташований.

Задача регресії полягає у тому, щоб знайти таку функціональну залежність між ознаками об’єкта та його вартістю, яка б якнайкраще наближала наявні дані. Таким чином, побудована модель дозволяє здійснювати прогнозування цін на основі відомих факторів, оцінювати внесок окремих характеристик у формування кінцевої вартості та застосовувати отримані результати для практичних цілей, зокрема в економіці, соціології та міському плануванні.

1.3.2 Формальна постановка задачі

Нехай задано вибірку даних $D = \{(x_i, y_i)\}_{i=1}^{N}$, де кожен об'єкт описується вектором ознак $x_i = (x_{i1}, x_{i2}, ..., x_{im}) \in \mathbb{R}^m$, а цільова змінна $y_i \in \mathbb{R}$ є числовою неперервною величиною, що у даному випадку означає вартість житла.

Мета регресійного аналізу полягає у відновленні невідомої залежності між вхідними ознаками та вихідною змінною. Для цього необхідно знайти таку функцію $f : \mathbb{R}^m \to \mathbb{R}$, яка для кожного об'єкта x_i повертає передбачене значення $\hat{y}_i = f(x_i)$, наближене до справжнього значення y_i.

Якість апроксимації визначається функціоналом похибки:

$$L(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)),$$

де $\ell(y_i, f(x_i))$ – обрана міра відстані між реальним значенням та прогнозом.

У задачах регресії використовують середньоквадратичну похибку (MSE):

$$\ell(y_i, f(x_i)) = (y_i - f(x_i))^2$$

або середню абсолютну похибку (MAE):

$$\ell(y_i, f(x_i)) = |y_i - f(x_i)|.$$

Отже, задача регресії зводиться до знаходження такої функції f^*, що мінімізує функцію втрат:

$$f^* = \arg\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)),$$

де F – простір можливих моделей.

1.4 Постановка задач дослідження

Виходячи з проведеного аналізу сучасних підходів до регресійного аналізу просторово-табличних даних, можна констатувати, що побудова моделей прогнозування вартості житла значно ускладнюється через наявність прихованих структурних зв’язків між об’єктами. Традиційні методи, засновані лише на табличних ознаках, не враховують просторову взаємодію між районами, що призводить до втрати вагомої частини інформації про їх контекст. Застосування графових нейронних мереж, зокрема Graph Attention Networks, дає можливість додатково враховувати топологію даних і динамічно визначати важливість сусідів при формуванні прогнозу.

Метою роботи є дослідження ефективності застосування графових нейронних мереж для розв’язання задач регресійного аналізу з урахуванням структурних зв’язків між об’єктами даних. У даній роботі розглядається використання розглянутого підходу до прогнозування вартості житла на основі даних з набору California Housing Prices [11].

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі регресійного аналізу з використанням методів машинного навчання;

– проаналізувати принципи роботи графових нейронних мереж і механізму уваги;

– реалізувати графову нейронну модель для задачі регресійного аналізу;

– провести обчислювальний експеримент та порівняти результати розв’язання задачі регресії за допомогою графової моделі з результатами класичних моделей машинного навчання.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Розв'язання задачі регресії багатошаровим персептроном

Для розв'язання задачі регресії методами глибокого навчання як базову модель часто використовують багатошаровий персептрон (MLP) – нейронну мережу прямого поширення, яка отримує на вході табличний вектор ознак об'єкта x та повертає прогноз неперервної цільової змінної $\hat{y}$. На відміну від графових моделей, MLP не використовує інформацію про зв'язки між об'єктами й працює з кожним спостереженням незалежно.

Навчання персептрона зводиться до підбору параметрів моделі шляхом мінімізації функції втрат. Для задач регресії найчастіше застосовують середньоквадратичну похибку:

$$L = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 .$$

Оптимізація параметрів виконується ітеративно за градієнтним спуском (або його модифікаціями, наприклад Adam):

$$\theta \leftarrow \theta - \eta \nabla_\theta L .$$

де η – крок навчання;

θ – параметри моделі.

Мережа MLP отримує попередньо оброблені табличні ознаки (масштабовані числові та one-hot кодовані категоріальні) і прогнозує значення цільової змінної.

Основним недоліком багатошарового персептрона у задачах зі структурованими даними є припущення про незалежність спостережень, унаслідок чого модель не враховує просторові, мережеві або ієрархічні зв'язки між об'єктами;

інформація про локальне оточення кожного об’єкта або повністю ігнорується, або може бути врахована лише опосередковано через ручне перетворення ознак, що призводить до втрати структурної інформації, зниження узагальнювальної здатності моделі та обмежує її ефективність у задачах, де значення цільової змінної суттєво залежить від характеристик пов’язаних об’єктів.

2.2 Основні відомості з теорії графових нейронних мереж

У сучасних задачах аналізу даних усе частіше виникає необхідність працювати зі структурами, які не можуть бути коректно представлені у вигляді регулярних решіток або лінійних послідовностей. До таких структур належать соціальні мережі, хімічні молекули, транспортні та логістичні системи, тривимірні геометричні моделі, а також карти знань. Спільною рисою подібних об’єктів є наявність складних взаємозв’язків між їх елементами, що природно описуються за допомогою графів, які складаються з вершин (об’єктів) і ребер (зв’язків між ними).

Графові нейронні мережі (Graph Neural Networks, GNN) були запропоновані як клас моделей машинного навчання, здатних безпосередньо працювати з графовими структурами даних. На відміну від традиційних підходів, які або ігнорують структуру зв’язків між об’єктами, або вимагають попереднього перетворення графа у табличний формат, GNN дозволяють одночасно враховувати як локальні ознаки вершин, так і топологію графа. Завдяки цьому такі моделі є більш придатними для аналізу структурованих даних.

Ключовою ідеєю графових нейронних мереж є поетапне оновлення представлень вершин на основі інформації, отриманої від їхніх сусідів. Даний процес реалізується за допомогою механізму передачі повідомлень (message passing). На кожному шарі нейронної мережі вершина агрегує ознаки сусідніх вершин відповідно до заданого правила та використовує їх для оновлення власного стану:

$$\vec{h}_i^{(k)} = \text{UPDATE}(\vec{h}_i^{(k-1)},\ \text{AGGREGATE}\{\vec{h}_j^{(k-1)} : j \in N(i)\}),$$

де $\vec{h}_i^{(k)}$ – вектор ознак вузла i на k-му шарі;

$N(i)$ – множина його сусідів [4].

У результаті багаторазового застосування такого оновлення кожна вершина отримує векторне представлення, що поєднує інформацію про її власні характеристики та контекст локального оточення. Це робить графові нейронні мережі універсальним інструментом для розв'язання широкого спектра задач, зокрема класифікації вершин, класифікації графів, передбачення зв'язків між об'єктами та регресійного аналізу числових властивостей.

У науковій літературі зазвичай виділяють два основні підходи до побудови графових нейронних мереж [4]:

– обрати інструменти програмної розробки і програмно реалізувати обраний метод;

– вибрати математичну модель двомірної траєкторії руху частинок і програмно її реалізувати.

Незважаючи на відмінності у формалізації, обидва підходи базуються на спільному принципі: локальні ознаки вершин послідовно збагачуються інформацією з їх оточення, а відповідні перетворення повторюються шар за шаром, формуючи більш інформативні представлення.

Ефективність графових нейронних мереж підтверджується численними практичними дослідженнями. Зокрема, вони демонструють високу точність у задачах прогнозування властивостей молекул, що дозволяє суттєво прискорити процес пошуку нових лікарських засобів. У галузі інженерних застосувань GNN використовуються для аналізу складних технічних систем. Застосування графового подання дає змогу зменшити розмірність задачі без втрати структурної інформації та підвищити узагальнювальну здатність моделей при роботі з новими даними. Так, у роботі [5] було показано, що графові моделі можуть безпосередньо працювати з 3D-CAD геометрією компонентів автомобільних аку-

муляторів, замінюючи обчислювально дорогі фізичні симуляції прогнозами, отриманими за допомогою нейронних мереж [5].

2.3 Шар Graph Attention (GATConv)

Graph Attention Networks (GAT) були запропоновані у 2018 році як альтернативний підхід до спектральних методів побудови графових нейронних мереж. Їх ключовою особливістю є використання механізму самоуваги (self-attention), який дозволяє адаптивно визначати ступінь впливу сусідніх вузлів під час оновлення ознак кожної вершини графа [4].

Нехай маємо множину вхідних ознак вузлів $h = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}$, $\vec{h}_i \in \mathbb{R}^F$, де N – кількість вузлів, F – розмірність ознак. Спочатку застосовується спільне лінійне перетворення $W \in \mathbb{R}^{F \times F}$. Потім для кожної пари сусідніх вузлів i, j обчислюється коефіцієнт уваги:

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j),$$

де $a(W\vec{h}_i, W\vec{h}_j)$ – невелика нейромережа (у базовому випадку одновимірний персептрон із функцією активації LeakyReLU).

Значення нормалізуються через softmax по сусідах:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}.$$

Оновлення ознак вузла відбувається як зважена сума ознак його сусідів:

$$\vec{h}_i' = \sigma\left(\sum_{j \in N_i} a_{ij} W \vec{h}_j \right),$$

де σ – нелінійна функція активації, наприклад ELU або ReLU.

З метою підвищення стабільності навчання та покращення здатності моделі до узагальнення у GAT використовується багатоголова увага (multi-head attention). У цьому випадку кілька незалежних механізмів уваги працюють паралельно, а їх результати об'єднуються шляхом підсумовування або усереднення, що дозволяє моделі одночасно враховувати різні аспекти локальної структури графа [6].

2.4 Застосування графових нейронних мереж до задач регресійного аналізу

У класичній постановці задачі регресії необхідно знайти функцію $f : \mathbb{R}^m \to \mathbb{R}$, $\hat{y}_i = f(x_i)$, яка для кожного об'єкта x_i апроксимує значення цільової змінної y_i. Оптимальне відображення f^* визначається як таке, що мінімізує середнє значення функції втрат на навчальній вибірці:

$$f^* = \arg\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)),$$

де $\ell(y_i, f(x_i))$ – міра похибки.

У випадку графових даних об'єкт x_i описується не лише власними ознаками відповідної вершини, а й інформацією про її локальне оточення. Це відображає той факт, що значення цільової змінної для окремого вузла може залежати від характеристик сусідніх вузлів і загальної структури графа [6].

Механізм уваги у GAT дозволяє формально записати побудову нового представлення вузла i у вигляді:

$$\vec{h}_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W \vec{h}_j \right),$$

де $\vec{h}_j$ – вектор ознак сусіда j;

W – матриця перетворення;

σ – нелінійна функція активації;

α_{ij} – коефіцієнт уваги, що визначає вагу сусіда j для вузла i.

Коефіцієнти α_{ij} обчислюються як нормалізовані скалярні значення:

$$\alpha_{ij} = \frac{\exp(a^{\mathrm{T}}[W\vec{h}_i \,\|\, W\vec{h}_j])}{\sum_{k \in N(i)} \exp(a^{\mathrm{T}}[W\vec{h}_i' \,\|\, W\vec{h}_k])},$$

де $\|$ позначає конкатенацію векторів;

a – параметри уваги, що навчаються.

Таким чином, нове представлення вузла $\vec{h}_i'$ є адаптивною зваженою комбінацією ознак його сусідів. Ця комбінація будується динамічно і відображає внесок кожного сусіда у формування прогнозу [7].

У задачі регресійного аналізу остаточне передбачення для вузла i отримується шляхом застосування вихідного шару:

$$\vec{y}_i = f(\vec{h}_i'),$$

де $f(\vec{h}_i')$ – лінійна або нелінійна функція перетворення, що повертає скалярне значення цільової змінної.

Оптимізація параметрів W та a здійснюється шляхом мінімізації функції втрат:

$$L(\Theta) = \frac{1}{N}(y_i - \hat{y}_i)^2,$$

де $\Theta = \{W, a\}$ – множина всіх параметрів моделі.

Таким чином, шар уваги GAT розв'язує задачу регресії через побудову інформативних представлень вершин графа, у яких враховується як власна інформація вузла, так і релевантний вплив його сусідів. Це дозволяє відновити залежність f^* у просторі графових структур і забезпечує точні прогнози для нових даних [8].

Використання механізму уваги дозволяє уникнути жорсткого припущення про рівнозначність усіх сусідів вузла, яке притаманне класичним методам агрегації, таким як усереднення або підсумовування.

У випадку багатошарових GAT-архітектур нове представлення вузла на вищих рівнях агрегує інформацію від сусідів другого та вищих порядків, що дозволяє моделі враховувати ширший контекст графа без явного збільшення розмірності ознак.

Висновки за розділом 2

У даному розділі було розглянуто теоретичні засади побудови та застосування графових нейронних мереж як сучасного інструменту аналізу структурованих даних. Показано, що у багатьох прикладних задачах класичні методи машинного навчання є обмеженими, оскільки вони не враховують складні взаємозв'язки між об'єктами, які природно виникають у просторових, мережевих та ієрархічних структурах даних.

Було встановлено, що графові нейронні мережі дозволяють формалізувати такі взаємозв'язки шляхом представлення даних у вигляді графа, де вершини відповідають окремим об'єктам, а ребра – їхнім зв'язкам. Основний принцип роботи GNN полягає у поетапному оновленні представлень вершин за рахунок

агрегації інформації від сусідніх вузлів. Це забезпечує поєднання локальних ознак об’єкта з контекстом його оточення та дозволяє отримувати більш інформативні векторні представлення.

Окрему увагу було приділено механізму уваги, який лежить в основі архітектури Graph Attention Networks. Показано, що використання самоуваги дозволяє адаптивно зважувати внесок кожного сусіднього вузла, уникаючи жорсткого припущення про рівнозначність усіх зв’язків. Такий підхід підвищує гнучкість моделі та сприяє більш точному відображенню реальних залежностей у даних. Застосування багатоголової уваги додатково покращує стабільність навчання та дозволяє одночасно враховувати різні аспекти локальної структури графа.

Графові нейронні мережі з механізмом уваги забезпечують ефективне розв’язання задач прогнозування неперервних числових величин. На відміну від класичних регресійних моделей, GAT дозволяють враховувати як індивідуальні ознаки об’єктів, так і вплив їх локального оточення, що є особливо важливим у задачах з просторовою або мережевою природою даних. Це створює можливість відновлення складних залежностей між змінними у просторі графових структур.

Ефективність графових нейронних мереж підтверджується численними практичними дослідженнями у різних галузях, зокрема у хімії, біоінформатиці та інженерних застосуваннях. Використання графового подання дозволяє зменшити розмірність задачі без втрати структурної інформації та підвищити узагальнювальну здатність моделей при роботі з новими даними.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Використання мови програмування Python

У даній кваліфікаційній роботі для програмної реалізації методів машинного навчання та проведення обчислювальних експериментів було використано мову програмування Python. Ця мова програмування є однією з найпоширеніших мов у сфері аналізу даних і штучного інтелекту завдяки простому синтаксису, широкій екосистемі бібліотек та активній науковій спільноті. Це робить її зручним інструментом для реалізації як класичних методів машинного навчання, так і сучасних моделей глибинного навчання.

Реалізація програмної частини та експериментальна перевірка запропонованих моделей здійснювалися у хмарному середовищі Google Colab. Google Colab є інтерактивним сервісом для виконання Python-коду у форматі блокнотів, який надає доступ до попередньо налаштованого програмного середовища. Основною перевагою цього сервісу для даного дослідження є можливість безкоштовного використання апаратних прискорювачів, зокрема графічних процесорів (GPU), що істотно скорочує час навчання нейронних мереж. Крім того, середовище Google Colab дозволяє швидко встановлювати необхідні бібліотеки, зберігати результати експериментів та забезпечує відтворюваність обчислень.

У рамках роботи було реалізовано повний цикл машинного навчання: попередню обробку даних, побудову моделей, навчання, валідацію, тестування та аналіз отриманих результатів. Для забезпечення відтворюваності експериментів використовувалося фіксування початкових значень генераторів випадкових чисел, що є стандартною практикою у дослідженнях з машинного навчання.

Для роботи з табличними та числовими даними застосовувалися бібліотеки NumPy та pandas. NumPy використовувався для виконання векторизованих чисельних обчислень, роботи з масивами та матрицями, а також для реалізації допоміжних операцій під час підготовки даних. Бібліотека pandas застосовува-

лася для зчитування та первинного аналізу набору даних, обробки пропущених значень і формування вибірок для подальшого навчання моделей [9].

Попередня обробка даних виконувалася із застосуванням інструментів бібліотеки scikit-learn. Зокрема, використовувалися методи стандартизації числових ознак, кодування категоріальних змінних та імпутації пропущених значень. Для розбиття даних на навчальну, валідаційну та тестову вибірки застосовувався метод випадкового розділення, що дозволило коректно оцінити узагальнювальну здатність моделей. Окрему роль відіграли алгоритми пошуку найближчих сусідів, які використовувалися для побудови графової структури даних на основі просторової близькості об’єктів.

Для реалізації моделей глибинного навчання застосовувалася бібліотека PyTorch, яка надає гнучкі засоби для побудови нейронних мереж та автоматичного диференціювання. PyTorch дозволяє явно керувати процесом навчання, визначати архітектуру моделей та виконувати обчислення як на центральному процесорі, так і на графічному прискорювачі. У роботі PyTorch використовувався для реалізації як класичної багатошарової нейронної мережі, так і більш складних графових моделей.

Для роботи безпосередньо з графовими нейронними мережами застосовувалася спеціалізована бібліотека PyTorch Geometric. Дана бібліотека розширює можливості PyTorch та надає готові реалізації шарів графових нейронних мереж, зокрема Graph Attention Networks. За допомогою PyTorch Geometric було реалізовано графову модель регресії, у якій вершини відповідають об’єктам набору даних, а ребра формуються на основі просторової близькості між ними. Використання цієї бібліотеки дозволило ефективно працювати з графовими структурами, враховувати атрибути ребер і застосовувати механізм уваги [10].

Для порівняльного аналізу результатів у роботі також були реалізовані базові моделі машинного навчання, зокрема багатошарова нейронна мережа та ансамблевий метод випадкового лісу. Це дозволило оцінити переваги графових моделей у порівнянні з класичними підходами, що не враховують структурні зв’язки між об’єктами.

Аналіз процесу навчання та результатів експериментів здійснювався за допомогою бібліотеки Matplotlib, яка використовувалася для побудови графіків кривих навчання, а також для візуалізації помилок на різних етапах тренування моделей. Графічне представлення результатів дозволило наочно проілюструвати процес збіжності моделей та порівняти їх ефективність.

Таким чином, використання мови програмування Python разом із сучасними бібліотеками машинного навчання та хмарним середовищем Google Colab забезпечило гнучку, масштабовану та відтворювану платформу для реалізації графових нейронних мереж і проведення обчислювальних експериментів. Обраний інструментарій повністю відповідає сучасним вимогам до досліджень у галузі машинного навчання та дозволяє ефективно розв’язувати задачі регресійного аналізу структурованих даних.

3.2 Представлення даних у вигляді графа та алгоритм розв’язання поставленої задачі

Для експериментів було використано California Housing Dataset – класичний набір даних, який містить інформацію про житло в різних районах Каліфорнії [11]. Кожний запис описує окремий житловий район і включає як числові, так і категоріальні характеристики. До числових ознак належать:

– географічні координати (широта, довгота);

– медіанний вік будівель;

– кількість кімнат;

– кількість спальних місць;

– чисельність населення в районі;

– кількість домогосподарств;

– медіанний дохід домогосподарств.

Додатково враховувалася одна категоріальна змінна – близькість до океану (*ocean proximity*). Цільовою змінною виступала медіанна вартість житла у

відповідному районі, виражена у доларах США.

Таким чином, для кожного району маємо набір ознак $x_i \in \mathbb{R}^m$ та скалярну цільову змінну $y_i \in \mathbb{R}$.

Для побудови структури графа (визначення зв'язків між вершинами) використовувалися лише географічні координати, оскільки вони природно відображають просторову близькість районів. Проте вектор ознак кожної вершини містив повний набір доступних характеристик (усі числові ознаки після нормалізації та категоріальна ознака після one-hot кодування). Таким чином, всі ознаки були враховані, а графова структура накладалася на весь набір даних.

Щоб використати графові нейронні мережі, дані були представлені у вигляді графа $G = (V, E, X, W)$, де V – множина вершин, що відповідають районам, E – множина ребер, які задають просторові зв'язки між районами, $X \in \mathbb{R}^{N \times m}$ – матриця ознак вершин, а $W \in \mathbb{R}^{|E|}$ – вектор ваг ребер.

Відсутні значення у числових змінних замінювалися на медіану відповідного стовпця. Після цього дані нормалізувалися за стандартною схемою:

$$x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\sigma_j},$$

де $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$, $\sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_j)^2}$.

Категоріальна ознака «близькість до океану» була закодована методом one-hot encoding:

$$z \in \{1, ..., K\} \Rightarrow e_z = (0, ..., 0, 1, 0, ..., 0) \in \{0,1\}^K.$$

При формуванні ребер графа для кожної вершини i обиралися $k = 12$ найближчих сусідів за метрикою гаверсинуса, яка враховує кривизну земної поверхні. Відстань між вузлами i та j дорівнює:

$$d_{ij} = 2R \cdot \arcsin\left(\sqrt{\sin^2\frac{\varphi_i - \varphi_j}{2} + \cos(\varphi_i)\cdot\cos(\varphi_j)\cdot\sin^2\left(\frac{\lambda_i - \lambda_j}{2}\right)}\right),$$

де φ – широта;

λ – довгота;

$R = 6371$ км – радіус Землі.

Таким чином множина ребер визначається як:

$$E = \{(i, j) \mid j \in KNN(i,k)\},$$

де $j \in KNN(i,k)$ означає множину k найближчих сусідів вершини i, тобто тих вершин, які мають найменшу географічну відстань до неї. У нашому випадку для кожної вершини бралися $k = 12$ найближчих сусідів, і саме з ними вона з'єднувалася ребрами у графі.

Вага кожного ребра обчислювалася за допомогою гаусового ядра:

$$w_{ij} = \exp\left(-\frac{d_{ij}^{\,2}}{\sigma^2}\right),$$

де $\sigma = \text{median}\{d_{ij} : d_{ij} > 0\}$.

Додатково в граф вводилися петлі-самозв'язки (i,i), що забезпечило збереження власних характеристик вершини під час агрегації:

$$(i,i) \in E,\ w_{ii} = 1.$$

Після побудови графа датасет було розбито на три підмножини, що не перетинаються: навчальну (80%), валідаційну (10%) та тестову (10%). Для кожної підмножини сформовано бінарні маски, які використовувалися під час

тренування та оцінювання моделі.

У роботі використано графову нейронну мережу на основі механізму уваги (GAT) для задачі регресії. Модель отримує на вході матрицю ознак вершин, матрицю інцидентності (список ребер) та атрибути ребер, після чого формує нові представлення вершин шляхом агрегації інформації з локального оточення. Для підвищення стабільності навчання застосовано два послідовні графові шари уваги з нелінійностями та регуляризацією. Завершальний вихідний шар перетворює отримані представлення у скалярний прогноз цільової змінної.

Навчання графової нейронної мережі здійснюється в режимі напівконтрольованого навчання, коли значення цільової змінної відомі лише для частини вершин графа.

Алгоритм навчання складається з таких основних етапів:

– ініціалізація параметрів моделі графової нейронної мережі;

– пряме поширення сигналу (forward pass), під час якого для кожної вершини графа обчислюється прогнозоване значення цільової змінної;

– обчислення функції втрат лише для вершин, що належать до навчальної підмножини;

– зворотне поширення похибки та оновлення параметрів моделі методом градієнтного спуску;

– оцінювання якості моделі на валідаційній підмножині;

– повторення процедури до досягнення збіжності або заданої кількості епох.

Такий підхід дозволяє використовувати всю структуру графа під час навчання, навіть якщо частина вершин не бере безпосередньої участі в оптимізації функції втрат.

3.3 Опис програми

Програмна реалізація виконана мовою програмування Python із використанням хмарного середовища Google Colab, що забезпечує доступ до апаратно-

го прискорення у вигляді графічних процесорів та дозволяє ефективно працювати з великими наборами даних.

Загальна структура програми має модульний характер і включає такі основні етапи:

– зчитування та попередня обробка табличних даних;

– формування графового подання даних на основі просторових зв’язків;

– побудова та ініціалізація моделей машинного навчання;

– навчання моделей і контроль здатності узагальнювати;

– оцінювання якості прогнозування та візуалізація результатів.

Центральним елементом програмної реалізації є графова нейронна мережа класу Graph Attention Network, адаптована до задачі регресійного аналізу. Вона працює на графі районів, де вершини – це окремі записи (райони) з табличними ознаками, а ребра з’єднують просторово близькі райони; кожне ребро має вагу, що відбиває географічну близькість (перетворену гаусовим ядром) [7].

Архітектура має таку логіку:

– вхідний лінійний проєктор ознак – робить легку адаптацію до простору, де увага працює стабільніше;

– два послідовні GAT-шари з багатоголовою увагою, кожен із нормалізацією, нелінійністю та внутрішнім “feed-forward” (FFN) блоком у резидуальному підключенні;

– вихідна «голова» (readout) – невеликий багатошаровий персептрон для перетворення ознак вузла у скалярний прогноз (лог-ціна);

– регуляризація: дропаут вузлів, edge-dropout (випадкове проріджування ребер під час навчання), нормалізація ознак ребер;

– функція втрат – середньоквадратична похибка (MSE) на лог-шкалі; оцінювання – MAE/RMSE на оригінальній грошовій шкалі через зворотнє перетворення.

Таким чином, кожен вузол на виході містить прогноз своєї цільової змінної з урахуванням не лише власних ознак, а й контексту з оточення.

Загальну структуру застосованої моделі подано на рисунку 3.1. Архітек-

тура поєднує методи обробки табличних ознак із механізмом графової уваги, що дозволяє враховувати просторові зв’язки між районами. Мережа складається з вхідного шару проекції ознак, двох послідовних графових блоків із багатоголовою увагою, нормалізацією та резидуальними з’єднаннями, а також вихідної регресійної «голови», яка перетворює отримане представлення кожного вузла у прогнозовану цільову змінну.

```
==========================================================================================================================================
Layer (type:depth-idx)                   Input Shape               Output Shape              Param #                   Trainable
==========================================================================================================================================
GATWrapper                               [20640, 13]               [20640, 1]                --                        True
├─GATReg: 1-1                            --                        [20640, 1]                --                        True
│    └─Linear: 2-1                       [20640, 13]               [20640, 13]               182                       True
│    └─GATConv: 2-2                      [20640, 13]               [20640, 256]              1,024                     True
│    │    └─Linear: 3-1                  [20640, 13]               [20640, 256]              3,328                     True
│    │    └─Linear: 3-2                  [247683, 1]               [247683, 256]             256                       True
│    │    └─SumAggregation: 3-3          [247683, 2, 128]          [20640, 2, 128]           --                        --
│    └─BatchNorm1d: 2-3                  [20640, 256]              [20640, 256]              512                       True
│    └─Sequential: 2-4                   [20640, 256]              [20640, 256]              --                        True
│    │    └─Linear: 3-4                  [20640, 256]              [20640, 256]              65,792                    True
│    │    └─ReLU: 3-5                    [20640, 256]              [20640, 256]              --                        --
│    │    └─Dropout: 3-6                 [20640, 256]              [20640, 256]              --                        --
│    │    └─Linear: 3-7                  [20640, 256]              [20640, 256]              65,792                    True
│    └─GATConv: 2-5                      [20640, 256]              [20640, 128]              896                       True
│    │    └─Linear: 3-8                  [20640, 256]              [20640, 256]              65,536                    True
│    │    └─Linear: 3-9                  [247683, 1]               [247683, 256]             256                       True
│    │    └─SumAggregation: 3-10         [247683, 2, 128]          [20640, 2, 128]           --                        --
│    └─BatchNorm1d: 2-6                  [20640, 128]              [20640, 128]              256                       True
│    └─Sequential: 2-7                   [20640, 128]              [20640, 128]              --                        True
│    │    └─Linear: 3-11                 [20640, 128]              [20640, 128]              16,512                    True
│    │    └─ReLU: 3-12                   [20640, 128]              [20640, 128]              --                        --
│    │    └─Dropout: 3-13                [20640, 128]              [20640, 128]              --                        --
│    │    └─Linear: 3-14                 [20640, 128]              [20640, 128]              16,512                    True
│    └─Sequential: 2-8                   [20640, 128]              [20640, 1]                --                        True
│    │    └─Linear: 3-15                 [20640, 128]              [20640, 128]              16,512                    True
│    │    └─ReLU: 3-16                   [20640, 128]              [20640, 128]              --                        --
│    │    └─Dropout: 3-17                [20640, 128]              [20640, 128]              --                        --
│    │    └─Linear: 3-18                 [20640, 128]              [20640, 1]                129                       True
==========================================================================================================================================
Total params: 253,495
Trainable params: 253,495
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 5.31
==========================================================================================================================================
Input size (MB): 6.44
Forward/backward pass size (MB): 1312.72
Params size (MB): 1.01
Estimated Total Size (MB): 1320.16
==========================================================================================================================================
```

Рисунок 3.1 – Загальна архітектура нейронної мережі

Послідовно розглянемо етапи обробки даних у цій архітектурі.

Перед першим GAT-шаром застосовується лінійне перетворення ознак із нелінійністю:

$$\vec{h}_i^{(0)} = \phi(W_{in} x_i + b_{in}),$$

де $\phi = \mathrm{ELU}$ – нелінійна активація;

$W_{in} \in \mathbb{R}^{m \times m}$;

$b_{in} \in \mathbb{R}^m$.

Після цього додається резидуальний місток $\vec{h}_i^{(0)} = \tilde{x}_i + x_i$ до вхідного вектора, що полегшує оптимізацію на ранніх етапах та зменшує ризик затухання градієнта.

Далі йде GAT-шар, який виконує перетворення ознак вершин та обчислює ваги уваги по сусідах. Для кожної голови $m = 1,...,H$ (тут H – число голів) застосовуються незалежні лінійні перетворення ознак вершин і, за потреби, ребер:

$$Q^{(h)} = XW_q^{(h)},\ K^{(h)} = XW_k^{(h)},\ e_{ij}'^{(h)} = \varphi^{(h)}(w_{ij}),$$

де $W_q^{(h)}$, $W_k^{(h)} \in \mathbb{R}^{m \times F_h}$ – матриці ваг;

$\varphi^{(h)} : \mathbb{R}^{de} \to \mathbb{R}^r$ – лінійна мапа ребрових ознак ($de = 1,\ r = 1$).

Після побудови матриць запитів $Q^{(h)}$, ключів $K^{(h)}$ та відображення реберних ознак $\varphi^{(h)}(e_{ij})$ виконується формування попередніх коефіцієнтів уваги. Ідея полягає в тому, що кожен вузол має «запит» до своїх сусідів, а сусіди у свою чергу подають свої «ключі»; додатково враховується інформація про ребро, що їх з'єднує. Комбінуючи ці три складові, отримуємо скалярну величину $e_{ij}^{(h)}$, яка відображає, наскільки суттєвим є вузол j для оновлення вузла i у межах конкретної голови h:

$$e_{ij}^{(h)} = \text{LeakyReLu}(a^{(h)\text{T}}[Q_i^{(h)} \,||\, K_j^{(h)} \,||\, \varphi^{(h)}(e_{ij})]),$$

де $||$ – конкатенація векторів;

$a^{(h)}$ – вектор параметрів, що навчається.

Функція LeakyReLU використовується для того, щоб уникнути затухання градієнтів у випадку, коли всі значення могли б стати від'ємними й заблокувати навчання. Таким чином, кожне з'єднання $i \to j$ отримує свою початкову вагу,

яка ще не нормалізована.

Наступним кроком є нормалізація ваг уваги. Щоб отримані значення можна було інтерпретувати як відносні «коефіцієнти важливості», вони пропускаються через softmax по всіх сусідах вузла i:

$$\alpha_{ij}^{(h)} = \frac{\exp(e_{ij}^{(h)})}{\sum_{k \in N(i)} \exp(e_{ik}^{(h)})}.$$

Таким чином, усі коефіцієнти $\alpha_{ij}^{(h)}$ для заданого вузла i утворюють розподіл ймовірностей на множині його сусідів $N(i)$. Це означає, що вплив кожного сусіда масштабується так, щоб у сумі дорівнювати одиниці. У математичному сенсі це – перехід від довільних «сирих» ваг до нормалізованих коефіцієнтів, які дозволяють робити коректні порівняння між різними сусідами.

Далі виконується агрегація повідомлень від сусідів. Для вузла i будується новий вектор ознак як зважена сума перетворених векторів його сусідів, де коефіцієнтами є $\alpha_{ij}^{(h)}$:

$$\vec{h}_i^{(h)\text{new}} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij}^{(h)} K_J^{(h)} \right).$$

Функція $\sigma\left(\sum_{j \in N(i)} \alpha_{ij}^{(h)} K_J^{(h)} \right)$ є нелінійною (зазвичай використовується ELU або ReLU), що дозволяє враховувати складніші залежності. Це означає, що нове представлення вузла i формується зваженим середнім його сусідів, де ваги обчислюються динамічно й залежать не лише від значень ознак, але й від характеристик самого ребра.

Оскільки використовується багатоголова увага (H незалежних механізмів), то остаточне оновлене представлення вузла утворюється комбінацією ре-

зультатів усіх голів. У першому шарі результати голів об’єднуються конкатенацією, тобто просто зшиваються в довший вектор:

$$\vec{h}_i^{\text{new}} = \|_{h=1}^{H} \vec{h}_i^{(h)\text{new}}.$$

Це дозволяє моделі одночасно враховувати кілька «ракурсів» уваги, кожен з яких може виділяти різні аспекти інформації. У подальших шарах іноді використовується усереднення, що зберігає компактність розмірності $\vec{h}_i^{\text{new}}$.

Після агрегації повідомлень необхідно забезпечити стабільність навчання. Для цього в архітектурі застосовується нормалізація ознак, а саме батч-нормалізація (Batch Normalization), яка для кожної координати вектора $\vec{h}_i^{\text{new}}$ віднімає середнє та ділить на стандартне відхилення, що обчислені на поточному міні-батчі, а також має коефіцієнти масштабування та зсуву, що підлягають навчанню. Формально це можна записати як

$$\hat{h}_{i,k} = \gamma_k \cdot \frac{h_{i,k}^{\text{new}} - \mu_k}{\sigma_k + \varepsilon} + \beta_k,$$

де $h_{i,k}^{\text{new}}$ – k-а координата вектора;

μ_k – середнє відхилення по цій координаті в межах батчу;

σ_k – стандартне відхилення по цій координаті в межах батчу;

γ_k, β_k – параметри, які навчаються.

Це вирівнює масштаби ознак і дозволяє уникнути їх неконтрольованого зростання, полегшуючи оптимізацію.

Далі вводиться feed-forward блок (FFN). Це окремий невеликий двошаровий персептрон, який застосовується до кожного вузла незалежно. Він має вигляд:

$$\mathrm{FNN}(\vec{h}_i) = W_2 \xi(W_1 \vec{h}_i + b_1) + b_2,$$

де ξ – нелінійна активація.

У цьому блоці часто використовується Dropout – випадкове занулення частини координат під час навчання, що знижує перенавчання. Сенс FFN полягає в тому, щоб дати мережі змогу виконувати додаткові нелінійні перетворення ознак після уваги, підвищуючи виразність моделі.

Щоб уникнути деградації сигналу при проходженні через багато перетворень, застосовується резидуальне з'єднання. Це означає, що вихід FFN додається до початкового вектора, який у нього входив:

$$\vec{h}_i^{\,\mathrm{out}} = \vec{h}_i^{\,\mathrm{in}} + \mathrm{FNN}(\hat{\vec{h}}_i).$$

Таким чином, модель фактично навчається не повністю «переписувати» ознаки, а коригувати їх. Це полегшує передавання градієнтів під час зворотного поширення та дає змогу навчатись глибшим архітектурам.

В описаному вигляді побудовано два послідовних блоки: перший повертає вектори більшої розмірності (через конкатенацію результатів багатьох голів), другий – стискає їх назад у компактнішу форму (використовуючи усереднення по головах). Послідовне застосування двох шарів уваги дозволяє враховувати не лише найближчих сусідів, а й сусідів другого порядку, що збагачує представлення вузлів ширшим контекстом. Після цих двох блоків залишається отримати остаточний прогноз для кожної вершини графа. Для цього використовується невелика вихідна «голова», яка є послідовністю лінійних перетворень з активаціями, що перетворює вектор ознак вузла у скаляр:

$$\hat{y}_i^{(\log)} = w^{\mathrm{T}} \varphi(W \vec{h}_i^{\,\mathrm{final}} + b) + c,$$

де $\varphi(W \vec{h}_i^{\,\mathrm{final}} + b)$ – нелінійність.

Результат $\hat{y}_i^{(\log)}$ інтерпретується як прогноз логарифма ціни житла. Логарифмування було використано для того, щоб зменшити вплив дуже великих значень і зробити розподіл цільової змінної більш «нормальним» з точки зору статистики. Фактична функція втрат, яку мінімізує модель під час навчання, L – це середньоквадратична похибка між передбаченими та істинними лог-цінами:

$$L = \frac{1}{|S_{train}|} \sum_{i \in S_{train}} (\hat{y}_i^{(\log)} - y_i^{(\log)})^2 .$$

Під час оцінки якості результатів перехід назад відбувається за формулою $\hat{y}_i = \exp(\hat{y}_i^{(\log)}) - 1$, після чого обчислюються метрики MAE і RMSE вже у грошовій шкалі.

Вибір саме такої архітектури є обґрунтованим, оскільки Graph Attention Networks дають можливість поєднати табличні ознаки з просторовими зв’язками та адаптивно визначати вагу кожного сусіднього району для поточного прогнозу. На відміну від класичних методів, які працюють лише з локальними параметрами об’єкта, GAT інтегрує контекст його оточення, що є критично важливим у задачах прогнозування цін нерухомості, де вплив географічних факторів та соціально-економічних характеристик навколишнього середовища часто домінує над індивідуальними ознаками об’єкта.

Навчання розробленої графової нейронної мережі здійснювалося у режимі керованого навчання з використанням навчальної підвибірки графових даних із 300 епохами. Для оптимізації параметрів моделі застосовано алгоритм Adam, який забезпечує адаптивне коригування швидкості навчання та добре зарекомендував себе в задачах глибокого навчання.

Функцією втрат обрано середньоквадратичну помилку (MSE), що мінімізується на логарифмічній шкалі цільової змінної. Такий підхід дозволяє зменшити вплив викидів та стабілізувати процес навчання. Контроль

перенавчання здійснювався шляхом використання валідаційної вибірки та збереження найкращого стану моделі за мінімальним значенням помилки MAE.

Для підвищення узагальнювальної здатності моделі застосовано регуляризаційні методи, зокрема dropout для ознак вершин та випадкове проріджування ребер графа під час навчання. Крім того, використання нормалізації ознак і резидуальних з'єднань дозволило уникнути деградації градієнтів при збільшенні глибини мережі.

Після завершення навчання якість моделі оцінювалася на тестовій підвибірці. Прогнозовані значення поверталися до початкової грошової шкали шляхом зворотного експоненціального перетворення, після чого обчислювалися метрики MAE та RMSE, які використовуються для оцінки точності регресійних моделей.

Для обґрунтування ефективності використання графової нейронної мережі було реалізовано та досліджено базові регресійні моделі, зокрема багатошаровий перцептрон та ансамблеву модель випадкового лісу. Ці моделі навчалися на тих самих попередньо оброблених ознаках, але без урахування графової структури даних. Порівняння результатів здійснювалося за допомогою метрик MAE та RMSE на тестовій вибірці. Окрім числових показників, було побудовано криві навчання, що відображають динаміку функції втрат та похибок на тренувальній і валідаційній вибірках. Візуалізація результатів виконувалася з використанням бібліотеки Matplotlib, що дозволило наочно проаналізувати процес збіжності моделей та порівняти їхню узагальнювальну здатність. Отримані результати підтверджують доцільність використання графових нейронних мереж у задачах прогнозування, де важливу роль відіграють просторові та структурні зв'язки між об'єктами.

Висновки за розділом 3

У даному розділі був описаний програмний комплекс для дослідження методів машинного навчання на графових даних у задачі регресійного аналізу.

Реалізація виконувалася мовою програмування Python із використанням сучасних бібліотек для аналізу даних і глибокого навчання, що забезпечило гнучкість, відтворюваність та розширюваність запропонованого підходу.

Використання хмарного середовища Google Colab дозволило ефективно організувати обчислювальні експерименти, зокрема завдяки доступу до апаратного прискорення у вигляді графічних процесорів. Це дало можливість навчати глибокі нейронні мережі на великих графових структурах без необхідності локальних обчислювальних ресурсів, а також спростило збереження результатів та повторюваність експериментів.

У межах розділу було запропоновано підхід до представлення табличних просторових даних у вигляді графа, де вершини відповідають окремим об’єктам, а ребра відображають просторову близькість між ними. Побудова графової структури на основі географічних координат із використанням метрики гаверсинуса дозволила врахувати просторові зв’язки між об’єктами без втрати інформації про їхні індивідуальні характеристики. Використання ваг ребер, обчислених за допомогою гаусового ядра, забезпечило плавне моделювання ступеня взаємного впливу між сусідніми вершинами. Для розв’язання задачі прогнозування було реалізовано графову нейронну мережу класу Graph Attention Network, адаптовану до регресійної постановки. Запропонована архітектура поєднує лінійні перетворення ознак, механізм багатоголової уваги, нормалізацію та резидуальні з’єднання, що забезпечує стабільність навчання та ефективне врахування як локальної, так і контекстної інформації графа. Механізм уваги дозволив моделі адаптивно визначати внесок кожного сусіднього вузла у формування прогнозу, що є принциповою перевагою порівняно з класичними методами агрегації.

У розділі також детально описано процедуру навчання моделі, включно з вибором функції втрат, оптимізатора та методів регуляризації. Застосування логарифмічного перетворення цільової змінної сприяло стабілізації процесу навчання та покращенню якості прогнозів. Оцінювання результатів здійснювалося

за допомогою метрик MAE та RMSE на початковій шкалі значень, що забезпечило інтерпретованість отриманих результатів.

Для обґрунтування доцільності використання графових нейронних мереж було реалізовано базові моделі машинного навчання, що використовуються у регресійному аналізі, зокрема багатошаровий перцептрон та випадковий ліс та порівняння отриманих за їх допомогою результатів.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Порівняльний аналіз результатів графових нейронних мереж і класичних моделей

Для оцінювання ефективності запропонованого підходу було проведено серію обчислювальних експериментів із використанням графової нейронної мережі на основі механізму уваги (GAT) та порівняльних базових моделей – багатошарового перцептрону (MLP) і ансамблевого методу випадкового лісу (Random Forest) на наборі даних California Housing Dataset.

На рисунку 4.1 зображено сабграф (фрагмент графа) другого порядку, побудований навколо тестового вузла графа. Вершини відповідають окремим об’єктам набору даних, а ребра – просторовим зв’язкам між ними, сформованим на основі k найближчих сусідів. Червоним кольором виділено центральний вузол, для якого здійснюється прогноз цільової змінної. Показаний фрагмент графа ілюструє локальне оточення об’єкта, інформація з якого агрегується моделлю Graph Attention Network під час формування векторного представлення вузла.

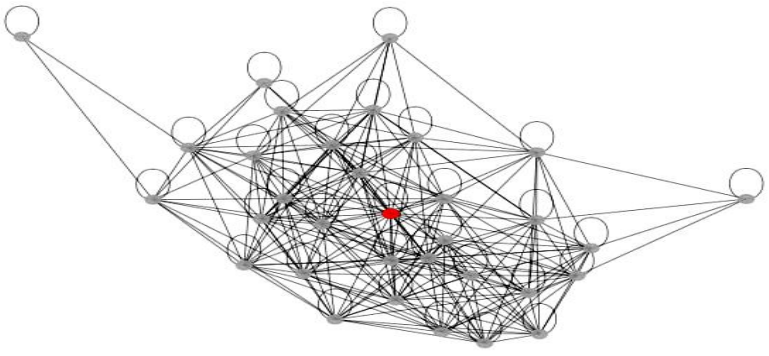

Рисунок 4.1 – Фрагмент графа для тестового об’єкта набору даних California Housing

Якість моделей оцінювалася за допомогою стандартних регресійних

метрик MSE, RMSE та MAE, що обчислювалися на валідаційній і тестовій вибірках. Додатково аналізувалися криві навчання, які відображають динаміку збіжності моделей та їхню узагальнювальну здатність. На графіках одночасно відображено значення помилок на тренувальній і валідаційній вибірках, а також наведено збільшений фрагмент для пізніх епох навчання (≥ 250), що дозволяє оцінити стабільність моделей після збіжності.

На рисунку 4.2 наведено криві зміни середньоквадратичної помилки (MSE) на логарифмічній шкалі для графової нейронної мережі та багатошарового перцептрону. Для моделі GAT спостерігається різке зменшення помилки на початкових етапах навчання та швидкий вихід на область низьких значень MSE. Після приблизно 100 епох крива стабілізується, а тренувальна та валідаційна помилки практично збігаються, що свідчить про хорошу узагальнювальну здатність моделі та відсутність перенавчання. У випадку MLP зменшення MSE відбувається значно повільніше. Навіть на пізніх етапах навчання значення помилки залишаються вищими, а розрив між тренувальною та валідаційною кривими є більш помітним, що вказує на обмежену ефективність моделі без урахування графової структури даних.

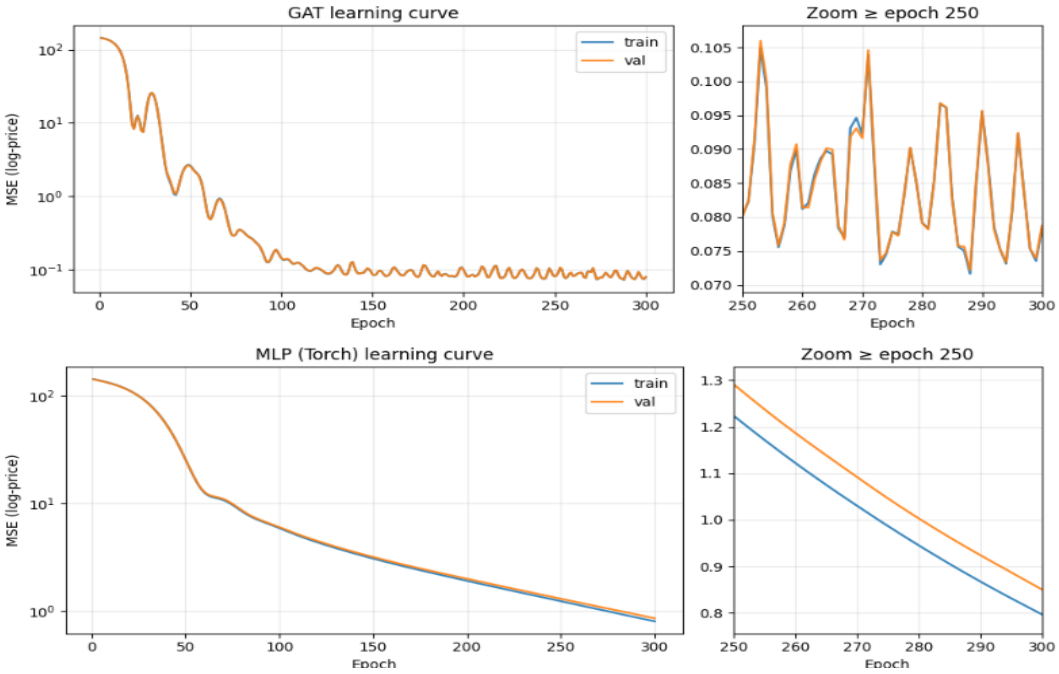

Рисунок 4.2 – Криві навчання для GAT та MLP за метрикою MSE

На рисунку 4.3 показано динаміку кореня середньоквадратичної помилки (RMSE) для моделей GAT та MLP. Для графової нейронної мережі значення RMSE швидко зменшується та стабілізується на низькому рівні. Невеликі коливання на пізніх епохах пояснюються стохастичністю навчання та використанням регуляризаційних механізмів, зокрема dropout. Для MLP значення RMSE зменшується поступово, проте залишається істотно більшим навіть після 300 епох навчання. Це свідчить про те, що модель не здатна повністю відтворити залежності, обумовлені просторовими зв’язками між об’єктами, лише за табличними ознаками.

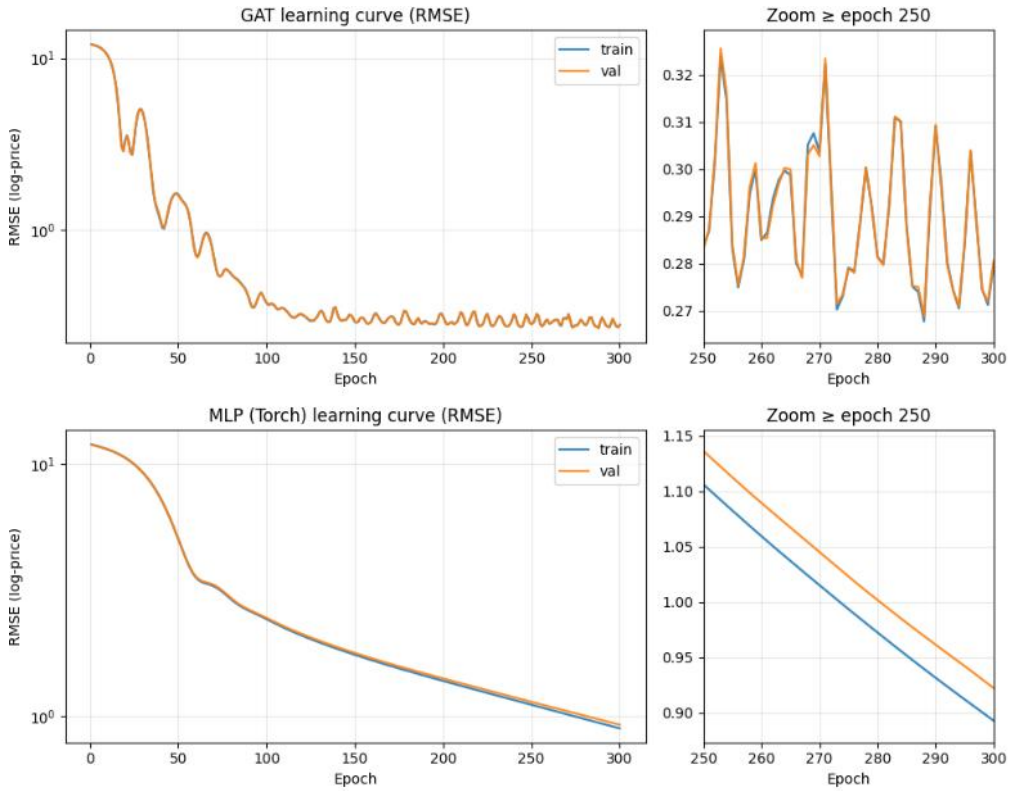

Рисунок 4.3 – Криві навчання для GAT та MLP за метрикою RMSE

На рисунку 4.4 наведено зміну середньої абсолютної похибки (MAE) для обох моделей. Для GAT після короткого перехідного етапу модель швидко виходить на стабільний рівень MAE, який зберігається протягом усього подальшого навчання. Тренувальна та валідаційна криві практично співпадають, що вказує на стабільність прогнозів. Для MLP характерною є нестабільна поведін-

ка на початкових етапах навчання з різкими піками MAE, що пов’язано з високою чутливістю моделі до масштабів цільової змінної. Навіть після стабілізації значення MAE для MLP залишаються на порядок більшими порівняно з GAT.

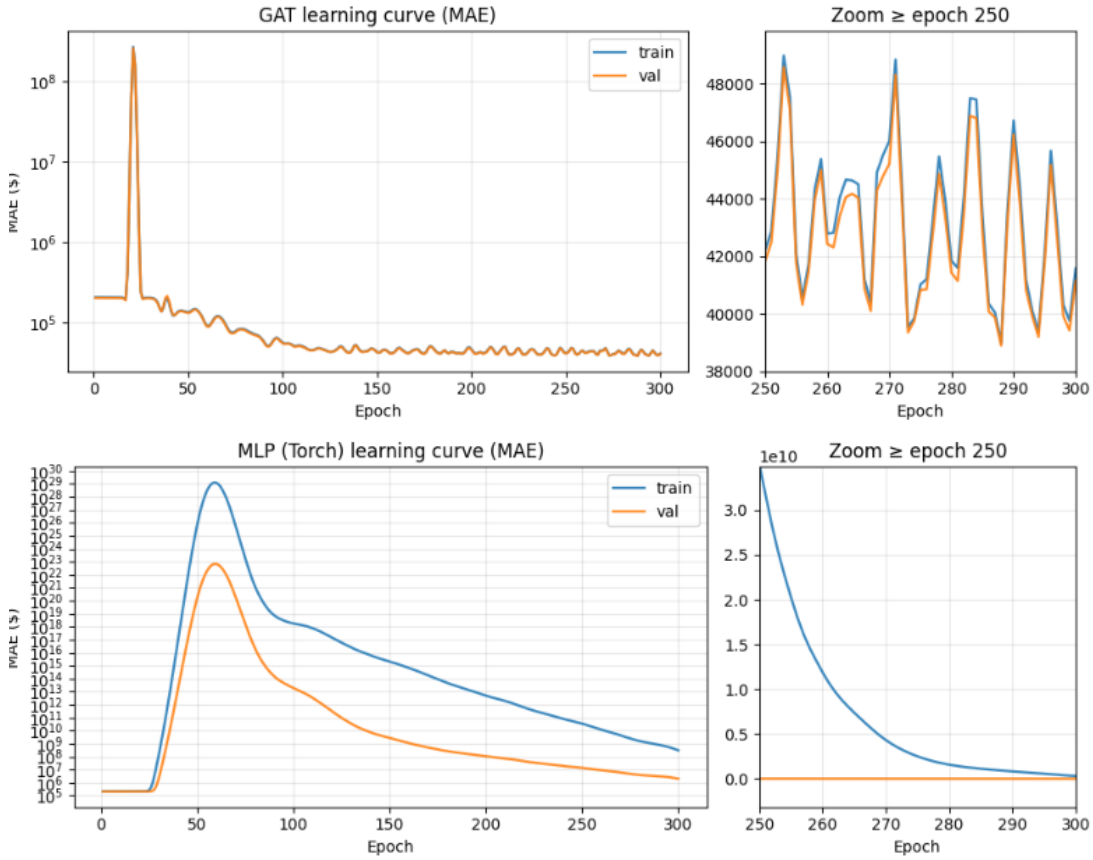

Рисунок 4.4 – Криві навчання для GAT та MLP за метрикою MAE

На рисунку 4.5 показано відносну важливість ознак, отриману за допомогою моделі випадкового лісу. Найбільший внесок у прогнозування медіанної вартості житла мають соціально-економічні та просторові характеристики, зокрема медіанний дохід домогосподарств, географічні координати та близькість до океану. Отримані результати підтверджують, що просторове розташування об’єктів та їхнє оточення є критично важливими факторами у задачі прогнозування, що узгоджується з ідеологією графових нейронних мереж, які безпосередньо моделюють такі залежності.

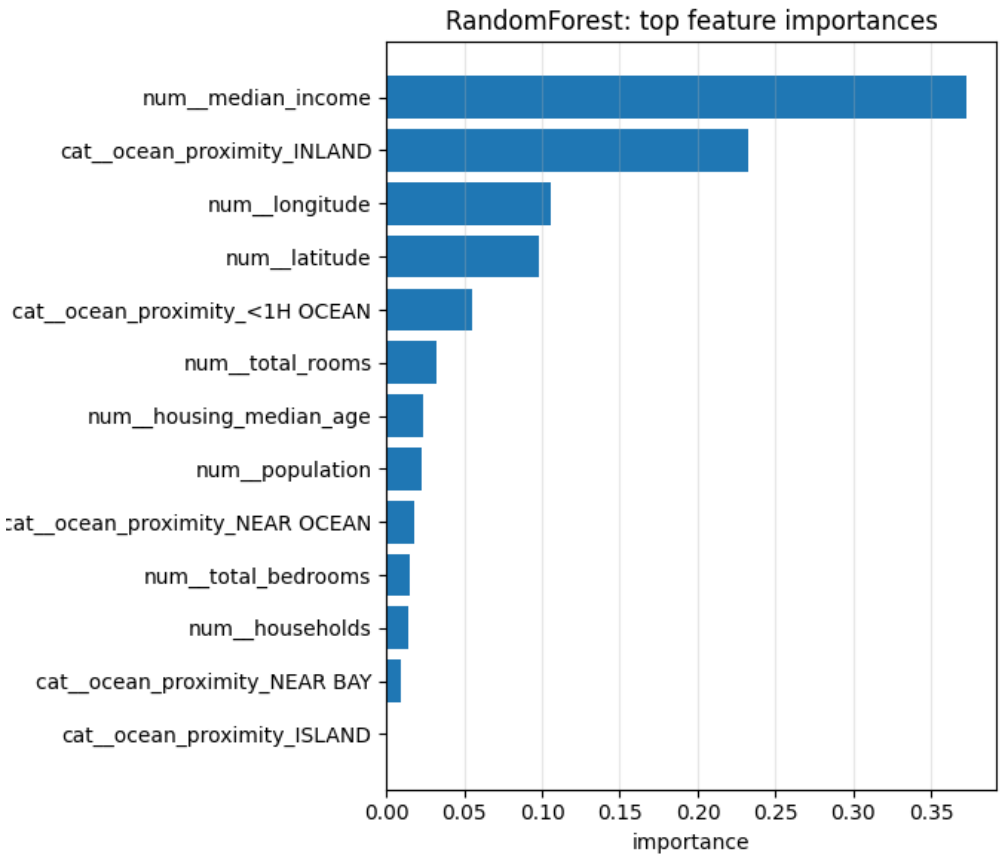

Рисунок 4.5 – Важливість ознак у моделі Random Forest

У таблиці 4.1 наведено значення метрик MAE та RMSE для трьох моделей: графової нейронної мережі (GAT), ансамблевої моделі випадкового лісу (Random Forest) та багатошарового перцептрону (MLP). Усі моделі навчалися та тестувалися на однакових підвибірках даних, що забезпечує коректність порівняння отриманих результатів.

Таблиця 4.1 – Порівняльна таблиця метрик якості регресійних моделей

Модель	MAE (дол. США)	RMSE (дол. США)
GAT (graph)	39 752	58 736
Random Forest	40 837	61 648
MLP (torch)	61 648	3 148 358

Аналіз наведених у таблиці результатів свідчить про явну перевагу графової нейронної мережі над іншими розглянутими моделями. Модель GAT продемонструвала найменші значення як MAE, так і RMSE, що вказує на її ви-

соку точність прогнозування та стабільну узагальнювальну здатність. Ансамблева модель Random Forest показала результати, близькі до графової нейронної мережі, однак поступається їй за обома метриками. Це підтверджує, що врахування нелінійних взаємодій між ознаками є важливим, проте без явного моделювання просторових зв'язків модель не здатна досягти максимальної точності. Натомість багатошаровий перцептрон MLP продемонстрував значно гірші результати, що проявляється у суттєво вищих значеннях MAE та RMSE. Це свідчить про те, що класичні нейронні мережі, які працюють виключно з табличними ознаками, неефективні для задач прогнозування на просторово структурованих даних. Таким чином, отримані числові результати підтверджують доцільність застосування графових нейронних мереж для регресійного аналізу в задачах, де між об'єктами існують суттєві просторові та структурні залежності. Використання механізму уваги дозволяє моделі адаптивно враховувати вплив оточення кожного об'єкта, що забезпечує підвищення точності прогнозування порівняно з традиційними підходами.

4.2 Аналіз точності прогнозування та інтерпретація результатів

Наведемо приклад роботи побудованих нейронних мереж під час прогнозування вартості житла на окремих об'єктах тестової множини (рис. 4.6). Можна побачити, що графова нейронна мережа на основі механізму уваги (GAT) у більшості випадків забезпечує найменшу абсолютну похибку прогнозу порівняно з багатошаровим перцептроном та ансамблевим методом випадкового лісу. Прогнози моделі GAT є більш стабільними та ближчими до реальних значень, особливо для об'єктів із середнім і високим рівнем вартості, що свідчить про ефективне використання інформації з локального графового оточення.

Водночас модель Random Forest демонструє конкурентні результати для окремих об'єктів, однак її похибка є менш стабільною та зростає для деяких прикладів тестової вибірки. Найбільші відхилення спостерігаються у випадку

багатошарового перцептрона, що підтверджує обмеженість підходу, який не враховує просторові та структурні зв'язки між об'єктами. Отримані результати узгоджуються з узагальненими метриками якості та підтверджують доцільність застосування графових нейронних мереж для задач регресійного аналізу з просторово пов'язаними даними.

index	idx	y_true_$	GAT_$	MLP_$	RF_$
0	2912	118399.984375	97393.828125	55725.5625	102647.85463969676
1	5076	119800.0	157921.40625	230181.375	147806.50994091557
2	14999	129700.0390625	144605.84375	201666.671875	147535.99375500093
3	47	137500.015625	134507.71875	155195.0	130944.8458055463
4	9975	189400.015625	125701.046875	161843.0625	122400.62878442224
5	10191	194699.96875	178933.171875	160547.140625	219202.33732460704
6	19251	197399.953125	181945.21875	730459.5625	235180.66683399296
7	6788	225800.09375	185282.53125	275552.46875	238184.47820988396
8	20276	242900.109375	301747.9375	62612.125	276877.15338897816
9	20178	283200.0	248831.859375	72748.03125	330561.6714954061
10	18607	394900.125	358511.09375	572707.4375	360076.19839534676
11	15809	427300.125	370492.71875	329932.375	343622.32777673594

Рисунок 4.6 – Порівняння реальних та прогнозованих значень вартості житла для окремих об'єктів тестової вибірки

Отримані експериментальні результати та їх порівняльний аналіз свідчать, що графова нейронна мережа на основі механізму уваги (GAT) забезпечує найкращу точність прогнозування серед розглянутих моделей. За значеннями метрик MAE та RMSE вона незначно перевищує модель Random Forest і суттєво випереджає багатошаровий перцептрон. Таким чином, модель на основі графової уваги демонструє кращу узагальнювальну здатність у задачі прогнозування, ніж класичні та ансамблеві архітектури. Це підтверджує ефективність використання графового подання даних для задач регресійного аналізу з просторовими залежностями.

Водночас слід зазначити, що перевага графової нейронної мережі не є абсолютною. Якість прогнозування GAT значною мірою залежить від особливостей набору даних, способу побудови графа, зокрема вибору метрики близькості, кількості сусідів, наявності та параметризації ваг ребер, а також архітектури самої мережі. За невдалої побудови графової структури або недостатньої кіль-

кості інформативних зв’язків ефективність GAT може знижуватися. Модель Random Forest, у свою чергу, демонструє стабільну якість прогнозування без явного урахування просторової структури даних. Вона може перевершувати графову нейронну мережу в ситуаціях, коли основну роль відіграють індивідуальні табличні ознаки, а просторові залежності є слабкими або зашумленими. Крім того, Random Forest менш чутлива до вибору гіперпараметрів та структури даних.

Таким чином, вибір між графовою нейронною мережею та моделлю Random Forest має ґрунтуватися на характері задачі та властивостях даних. GAT є більш придатною для задач із вираженою просторовою або структурною взаємозалежністю між об’єктами, тоді як Random Forest може бути конкурентною або навіть переважною альтернативою для задач із переважно табличною природою ознак.

Висновки за розділом 4

За результатами порівняння моделей встановлено, що графова нейронна мережа класу Graph Attention Network забезпечує найкращу точність прогнозування серед розглянутих підходів. Урахування просторових зв’язків між районами дозволяє моделі ефективніше використовувати інформацію про навколишній контекст об’єкта, що є критично важливим для задачі оцінювання вартості житла.

Модель Random Forest продемонструвала близькі до GAT результати, що свідчить про її високу ефективність у задачах регресії. Проте відсутність явного механізму моделювання просторових залежностей обмежує потенціал цієї моделі в умовах, коли географічний контекст відіграє суттєву роль.

Багатошаровий перцептрон показав значно гіршу якість прогнозування, що підтверджує недостатність підходів, які розглядають об’єкти як незалежні та не враховують взаємозв’язки між ними. Таким чином, результати експери-

ментів підтверджують доцільність застосування графових нейронних мереж для задач регресії з просторово структурованими даними та демонструють їхню перевагу над класичними нейронними та ансамблевими методами в подібних умовах.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено експериментальне дослідження застосування графових нейронних мереж до нетипової для них задачі регресійного аналізу. На відміну від класичних задач класифікації або прогнозування на явно заданих графах, у даній роботі графова структура формувалася штучно на основі просторової близькості об’єктів, що дозволило перевірити ефективність графових моделей у менш очевидних умовах.

У ході дослідження було показано, що графові нейронні мережі мають значний потенціал для задач регресії, однак їхня ефективність суттєво залежить від способу побудови графового представлення даних. Формування зв’язків між об’єктами, вибір метрики близькості, кількості сусідів та параметризації ваг ребер відіграють ключову роль у якості кінцевого прогнозу. Таким чином, використання графових моделей потребує усвідомленого підходу та ретельного аналізу властивостей вихідних даних. Для оцінювання доцільності застосування графових нейронних мереж було виконано порівняння з класичними методами машинного навчання. Результати експериментів показали, що за наявності виражених просторових взаємозв’язків між об’єктами графова нейронна мережа здатна перевершувати табличні моделі. Водночас у випадках, коли такі взаємозв’язки є слабкими або некоректно змодельованими, традиційні підходи, зокрема ансамблеві методи, можуть демонструвати співставну або навіть кращу точність.

Проведене дослідження підтвердило, що механізм графової уваги є ефективним інструментом для адаптивного врахування впливу оточення кожного об’єкта. Разом із тим, отримані результати свідчать про відсутність універсального рішення: графові нейронні мережі не є заміною класичним методам, а повинні розглядатися як потужне доповнення, яке доцільно застосовувати у задачах зі структурованими або просторово залежними даними. Таким чином, у роботі було не лише продемонстровано можливість використання графових нейронних мереж у задачах регресійного аналізу, але й показано межі їх застосов-

ності. Отримані висновки підкреслюють необхідність обґрунтованого вибору моделей машинного навчання та можуть слугувати основою для подальших досліджень у напрямку графових нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Авсітідійський М. М. Application of graph neural networks for regression analysys tasks. *4-та Міжнародна науково-практична конференція «Європейські студії. Навчання і викладання у світі технологій»*: зб. матеріалів конференції (м. Клуж-Напока, Румунія, 12 листопада 2025 р.). 2025. С. 222.

2. Авсітідійський М. М. Застосування графових нейронних мереж для задач регресійного аналізу. *Міжнародний конкурс студентських наукових робіт зі штучного інтелекту*: зб. матеріалів конкурсу (м. Київ, травень–листопад 2025 р.). Київ : КПІ, 2025.

3. Авсітідійський М. М. Графові нейронні мережі у задачах прогнозування параметрів технологічних процесів. *Всеукраїнська науково-практична конференція здобувачів вищої освіти і молодих учених «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві»*: зб. матеріалів конференції (м. Харків, 25 листопада 2025 р.). Харків : ХНАДУ, 2025.

4. Zhou J., Cui G., Hu S., Zhang Z., Yang C., Liu Z., Wang L., Li C., Sun M. Graph neural networks: A review of methods and applications. *AI Open. 2020*. Vol. 1. P. 1–26.

5. Klinke N., Buchkremer S., Elend L., Kalaidov M., von Tschammer T. AI-based performance prediction and its application on the design and simulation of cooling plates for battery electric vehicles. *Future Automotive Production Conference (FAPC).* Wolfsburg, Germany, 17–18 May 2022. P. 1–10.

6. Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. Graph Attention Networks. *Proc. of ICLR*. 2018. С. 1–12.

7. He J., Koric S., Abueidda D., Najafi A., Jasiuk I. Geom-DeepONet: A point-cloud-based deep operator network for field predictions on 3D parameterized geometries. *Computer Methods in Applied Mechanics and Engineering.* 2024. Vol. 429. P. 1–19.

8. Bonnet F., Mazari J. A., Munzer T., Yser P., Gallinari P. An extensible

benchmarking graph-mesh dataset for studying steady-state incompressible Navier–Stokes equations. *Workshop on Geometrical and Topological Representation Learning, ICLR*. 2022 P. 10-24.

9. Zhang A., Lipton C. Z.., Li M., Smola J. A. Dive into Deep Learning. Cambridge University Press. 2023. 30–54 p.

10. Stevens E., Antiga L., Viehmann T. Deep Learning with PyTorch. New York : Manning. 2020. 190–246 p.

11. Kaggle. California Housing Prices. URL: https://www.kaggle.com/datasets/camnugent/california-housing-prices (дата звернення: 19.11.2025).

12. Басюк Т. М., Литвин В. В., Захарія Л. М., Кунанець Н. Е. Машинне навчання. Львів : Видавництво «Новий Світ - 2000», 2021. 315 с.

13. Кононова К. Ю. Машинне навчання: методи та моделі. Харків : ХНУ імені В. Н. Каразіна, 2020. 280 с.

14. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., 2019. 850 p.

15. Литвин В. В., Пасічник В. В., Нікольський Ю. В. Аналіз даних та знань. Львів : Магнолія 2006, 2021. 276 с.

16. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця : ВНТУ, 2024. 258 с.

17. Фісун М. Т., Кравець І. О., Казмірчук П. П., Ніколенко С. Г. Інтелектуальний аналіз даних: практикум Львів : «Новий Світ – 2000», 2024. 162 с.

18. Bruce P., Bruce A., Gedeck P. Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. O'Reilly Media. 2020. 360 p.

19. Frasconi P., Gori M., Sperduti A. A general framework for adaptive processing of data structures. IEEE Transactions on Neural Networks. 1998. Vol. 9, No. 5. P. 768–786.

20. Xu K., Hu W., Leskovec J., Jegelka S. How powerful are graph neural networks? International Conference on Learning Representations (ICLR). 2018. P. 1–17.

21. Daigavane A., Ravikumar P., Aggarwal C. Understanding convolutions on graphs. Distill. 2021. Vol. 6, No. 2. P. 23.

22. Heindl C. Graph neural networks for node-level predictions. arXiv preprint arXiv:2007. 2020. P. 1–29.

23. Zhang M., Cui Z., Neumann M., Chen Y. An end-to-end deep learning architecture for graph classification. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018. Vol. 32, No. 1. P. 4438–4445.

24. Nikolentzos G., Thomas M., Ramirez Rivera A., Vazirgiannis M. Image classification using graph-based representations and graph neural networks. *Computer Vision – ECCV 2020* Workshops. Springer. 2021. P. 1–17.

25. Defferrard M., Bresson X., Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems (NeurIPS). 2016. Vol. 29. P. 3844–3852.

26. Левицька Т., Котихова Л. Графові нейронні мережі та алгоритми PageRank у задачах прогнозування популярності хештегів у соціальних мережах. *Вісник Приазовського Державного Технічного Університету. Серія: Технічні науки*. 2025. №51. С. 50–56.

27. Bloemheuvel S., Brouwer L., van der Veen, M. Graph neural networks for multivariate time series regression: TISER-GCN. *Journal of Intelligent & Fuzzy Systems.* 2023. Vol. 45, No. 5. P. 1–15.

28. Chen X., Zhang J., Ni J., Li X., Bian Y., Islam M. M., Mondal A., Wei H., Luo D. RegExplainer: Generating explanations for graph neural networks in regression tasks. arXiv preprint arXiv:2307.07840. 2024. P. 1–19.

29. Li C. T. Graph neural networks for tabular data learning: methods and applications. arXiv preprint arXiv:2401.02143. 2024. P. 1–48.

30. Yamada M., Sugiyama M. A. Survey of dynamic graph neural networks. arXiv preprint arXiv:2404.18211. 2024. P. 1–76.