КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітньо-наукова програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:
"Розробка технології у сфері оцінки нерухомості методами науки про дані**"**

Студентки 2-го курсу групи ІАВ-21
Вікторії НЕЧИПОРУК
(ім'я, прізвище)

_____
(підпис студента)

Науковий керівник:
Кандидат економічних наук, доцент
(науковий ступінь, вчене звання)
Ігор МІРОШНИЧЕНКО
(ім'я, прізвище)

_____ _____
(дата) (підпис)

Попередній захист:

_____

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління _____ Віктор МОРОЗОВ _____
(підпис) (прізвище, ініціали) (дата)

Київ - 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор
МОРОЗОВ

_____
«____» ____20___року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Вікторія НЕЧИПОРУК

Група ІАВ-21

1. Тема кваліфікаційної роботи

«Розробка технології у сфері оцінки нерухомості методами науки про дані»

Затверджена наказом по університету від «_____»_____ 2024р. № _____ _____.

2. Строк подання студентом готової роботи – «19» травня 2025р.

3. Цільова установка та вихідні дані до роботи.

Цільова установка роботи полягає в аналізі та розробці технології автоматизованої оцінки вартості житлової нерухомості на вторинному ринку міста Києва з використанням методів науки про дані, зокрема алгоритмів машинного навчання, таких як XGBoost. Основна мета - створення високоточної моделі прогнозування цін на основі характеристик об'єктів нерухомості та її інтеграція у веб-додаток для практичного застосування в ріелторській, банківській та інших сферах. Реалізація системи передбачає вибір відповідного

стеку технологій, зокрема Python, бібліотек машинного навчання (scikit-learn, XGBoost) та фреймворку Flask для створення веб-інтерфейсу. Вихідні дані включають набір даних про квартири, зібраний з відкритих джерел, таких як платформа LUN.ua, що містить інформацію про ціни, площу, розташування, технічні характеристики, стан ремонту та географічні ознаки.

4. Зміст роботи

Зміст роботи полягає у визначенні оптимального підходу до автоматизованої оцінки вартості житлової нерухомості на ринку міста Києва з використанням методів машинного навчання, зокрема алгоритму XGBoost, та розробці веб-додатку для практичного застосування результатів. Дослідження включає виокремлення ключових факторів, що впливають на ціноутворення, а також аналіз їхнього впливу на точність прогнозування. Здійснюється порівняння ефективності різних моделей машинного навчання (лінійна регресія, випадковий ліс, XGBoost) за метриками точності. Проводиться аналіз існуючих рішень для оцінки нерухомості з метою визначенаня їхніх переваг і обмежень. Описується методологія розробки, що базується на фреймворку CRISP-DM, яка охоплює етапи збору та обробки даних, моделювання, оцінки моделей і розгортання у вигляді веб-додатку. Пояснюється доцільність використання методів машинного навчання для підвищення точності та автоматизації оцінки порівняно з традиційними методами, а також пропонуються рекомендації щодо впровадження розробленої технології в реальні ринкові процеси.

5. Перелік графічного матеріалу (слайдів)

Дана кваліфікаційна робота магістра налічує 2 таблиці та 46 рисунків, які позначають математичні формули для побудови моделей лінійної регресії, методу випадкового лісу, екстремального градієнтного бустингу порівняльних характеристик, прикладів, додатків, джерел, датасетів та отриманих результатів.

6. Календарний план

№ з/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.10.24	01.10.24
2.	Затвердження тем дипломних робіт та призначення наукових керівників	2	27.12.24	27.12.24
3.	Формування переліку матеріалів, літератури з проблематики дипломної роботи	10	08.01.25	07.01.25
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.25	18.01.25
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	19.01.25 - 20.01.25	20.01.25
6.	Підготовка розділу 1 «АНАЛІЗ ТЕОРЕТИКО-МЕТОДОЛОГІЧНИХ ОСНОВИ ОЦІНЮВАННЯ ОБ'ЄКТІВ НЕРУХОМОСТІ»	10	12.02.25	01.05.25
7.	Підготовка розділу 2 «ФОРМАЛІЗАЦІЯ МЕТОДІВ АНАЛІЗУ ДАНИХ У ПРОЄКТІ РОЗРОБКИ ТЕХНОЛОГІЇ ДЛЯ ОЦІНКИ ЖИТЛА»	14	08.03.25	01.05.25
8.	Підготовка розділу 3 «РОЗРОБКА МОДЕЛІ ПРОГНОЗУВАННЯ ВАРТОСТІ НЕРУХОМОСТІ МЕТОДАМИ НАУКИ ПРО ДАНІ»	14	01.04.25	01.05.25
9.	Підготовка розділу 4 «ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ	13	20.04.25	01.05.25

	ОЦІНКИ У СФЕРІ НЕРУХОМОСТІ»			
10.	Оформлення роботи. Підготовка висновків і пропозицій	15	27.04.25	01.05.25
11.	Передача кваліфікаційної роботи науковому керівникові	2	01.05.25	01.05.25
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	12.05.25	12.05.25
13.	Попередній захист кваліфікаційної роботи	5	12.05.25	15.05.25

Дата видачі завдання «_____» ___2024р.

Керівник роботи: кандидат економічних наук, доцент Ігор МІРОШНИЧЕНКО
(посада, ім’я, прізвище)

_____
(підпис)

Завдання прийняла до виконання студентка групи ІАВ-21:_____

Вікторія НЕЧИПОРУК_____

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Вікторії НЕЧИПОРУК.

Тема роботи – «Розробка технології у сфері оцінки нерухомості методами науки про дані».

Мета дипломної роботи магістра – підвищення якості прогнозу вартості об'єктів на ринку нерухомості шляхом розробки технології оцінки з використанням удосконалених методів науки про дані та алгоритмів машинного навчання.

Об'єкт дослідження – процес оцінки вартості нерухомості.

Предмет дослідження – методи та моделі науки про дані, зокрема алгоритми машинного навчання, для аналізу даних і прогнозування вартості нерухомості.

Наукова новизна роботи полягає в удосконаленні технології оцінки нерухомості, яка інтегрує моделі градієнтного бустінгу з оптимізованими методами обробки даних, що забезпечує вищу точність прогнозів порівняно з традиційними підходами завдяки врахуванню нелінійних залежностей і географічних факторів. Практична цінність роботи полягає в розробці веб-додатку, який автоматизує процес оцінки, надаючи зручний інструмент для користувачів, а також у формуванні рекомендацій для впровадження технології.

Дипломна робота складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього налічує 106 сторінки, з них перелік посилань з 22 джерел на 2 сторінках та додатки на 15 сторінках.

Ключові слова: нерухомість, методи науки про дані, лінійна регресія, Xgboost, метод випадкового лісу.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

CRISP-DM (Cross-Industry Standard Process for Data Mining).

XGBoost (XGB) - eXtreme Gradient Boosting.

RFM – Random Forest Method.

ІШІ – штучний інтелект.

MAE - Mean Absolute Error.

MAPE - Mean Absolute Percentage Error.

MdAE - Median Absolute Error.

MdAPE -Median Absolute Percentage Error.

NLP – Natural Language Processing.

ВСТУП

Оцінка нерухомості є ключовим елементом функціонування ринку нерухомості, який відіграє важливу роль у бізнесі, інвестиціях, кредитуванні та державному управлінні. У сучасних умовах зростання обсягів даних, розвитку інформаційних технологій та методів науки про дані з'являються нові можливості для підвищення точності, об'єктивності та автоматизації оцінки вартості нерухомих об'єктів. Традиційні методи оцінки, такі як порівняльний, дохідний чи витратний, часто є суб'єктивними, трудомісткими та обмеженими у врахуванні складних ринкових факторів. У цьому контексті застосування методів науки про дані, зокрема машинного навчання, стає актуальним рішенням, яке дозволяє обробляти великі об'єми даних, знаходити неявні закономірності та прогнозувати точну вартості нерухомості.

Мета роботи полягає в підвищенні якості прогнозу вартості об'єктів на ринку нерухомості шляхом розробки технології оцінки з використанням удосконалених методів науки про дані та алгоритмів машинного навчання.

Завдання дослідження включають:

1. Аналіз сучасних методів оцінки нерухомості та можливостей науки про дані.
2. Визначення ключових факторів, що впливають на вартість нерухомості.
3. Розробка та порівняння моделей машинного навчання (лінійна регресія, випадковий ліс, екстремальний градієнтний бустинг) для прогнозування цін.
4. Оцінка ефективності моделей за метриками точності.
5. Створення веб-додатку на базі Flask для автоматизації оцінки.
6. Формування рекомендацій щодо впровадження розробленої технології.

Актуальність теми дослідження спричинена багатьма факторами. По-перше, ринок нерухомості в Україні, зокрема у великих містах, таких як Київ, характеризується високою динамікою цін, що залежить від численних факторів: розташування, інфраструктури, економічних умов, попиту та пропозиції. По-

друге, сучасні технології дозволяють автоматизувати процеси оцінки, зменшуючи залежність від суб'єктивних оцінок експертів і підвищуючи прозорість ринку. По-третє, зростання попиту на цифрові рішення в сфері нерухомості, таких як proptech-платформи та автоматизовані системи оцінки, вимагає розробки нових інструментів, які відповідають сучасним викликам. Таким чином, розробка технології оцінки нерухомості на основі методів науки про дані є актуальною як з наукової, так і з практичної точки зору, оскільки сприяє підвищенню ефективності ринкових процесів та прийняттю обґрунтованих рішень.

Об'єкт дослідження – процеси оцінки вартості нерухомості на вторинному ринку житлової нерухомості міста Києва.

Предмет дослідження – методи науки про дані, зокрема алгоритми машинного навчання, для аналізу даних і прогнозування вартості нерухомості.

Моделі та методи дослідження включають використання алгоритмів машинного навчання (лінійна регресія, випадковий ліс, XGBoost) для прогнозування цін, методологію CRISP-DM для структуризації процесу розробки, методи передобробки даних (One-Hot Encoding, Target Encoding, дискретизація ознак), а також статистичний аналіз для оцінки якості моделей за метриками.

Наукова новизна роботи полягає в удосконаленні технології оцінки нерухомості, яка інтегрує моделі градієнтного бустінгу з оптимізованими методами обробки даних, що забезпечує вищу точність прогнозів порівняно з традиційними підходами завдяки врахуванню нелінійних залежностей і географічних факторів. Практична цінність роботи заключається в розробці веб-додатку, який автоматизує процес оцінки, надаючи зручний інструмент для ріелторів, інвесторів і покупців, а також у формуванні рекомендацій для впровадження технології в реальні ринкові процеси.

РОЗДІЛ 1
АНАЛІЗ ТЕОРЕТИКО-МЕТОДОЛОГІЧНИХ ОСНОВИ ОЦІНЮВАННЯ ОБ’ЄКТІВ НЕРУХОМОСТІ

1.1 Аналіз методології науки про дані у сфері оцінки нерухомості

У сучасному світі ринок нерухомості є важливим сегментом економіки, який впливає на багато інших галузей. Оцінка нерухомості є ключовою складовою цього ринку, яка допомагає визначити вартість об'єктів нерухомості для різних цілей: купівлі-продажу, оренди, інвестицій тощо. Застосування методології науки про дані для оцінки нерухомості має значний практичний потенціал, сприяючи автоматизації процесів, збільшенню точності прогнозів та розв’язанню основних задач ринку нерухомості, таких як суб’єктивність оцінки чи нестача даних.

Ось деякі з основних етапів і методів, які можуть бути використані в методології Data Science для оцінки нерухомості:

1. Збір та підготовка даних: Отримання різноманітних даних про нерухомість, таких як ціни на житло, характеристики нерухомості (площа, кількість кімнат, розташування), демографічні дані про район, дані про ринок нерухомості тощо. Підготовка даних може включати очищення, перетворення та нормалізацію.

2. Візуалізація та розвідувальний аналіз: Важливо візуалізувати дані для отримання загального уявлення про їх розподіл, виявлення аномалій та визначення зв'язків між різними змінними. Наприклад, графіки розподілу цін на житло за різними районами або відношення ціни до площі.

3. Моделювання цін на нерухомість: Використання алгоритмів машинного навчання для прогнозування цін на нерухомість на основі її характеристик. Можливі підходи включають метод градієгтного бустингу, ансамблеві методи тощо. Важливо також враховувати фактори, що впливають на ціни, такі як економічні та соціальні чинники, інфраструктура тощо.

4. Оцінка моделі та валідація: Після побудови моделі важливо оцінити її ефективність за допомогою відповідних метрик, таких як середньоквадратична помилка для задачі регресії. Для оцінки моделі градієнтного бустингу в задачі прогнозування вартості нерухомості варто використовувати комбінацію кількох метрик, таких як MSE, MAE, R-квадрат, RMSE, MAPE та медіанну абсолютну помилку. Це дозволить отримати більш повне уявлення про точність і надійність моделі, а також її стійкість до викидів та помилок. Модель потрібно також звірити з тестовим набором даних задля підтвердження її прикладного використання.

5. Інтерпретація результатів: Після побудови та оцінки моделі важливо інтерпретувати результати для зрозуміння факторів, які впливають на ціни на нерухомість та можливостей для оптимізації цінової політики.

6. Впровадження та моніторинг: У разі успішної оцінки та валідації моделі, її можна впровадити для прогнозування цін на нерухомість. Проте важливо постійно моніторити її роботу та вносити корективи у випадку зміни умов або якості даних.

Для реалізації розробки технології науки про дані у сфері оцінки нерухомості було обрано Існує кілька методологій та фреймворків, що використовуються в науці про дані, зокрема:

CRISP-DM (Cross-Industry Standard Process for Data Mining) – це загальноприйнята методологія, яка охоплює всі етапи проекту в галузі науки про дані [1]. Вона включає такі етапи: осмислення бізнесових задач, аналіз даних, підготовка та збір даних, моделювання, оцінка та розгортання (рисунок 1.1.1).

Методологія складається з шести етапів, адаптованих до контексту оцінки вартості житла:

1. Розуміння бізнес-задачі: Визначення цілей, таких як автоматизація оцінки чи виявлення ключових факторів впливу на ціну, з урахуванням потреб ріелторів, інвесторів або покупців.
2. Розуміння даних: Аналіз доступних даних про нерухомість, включаючи характеристики об’єктів, ринкові тренди та геолокаційні особливості.

3. Підготовка даних: Очищення (видалення пропусків, дублікатів), трансформація (нормалізація числових змінних, кодування категорій) та збагачення даних (додавання нових ознак, наприклад, індексу престижності району).
4. Моделювання: Застосування алгоритмів машинного навчання (градієнтний бустинг, нейронні мережі) для прогнозування цін.
5. Оцінка моделі: Перевірка моделі на тестових даних з використанням метрик (RMSE, R^2) для оцінки її точності та універсальності.
6. Розгортання моделі: Впровадження моделі у вигляді сервісу (наприклад, через API) для автоматизованої оцінки вартості нерухомості станом на зараз, з можливістю масштабування.

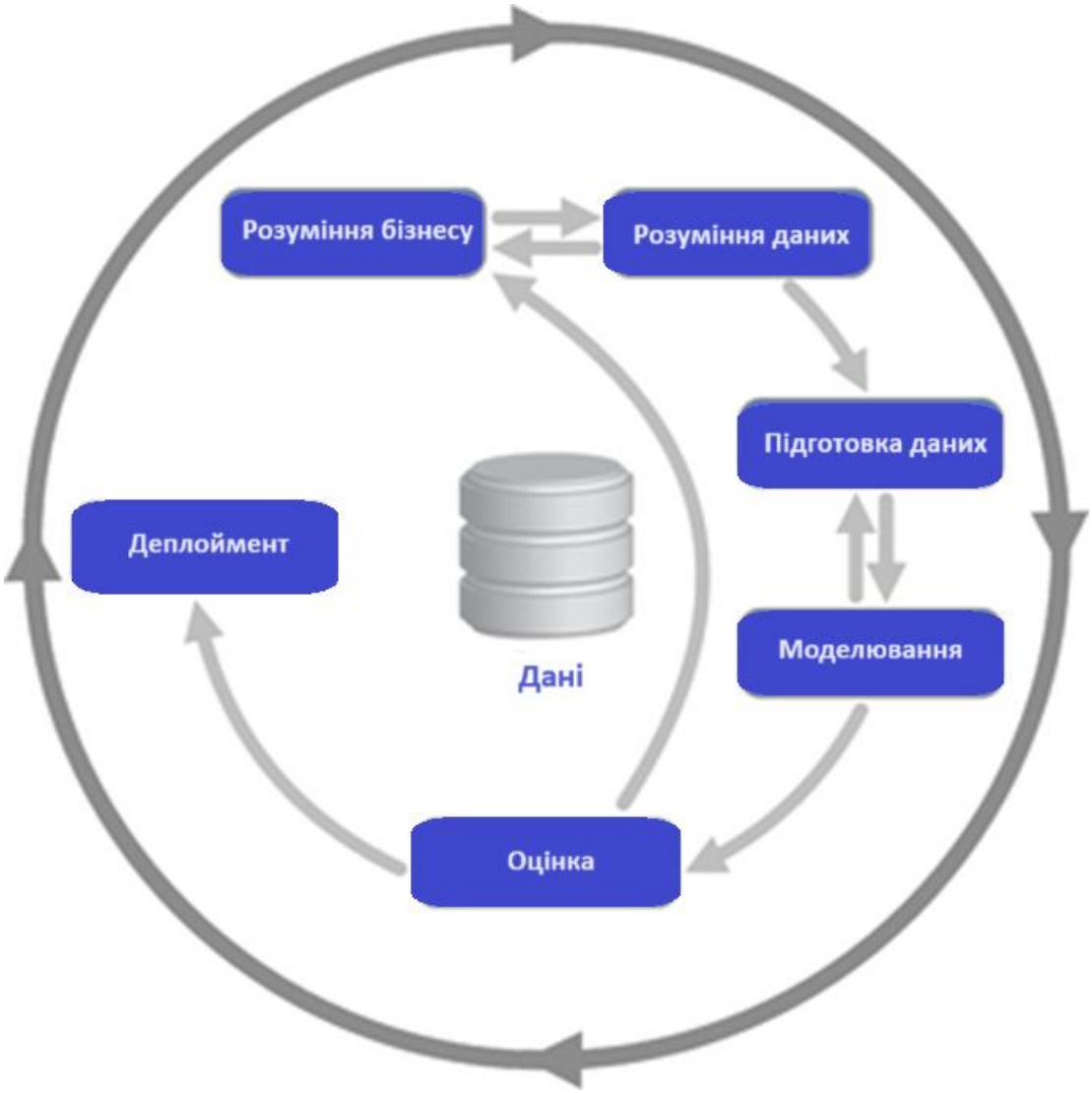


Рисунок 1.1.1 Етапи методології CRISP-DM

1.2 Аналіз факторів впливу та загальних методів встановлення вартості нерухомого майна

- На формування та оцінку вартості житлової нерухомості впливають безліч факторів, серед яких:
- Місцезнаходження: Відстань від нерухомості до центру міста, транспортна доступність, близькість до зон громадських послуг, наявність торговельних центрів, медичних та освітніх закладів, місць поклоніння, водойм, берегової лінії тощо. Відстань до джерел шуму також суттєво впливає на цінність нерухомості.
- Топографічні та геологічні характеристики: Характеристики ділянки, такі як рельєф і тип ґрунту, суттєво впливають на витрати та складність будівництва. Нерівна місцевість або складний рельєф можуть вимагати додаткових інженерних рішень, наприклад, укріплення ґрунту чи спорудження підтримуючих конструкцій, що збільшує витрати і час на будівництво. Вирівнювання ділянки за допомогою засипки може створювати ризики для стійкості ґрунту. Будівництво інфраструктури в таких умовах також ускладнене, що підвищує загальні витрати
- Корисна площа: це територія, яку можна вільно використовувати без юридичних чи фізичних перешкод. Юридичні обмеження можуть включати, наприклад, права проходу чи проїзду, а фізичні — наявність скелястих ділянок або водойм. Більша корисна площа підвищує цінність об’єкта, адже вона визначає, наскільки ефективно нерухомість може бути задіяна для різних потреб
- Технічні характеристики нерухомості: До них належать матеріали, використані в будівництві, проєкт, електрика, опалення, теплоізоляція, стійкість до землетрусів, характеристики ліфтів, системи безпеки. Якісний, добре спланований житловий проєкт із подібними фізичними характеристиками буде ціннішим, ніж звичайний.

- Характеристики навколишнього середовища: розташування нерухомості безпосередньо впливає на її цінність. Це включає стан інфраструктури та надбудови, щільність забудови, близькість до центру, наявність рекреаційних зон і зелених насаджень, рівень шуму, забруднення, проблеми з паркуванням, доступ до освіти, медицини, місць поклоніння, магазинів, візуальну якість, кліматичні та мікрокліматичні особливості, соціальну структуру та близькість до водних ландшафтів.
- Попит і пропозиція: Бажання купити чи продати, стиль ведення переговорів, наявність посередників, кількість власників нерухомості чи адресатів, а також родинні чи дружні зв’язки між покупцем і продавцем можуть впливати на ціну.
- Економічні показники: Загальні економічні тенденції в країні впливають на баланс попиту та пропозиції, що відображається на вартості нерухомості разом з іншими факторами оточення. Нерухомість вважається довгостроковим інвестиційним інструментом, що захищає активи від інфляції. Об’єкти, які приносять дохід від оренди та зростають у ціні, залежать від глобальних і національних фінансових політик. Інфляція підвищує витрати на будівництво, що відображається на вартості нерухомості.
- Фактор зонування: Зонування є одним із найважливіших технічних факторів оцінки, особливо для земельних ділянок. Щоб нерухомість вважалася земельною ділянкою на момент оцінки, необхідно враховувати відповідні зональні характеристики.
- Політичні чинники: Політична стабільність і безпекова ситуація суттєво впливають на ціноутворення в секторі нерухомості. Збройний конфлікт в Україні, зокрема його ескалація у 2022 році, спричинив зниження вартості об'єктів у прифронтових регіонах через високі ризики, водночас у західних областях спостерігається зростання попиту та цін на тлі внутрішньої міграції. Така нестабільність ускладнює прогнозування цін і знижує інвестиційну привабливість ринку.

Процес оцінки нерухомості ґрунтується на застосуванні низки підходів, які умовно можна класифікувати на три основні групи: традиційні, статистичні та сучасні методи оцінки. Певні групи методів мають свої плюси та мінуси, а також вибір відповідного підходу залежить від цілей оцінки, доступності даних і характеру об’єкта. Їх перелік та структуру було описано в Modern Methods Approach in Real Estate Valuation (рис.1.1.2).

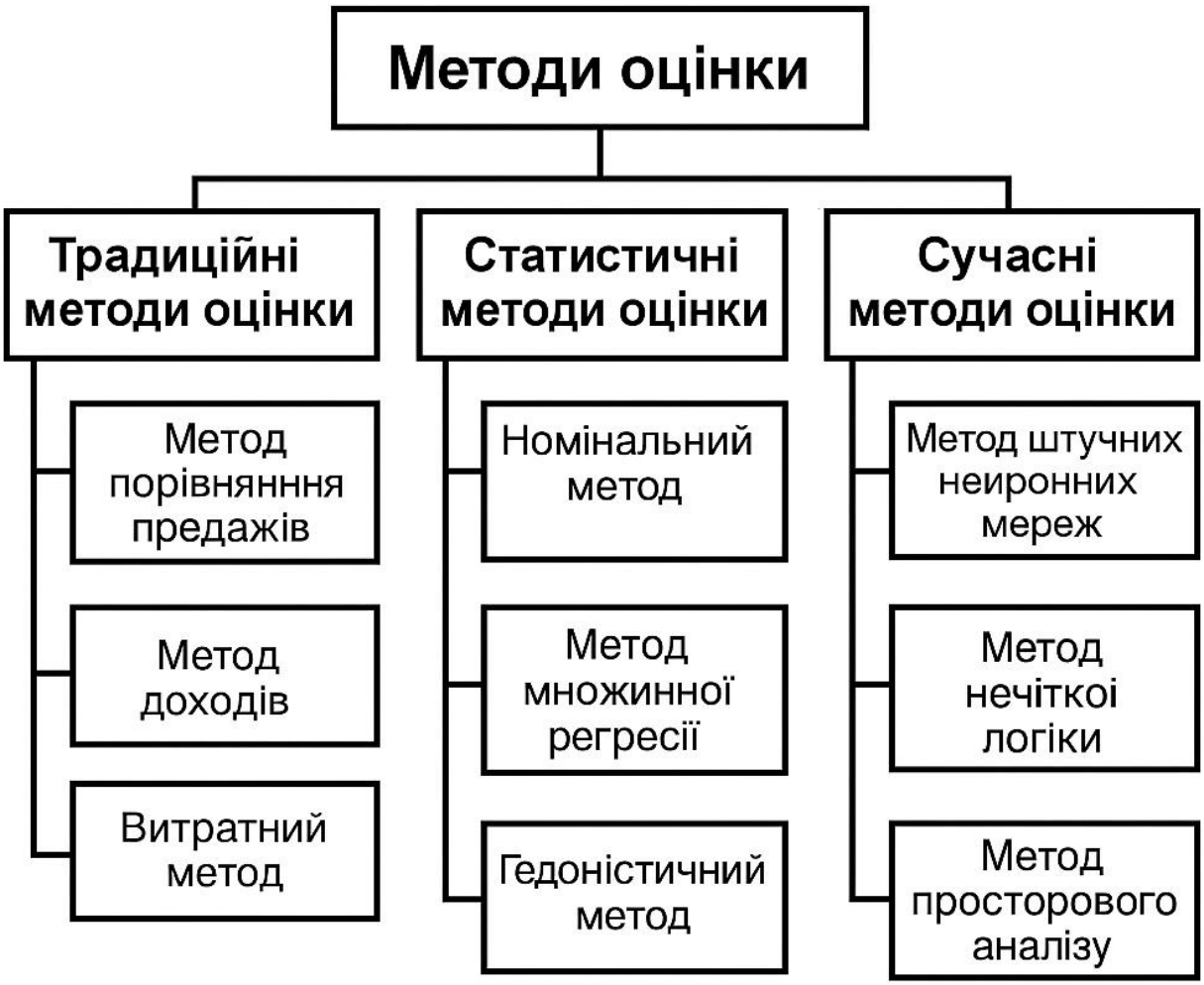


Рисунок 1.2.1 Методи оцінки у сфері нерухомості[2]

Традиційні методи оцінки

- Метод порівняння продажів. Цей метод базується на аналізі ринкових угод із подібними об’єктами нерухомості. Він передбачає порівняння оцінюваного об’єкта з аналогами, приймаючи до уваги їх характеристики, стан, розташування та час продажу.

- Дохідний метод :Оцінка ґрунтується на потенційному доході, який може генерувати об'єкт нерухомості. Застосовується переважно для комерційної нерухомості, яка приносить регулярний дохід (наприклад, оренда).
- Витратний метод: Вартість визначається як сума витрат на створення (або відновлення) об’єкта з урахуванням фізичного стану, збереження функціональності та економічних витрат.

Статистичні методи оцінки

- Номінальний метод: Метод передбачає просту побудову залежності між базовими параметрами об’єкта (площа, кількість кімнат тощо) та його вартістю.
- Метод множинної регресії: Базується на математичному моделюванні залежності вартості від множини незалежних предикторів. Регресійний аналіз дозволяє виявити та кількісно оцінити внесок кожного з факторів у загальну вартість об'єкта. Метод є популярним серед практикуючих аналітиків через простоту інтерпретації та широке застосування у програмних засобах (Excel, Python, R).
- Гедоністичний метод : Оцінка базується на припущенні, що кожна характеристика об’єкта (наприклад, вид із вікна, близькість до транспорту) має свою окрему вартість. Загальна ціна формується як сума вартостей усіх таких характеристик. Метод дозволяє точно моделювати споживчі переваги і широко застосовується у містах із розвиненим ринком нерухомості.

Сучасні методи оцінки (методи науки про дані)

З розвитком технологій та накопиченням великих обсягів даних з’явилася можливість застосовувати сучасні методи аналізу — зокрема, методи науки про дані у сфері оцінки нерухомості. Ці підходи мають низку переваг: здатність виконувати обробку великої частини інформації, знаходити складні нелінійні залежності, автоматизувати процеси прийняття рішень та підвищувати точність оцінок.

- Метод штучних нейронних мереж: Цей підхід дозволяє моделювати складні залежності між характеристиками об'єкта нерухомості та його вартістю без необхідності жорстко формалізувати функціональний зв’язок. Мережі здатні самостійно виявляти важливі ознаки (features) та взаємозв’язки. Особливо ефективні при аналізі великих та різнорідних наборів даних.
- Метод нечіткої логіки: Цей метод застосовується в умовах невизначеності або коли ознаки не піддаються точній кількісній оцінці. Наприклад, поняття «гарна інфраструктура» чи «поганий стан будівлі» можуть бути описані нечіткими множинами. Метод дозволяє враховувати експертні судження в аналітичних моделях.
- Метод просторового аналізу: Передбачає врахування просторової прив’язки об'єкта до елементів міської інфраструктури (зупинки транспорту, школи, лікарні тощо), зон планування та характеристик навколишнього середовища. Реалізується за допомогою геоінформаційних систем (GIS) або бібліотек аналізу просторових даних у Python (geopandas, shapely). Просторовий аналіз є основою побудови сучасних автоматизованих систем оцінки (AVM — Automated Valuation Models).
- Метод випадкового лісу (Random forest) – це ансамблевий підхід у машинному навчанні, який застосовується для задач класифікації, регресії та інших. Його суть полягає в тому, що під час навчання моделі створюється багато дерев рішень, а підсумковий результат формується шляхом голосування більшості (для класифікації) або усереднення прогнозів (для регресії) цих дерев. Такий підхід дозволяє підвищити точність і стійкість моделі, зменшуючи ризик перенавчання.
- Метод градієнтного бустингу (Gradient Boosting) – це підхід до машинного навчання, який можна використовувати для вирішення проблем класифікації та регресії. Він заснований на ідеї, що група слабких учнів може створити більш точний предиктор, коли вони працюють разом.

- Метод кластерного аналізу :Використовується для групування об’єктів нерухомості за схожими характеристиками, наприклад, поділ районів на кластери за рівнем цін чи інфраструктурним розвитком. Такі методи, як K-Means або DBSCAN, допомагають виявити приховані патерни в даних (наприклад, зони з подібними ціновими трендами) і можуть бути основою для побудови сегментованих моделей оцінки, що враховують локальні особливості ринку.

Ці методи є основними в сучасних підходах до оцінки нерухомості, але не охоплюють усі можливі варіанти. Існують також інші методи, такі як гедонічна регресія, що розкладає ціну на складові характеристики, аналіз часових рядів для прогнозування ринкових трендів, а також гібридні підходи, які комбінують машинне навчання з експертними системами чи статистичними моделями.

1.3 Огляд літературних джерел

Для того щоб обґрунтувати актуальність проблематики оцінки вартості нерухомості, було зроблено огляд літератури.

У статті Predictive Analysis of Local House Prices було здійснено дослідження для прогнозу цін на житло в Техасі для подальшої оцінювання доцільності інвестування в нерухомість [3]. Дані для аналізу були взяті з онлайн-платформи нерухомості Redfin та бази даних для рієлторів Multiple Listing Service. Серед методів, які застосовувались, були лінійна регресія та алгоритм Random Forest. Виявилося, що Random Forest працює значно точніше: середня абсолютна відносна помилка (MAPE) склала 3.59%, у той час як для лінійної регресії - 4.94%. Найбільш впливовим фактором у моделі виявилася медіанна ціна за квадратний фут. Такий результат пояснюється тим, що Random Forest краще адаптується до складних умов ринку й може враховувати нелінійні залежності між ознаками, що важко реалізувати за допомогою простої лінійної моделі. У випадках, коли дані є неоднорідними, це справді дає перевагу більш гнучким підходам.

Автори в дослідженні Case Study of Prishtina's Real Estate Market [4] для прогнозування цін на житло порівнювали результати моеделей машинного навчання, серед яких були лінійна регресія, дерева рішень, k-найближчих сусідів і регресія опорних векторів. Використовувався набір даних на основі транзакцій нерухомості в Косово. Найкраще себе показала модель дерев рішень із із найнижчою середньоквадратичною помилкою (RMSE) та найвищим коефіцієнтом детермінації (R^2).

Стаття Prediction and Analysis of Rental Price using Random Forest Machine Learning Technique[5] відображає дослідження цін на оренду житла в Шанхаї та Ухані (Китай) за допомогою моделі Random Forest. Дані розміром у близько 2500 об'єктів для кожного з міст були взяті з сайту для пошуку оренди з характеристиками, такими як площа, розташування, наявність ліфта, оздоблення та кількість кімнат. Найкращі результати були для Уханю з найвищим коефіцієнтом детермінації в R^2 0.9157, втім для Шанахаю модель була не настільки точною з показником R^2 0.8343. Причиною цього, ймовірно, стала складна ситуація на ринку та недостатня кількість важливих факторів у моделі. В Ухані найбільше на ціну впливали район площа квартири та наявність ліфта. У Шанхаї основними факторами були площа та розташування. Це дослідження показує, як важливо правильно підбирати змінні для моделі, і може допомогти як орендодавцям, так і орендарям краще орієнтуватися в цінах.

У статті «House price prediction modeling using machine learning techniques: a comparative study» [6] (2023) порівнюються алгоритми машинного навчання для прогнозування цін на житло. Аналізували лінійну регресію, Random Forest, Gradient Boosting і SVR на основі даних про площу, кількість кімнат, розташування та інфраструктуру. Найточнішими виявилися Random Forest і Gradient Boosting, які краще враховують складні та нелінійні залежності. Лінійна регресія була менш ефективною, а SVR вимагав точного налаштування. Ключовими факторами впливу стали площа, розташування та близькість до інфраструктури.

Також слід звернути увагу на дослідженння «Real Estate Valuation Decision-Making System Using Machine Learning and Geospatial Data» у рамках автоматизованого підходу оцінки нерухомості [7]. Автор розглядає прогнозування вартості об'єктів житлової нерухомості в Німеччині у період 2024–2025 років із використанням виключно геопросторових змінних, без урахування економічних характеристик або технічного стану об'єктів. У дослідженні використано дані з німецького онлайн-ресурсу оголошень про нерухомість Immowelt, що охоплюють 233 міста. До набору ознак увійшли такі географічні характеристики, як:

- відстань до аеропорту,
- відстань до центру міста,
- міський рейтинг,
- наближеність до об'єктів інфраструктури (зупинок, шкіл тощо).

Результати дослідження продемонстрували:

- Точність прогнозу для XGBoost щодо цінових діапазонів — 62,63%;
- Середню абсолютну похибку (MAE) та середню абсолютну девіацію (MAD), масштабовану до 319 умовних одиниць;
- Найбільш впливовими факторами виявились: місто розташування, чисельність населення та відстань до аеропорту.

Один із ключових висновків полягає в тому, що геопросторові дані (тобто інформація, прив'язана до місця розташування) можуть забезпечити суттєво вищу точність оцінки вартості нерухомості, порівняно з традиційними методами. Автор відзначає, що такі підходи сприяють зниженню фінансових ризиків, підвищують прозорість під час переговорів між продавцем і покупцем, а також допомагають виявляти зони підвищеного попиту. Попри позитивні результати, дослідження також виявило обмеження, пов'язані з неповнотою даних і високими вимогами до попередньої обробки інформації.

У статті "Прогнозування ціни на нерухомість з використанням алгоритмів машинного навчання" [8] розглядається підхід до оцінки вартості нерухомості з використанням сучасних інструментів машинного навчання. Основна мета -

побудова ефективних моделей, здатних точно передбачати ринкову ціну заміських об'єктів за доступними даними. Для цього було проаналізовано ринок житлової нерухомості в Україні. Виявлено, що для отримання прогнозів на ціну нерухомості за містом можна використовувати алгоритми машинного навчання. Розглянуто декілька моделей: множинної лінійної регресії, регуляризації, випадкового лісу та XGBoost. Після перевірки ефективності моделей на тестовому наборі даних та порівняння результатів між собою було визначено, що за показниками RMSE та MAPE алгоритм XGBoost забезпечує найбільш точний та ефективний метод. Було проведено порівняння між прогнозованими результатами алгоритму градієнтного бустингу і моделі множинної лінійної регресії, яка проводилась на одному випадковому об’єкту з набору даних. Таким чином, модель XGB краще розуміється на даних і з більшою ймовірністю точно прогнозує реальні ціни.

У статті узагальнено результати аналізу та тестування даних і виявлено, що модель XGBoost може певною мірою ефективно прогнозувати та аналізувати ціни на житло. Водночас, точність та ефективність моделювання цін на нерухомість можна ще більше підвищити, використовуючи більш досконалі методи машинного навчання у порівнянянні з традиційними методами.

Стаття "AI in real estate property valuation: Is it really a game-changer?" [9] досліджує, як штучний інтелект (ШІ) змінює підхід до оцінювання вартості нерухомості. Раніше оцінка залежала від людського аналізу, який часто був неточним і забирав багато часу через ручне опрацювання значної кількості інформації. ШІ кардинально покращує цей процес, роблячи його швидшим, точнішим і зрозумілішим. Він аналізує різноманітні дані, від історії угод до місцевих ринкових тенденцій . На основі цього він формує оцінку, яка відповідає реальній ринковій ціні.

У статті пояснюється, як ШІ працює з величезними масивами інформації, виявляє приховані закономірності та враховує ринкові зміни в реальному часі. Це допомагає позбутися суб’єктивності, властивої традиційним методам. До того ж ШІ економить час і кошти: оцінка, яка раніше могла тривати тижні, тепер

займає години, а потреба в повторних перевірках зменшується. Прозорість процесу також зростає, адже ШІ чітко показує, на основі чого зроблено висновки.

Далі розглядаються практичні приклади застосування засобів шутчного інтелекту: в іпотечному кредитуванні він прискорює схвалення позик, в інвестиціях допомагає знаходити перспективні об’єкти, в оцінці податків забезпечує справедливість, а в управлінні портфелями нерухомості підказує, як оптимізувати активи. Автор наголошує, що ШІ — це не просто тренд, а необхідний інструмент для точного оцінювання та раціональних рішень на ринку нерухомості.

У статті Machine Learning for Real Estate Valuation: A Critical Review [10] проведено критичний огляд моделей машинного навчання для оцінки нерухомості. Автори зазначають, що лінійна регресія та SVR часто поступаються Random Forest і XGBoost через нездатність обробляти нелінійні залежності та шум у даних. XGBoost і Gradient Boosting є ефективними, але їхня точність залежить від якості даних і правильного вибору ознак. Дослідження підкреслює обмеження багатьох моделей через недостатнє врахування геопросторових і соціоекономічних факторів, а також закликає до інтеграції ШІ для підвищення адаптивності моделей, що узгоджується з перспективами цієї роботи.

Таким чином, традиційні методи оцінки є суб’єктивними та трудомісткими через залежність від експертних оцінок і обмежену здатність обробляти великі дані. Лінійна регресія, хоч і проста, не справляється з нелінійними залежностями ринку нерухомості. Random Forest і дерева рішень показують вищу точність, але схильні до перенавчання за недостатньої кількості даних чи неправильно підібраних ознак. XGBoost вирізняється високою точністю завдяки адаптивному бустингу, однак потребує ретельної підготовки даних і налаштування параметрів. Регресія опорних векторів і k-найближчих сусідів менш ефективні через чутливість до шуму та складність налаштування. Геопросторові дані підвищують точність, але ігнорування економічних і технічних факторів знижує універсальність моделей. Штучний інтелект відкриває нові можливості, але його ефективність поки складно оцінити через брак конкретних даних. Методи

машинного навчання, зокрема XGBoost, переважають традиційні підходи, але потребують адаптації до локального ринку.

1.4 Аналіз сучасних інструментальних засобів для визначення ціни на нерухомість

Redfin - американська технологічна брокерська компанія з нерухомості, заснована в 2004 році в Сіетлі, що працює як онлайн-платформа для купівлі, продажу та оренди житла[11]. Компанія відома нижчими комісійними для продавців порівняно з традиційними агентствами. Redfin використовує ШІ та методи науки про дані для оцінки нерухомості, зокрема модель машинного навчання, яка аналізує понад 500 показників із точністю до 98% для будинків на ринку. AI-асистент Ask Redfin допомагає відповідати на запити про об’єкти, а ШІ також застосовується для персоналізованих рекомендацій і покращення фотографій лістингів. Ці технології оптимізують оцінку та підвищують ефективність роботи з клієнтами.

Zillow є провідною онлайн-платформою з ринку нерухомості та оренди житла в США[12].Вона надає користувачам доступ до великої бази даних про житло, включаючи ціни, історію угод, характеристики об’єктів та демографічну інформацію. Її застосунок Zestimateавтоматично розраховує орієнтовну ринкову вартість об’єкта, спираючись на публічні дані, історію продажів, ринкові тренди та машинне навчання. Сервіс також надає інформацію про школи, транспортну доступність, податкові ставки та потенційну динаміку цін, що робить його надзвичайно зручним для покупців та інвесторів.Цей додаток використовує поєднання кількох сучасних технологій: алгоритми машинного навчання, аналіз великих даних (Big Data) та моделі автоматичної оцінки AVM (Automated Valuation Model), щоб були описані вище.

DOM.RIA - одна з найпопулярніших платформ нерухомості в Україні, яка пропонує широкий спектр оголошень про житлову та комерційну нерухомість по всій країні[13]. Веб-сайт надає перевірені списки з фотографіями, описами та

геолокацією, що робить його надійним джерелом для покупців та орендарів. DOM.RIA також пропонує калькулятор оцінки нерухомості (рис.1.4.1), який допомагає оцінити ринкову вартість об'єкта нерухомості на основі даних користувача. Калькулятор враховує місце розташування, тип нерухомості, площу кількість кімнат і стан квартири. Він використовує історичні ринкові дані та тенденції для формування цінового діапазону (рис.1.4.2). Інструмент є дійсно корисним, як і для покупців та власникиків нерухомості, яку хочуть продати чи здати в оренду для справедливої оцінки ринкової вартості нерухомості.

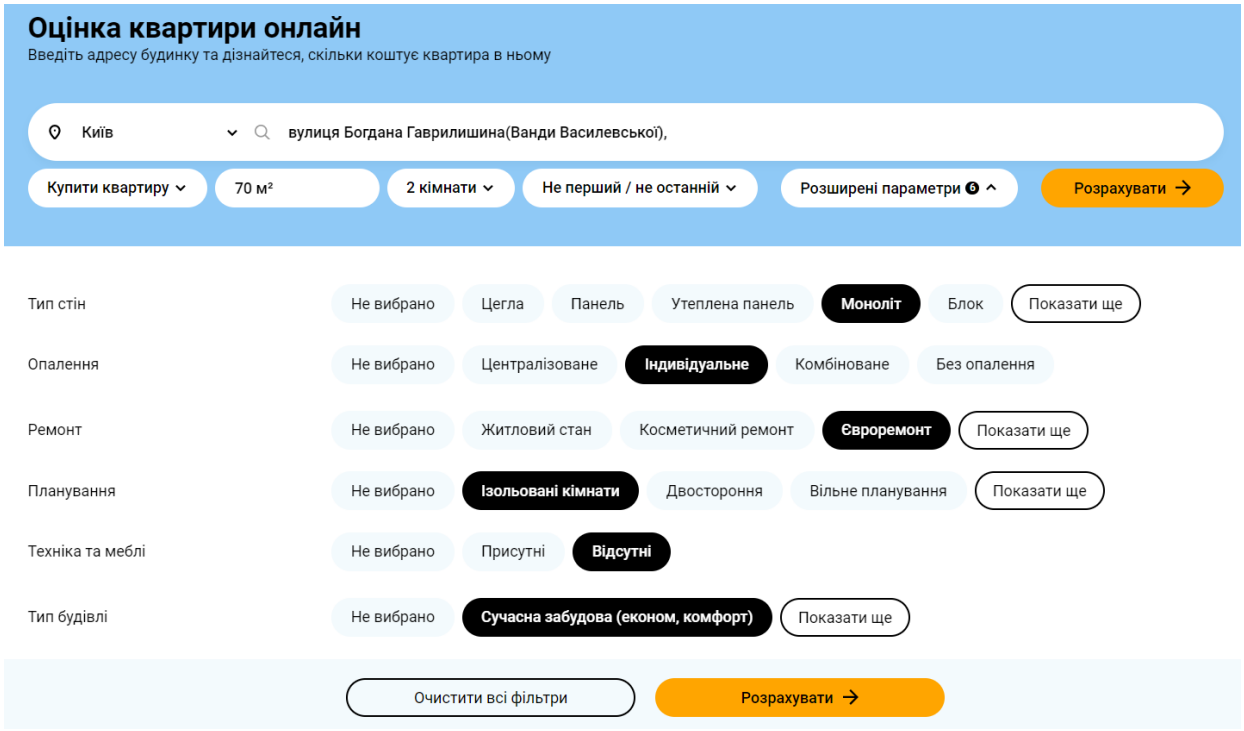


Рисунок 1.4.1 Оцінка квартири на DOM.ria

Рисунок 1.4.2 Результат оцінювання вартості квартири

OLX.ua — найбільша українська платформа онлайн-оголошень, заснована в 2006 році в складі міжнародної OLX Group[14]. Вона об’єднує користувачів для купівлі, продажу чи оренди товарів і послуг, зокрема в категорії нерухомості, де представлені квартири, будинки, земельні ділянки та комерційні об’єкти. Сервіс оцінки нерухомості OLX.ua, зокрема калькулятор цін на квартири, є інструментом для визначення ринкової вартості продажу чи оренди на основі аналізу актуальних оголошень (рис.1.4.3, рис.1.4.4).

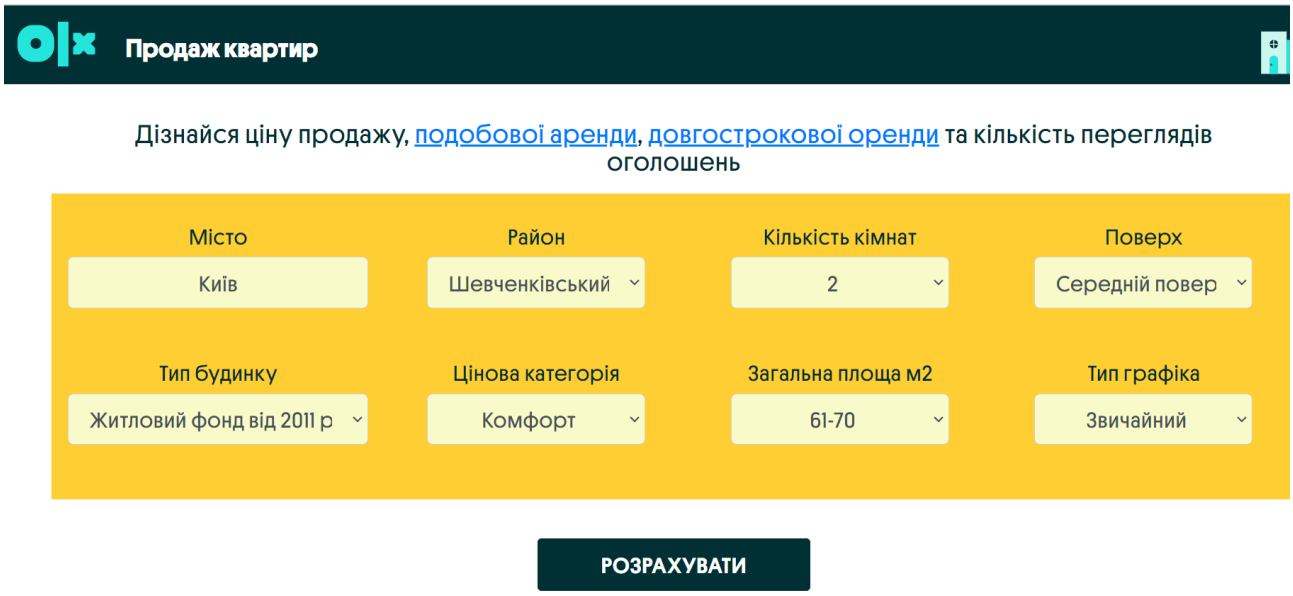


Рис 1.4.3 Оцінка квартири на OLX.ua

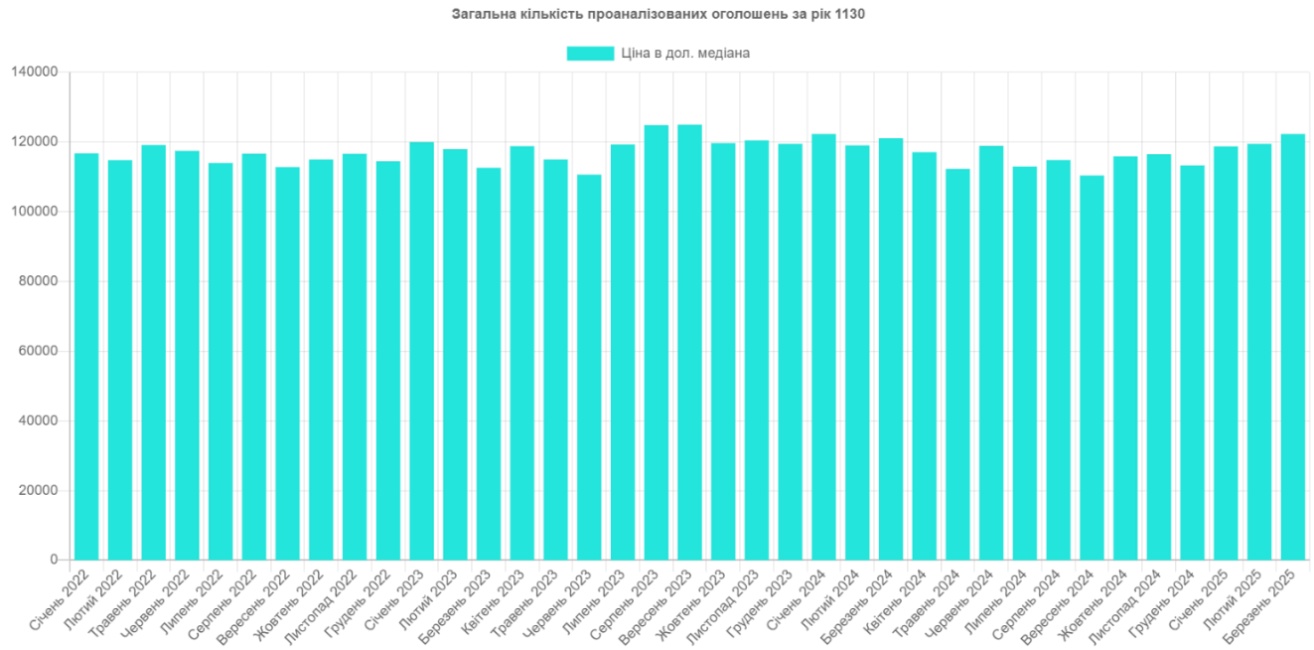


Рис 1.4.4 Результат оцінювання вартості квартири

Він враховує ключові параметри, такі як місце розташування, площа, кількість кімнат і стан об’єкта, що дозволяє користувачам швидко отримати орієнтовну ціну(рис.1.4.4). Цей сервіс є цінним для продавців і покупців, адже він спрощує процес оцінки, підвищує довіру до ціноутворення та сприяє прозорості ринку.

LUN.ua — це провідна українська онлайн-платформа для пошуку та аналізу нерухомості, яка агрегує оголошення з численних джерел, охоплюючи як первинний, так і вторинний ринок житла[15]. Користувачі можуть фільтрувати пропозиції за параметрами, переглядати інформацію про забудовників, інфраструктуру та транспортну доступність, а також використовувати інтерактивну карту для перегляду актуальних цін по районах і житлових комплексах(рис.1.4.5). Платформа активно впроваджує методи штучного інтелекту для автоматичного визначення характеристик об'єктів (наприклад, наявності ремонту), об'єднання дублікатів, класифікації оголошень, пошуку схожих варіантів та прогнозування цінових тенденцій. Окрім того, LUN.ua використовує геоаналітику для оцінки вартості залежно від локації та надає зручні візуалізації, що робить платформу корисною як для пересічних користувачів, так і для фахівців з аналізу ринку.

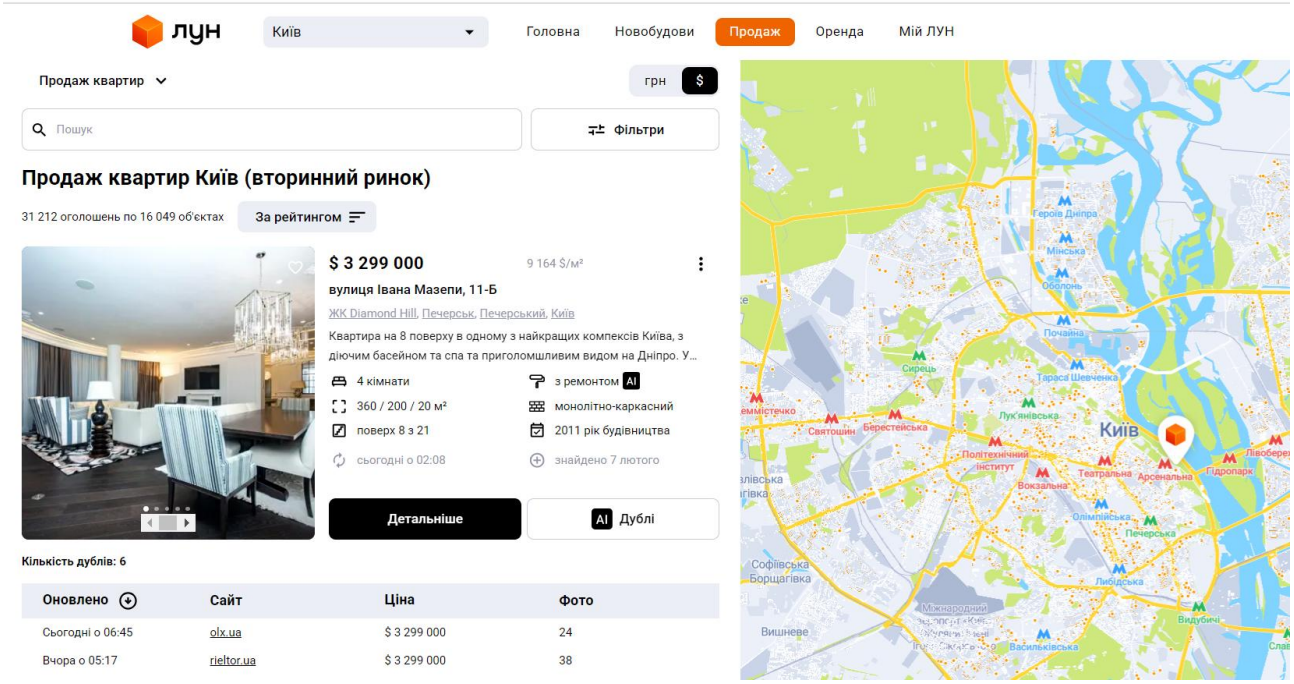


Рисунок 1.4.5 Інтерфейс ЛУН

У підсумку було проаналізовано деякі сучасні інструменти оцінки вартості нерухомості, які використовують ШІ, машинне навчання та аналіз великих даних для підвищення точності. Різні розглянуті платформи з нерухомості пропонують ефективні калькулятори та моделі, що враховують розташування, стан об’єкта й ринкові тенденції, забезпечуючи прозорість і зручність. Ці рішення допомагають користувачам швидко визначати конкурентну ціну, сприяють довірі на ринку та оптимізують процеси купівлі-продажу. Особливу увагу приділено інтеграції геоаналітики та прогнозуванню цін, що дозволяє адаптувати оцінку до локальних особливостей ринку. Такі інструменти зменшують ризик помилок у ціноутворенні та сприяють прийняттю обґрунтованих рішень.

1.5 Постановка задачі

Метою магістерської роботи є підвищення якості прогнозування вартості житлової нерухомості на вторинному ринку міста Києва шляхом розробки технології оцінки, що базується на методах машинного навчання, з подальшою інтеграцією у веб-додаток. Це дозволить підвищити точність і об’єктивність оцінок, автоматизувати процеси та сприяти ефективності ринкових операцій.

Для реалізації поставленої мети передбачено виконання таких етапів:

1. Збір даних про квартири з відкритих джерел, що включають характеристики об’єктів (площа, кількість кімнат, розташування, рік побудови, стан ремонту) та їхню ринкову вартість.
2. Передобробка даних: очищення від пропусків і викидів, кодування категоріальних ознак (One-Hot Encoding для низькокардинальних, Target Encoding для висококардинальних), масштабування числових ознак, створення нових ознак.
3. Проведення розвідувального аналізу для виявлення ключових факторів, що впливають на ціну, та формування інформативних ознак.
4. Розробка й оптимізація моделей машинного навчання із застосуванням крос-валідації для підбору гіперпараметрів.

5. Оцінка якості моделей за метриками для вибору найточнішої моделі.
6. Інтеграція оптимальної моделі у веб-додаток на базі Flask, який забезпечує зручний інтерфейс для введення даних про квартиру та отримання прогнозованої вартості.
7. Тестування системи на реальних прикладах, порівняння результатів із ринковими оцінками та оцінка можливостей практичного застосування в ріелторській діяльності, банківському кредитуванні чи оцінці заставного майна.

Розроблена технологія має забезпечити автоматизацію оцінки нерухомості, підвищити прозорість і ефективність ринкових процесів, а також стати основою для універсального інструменту для учасників ринку нерухомості.

Висновки

Аналіз теоретичних і методологічних основ оцінки вартості нерухомості показав, що це складне завдання, яке потребує інтеграції традиційних, статистичних і сучасних підходів науки про дані. Методологія CRISP-DM є ефективним інструментом, що структурує процес від розуміння бізнес-задачі до впровадження моделі, сприяючи автоматизації оцінки. На ціноутворення впливають численні фактори, зокрема місцезнаходження, технічні характеристики, інфраструктура, економічні та політичні умови, а також попит і пропозиція, причому геопросторові дані відіграють ключову роль у врахуванні локальних особливостей ринку.

Традиційні методи оцінки, такі як порівняння продажів, дохідний і витратний підходи, залишаються актуальними, але обмежені суб’єктивністю та залежністю від якості даних. Статистичні методи, зокрема множинна регресія та гедоністичний підхід, забезпечують вищу точність, але поступаються сучасним алгоритмам машинного навчання, таким як Random Forest і XGBoost, які ефективно обробляють великі обсяги даних і виявляють нелінійні залежності, демонструючи високу точність прогнозів.

Сучасні платформи, такі як Redfin, Zillow, DOM.RIA, OLX.ua та LUN.ua, використовують штучний інтелект, машинне навчання та геоаналітику для автоматизованої оцінки, підвищуючи прозорість ціноутворення, зручність для користувачів і довіру до ринку. Інтеграція калькуляторів цін і моделей автоматичної оцінки оптимізує процеси купівлі-продажу.

Застосування методів машинного навчання та геопросторових даних створює можливості для підвищення точності оцінки, зниження фінансових ризиків і забезпечення прозорості ринку. Аналіз підтверджує актуальність і практичну цінність науки про дані для оцінки нерухомості та необхідність розвитку автоматизованих систем для підвищення ефективності ринкових процесів.

РОЗДІЛ 2. ФОРМАЛІЗАЦІЯ МЕТОДІВ АНАЛІЗУ ДАНИХ У ПРОЄКТІ РОЗРОБКИ ТЕХНОЛОГІЇ ДЛЯ ОЦІНКИ ЖИТЛА

2.1 Математичний опис методу лінійної регресії

Лінійна регресія - це метод в статистичному аналізі, який використовується в різних моделях машинного навчання для прогнозування значення невідомих даних за допомогою інших значень пов'язаних даних. Цей метод застосовують для визначення зв'язку між залежною та незалежною змінними.

Лінійна регресія є дуже поширеною формулою, яка використовується в різних моделях машинного навчання, що виконують прогнозний аналіз. У лінійній регресії є дві змінні, і вони розглядаються як незалежна змінна і залежна змінна. Також припускається лінійний зв'язок між змінними, що означає, що зміни в незалежних змінних пов'язані з пропорційними змінами в залежній змінній.

Різні лінійні регресії, які зазвичай використовуються:

- Проста лінійна регресія. Це найпростіша форма, коли є одна змінна, яку необхідно передбачити, і одну змінну, яка попередньо може на неї вплинути. Наприклад, прогнозний аналіз, де передбачають вагу людини на основі її зросту.
- Множинна лінійна регресія. все ще прогнозується одна змінну, але тепер розглядається кілька факторів, які можуть на неї вплинути. Наприклад, передбачення ваги людини на основі її зросту, віку і, можливо, навіть її харчових звичок.
- Логістична регресія. Цей метод використовується, коли є справа з бінарними результатами. Ми все ще розглядаємо безліч факторів, які можуть відігравати певну роль.
- Порядкова регресія. Іноді те, що намагаються передбачити, не зовсім числове, але має певний порядок, як приклад оцінку чогось від 1 до 5 зірок. Цей вид регресії допомагає передбачити такі порядкові результати.

- Мультиноміальна регресія: Коли результат має кілька категорій, але не має впорядкованості.
- Дискримінантний аналіз: Подібно до багатофакторної регресії, він допомагає нам, коли є кілька категорій для змінної результату, але увага зосереджується на класифікації випадків за цими категоріями на основі предикторних змінних.

Формула лінії лінійної регресії записується має вигляд :

$$y = a + bx \qquad (2.1.1)$$

де *x* - незалежна змінна, побудована вздовж осі *X;*

y залежна змінна, побудована вздовж осі *Y.*

Нахил лінії регресії позначається як “b”, а значення перетину лінії регресії з віссю y (значення y, коли x = 0) позначається як “a”.

Значення перетину, a, та нахил прямої, b, обчислюються за допомогою формул, наведених нижче:

$$a = \frac{\sum y \sum x^2 - \sum x \sum xy}{n\sum x^2 - (\sum x)^2} \qquad (2.1.2)$$

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \qquad (2.1.3)$$

де *y* - залежна змінна розташовується вздовж осі *Y;*

a - Перетин з віссю *Y*;

b - нахил лінії регресії;

x - незалежна змінна розташовується вздовж осі *X*.

У лінії лінійної регресії, якщо визначено параметри регресії a_0 та a_1 , властивості наведено нижче:

- Лінія лінійної регресії мінімізує суму квадратичних різниць між спостережуваними та передбаченими значеннями.
- Лінія лінійної регресії завжди проходить через середні значення змінних *X* та *Y*.
- Константа лінійної регресії b_0 дорівнює *y*-перетину лінійної регресії.

- Коефіцієнт лінійної регресії b_1 є нахилом лінії регресії.

Метод найменших квадратів - це загальновідомий метод, на основі якого будуються лінії регресії на графіку *X*-*Y*. У цьому процесі визначається лінія найкращої відповідності, зменшуючи сумарне квадратичне відхилення значень по вертикалі від кожної точки даних до лінії.

Для будь-якої точки, яка точно відповідає лінії, її перпендикулярне відхилення дорівнює нулю. Лінія лінійної регресії показана на рисунку 2.1.1, доданому нижче:

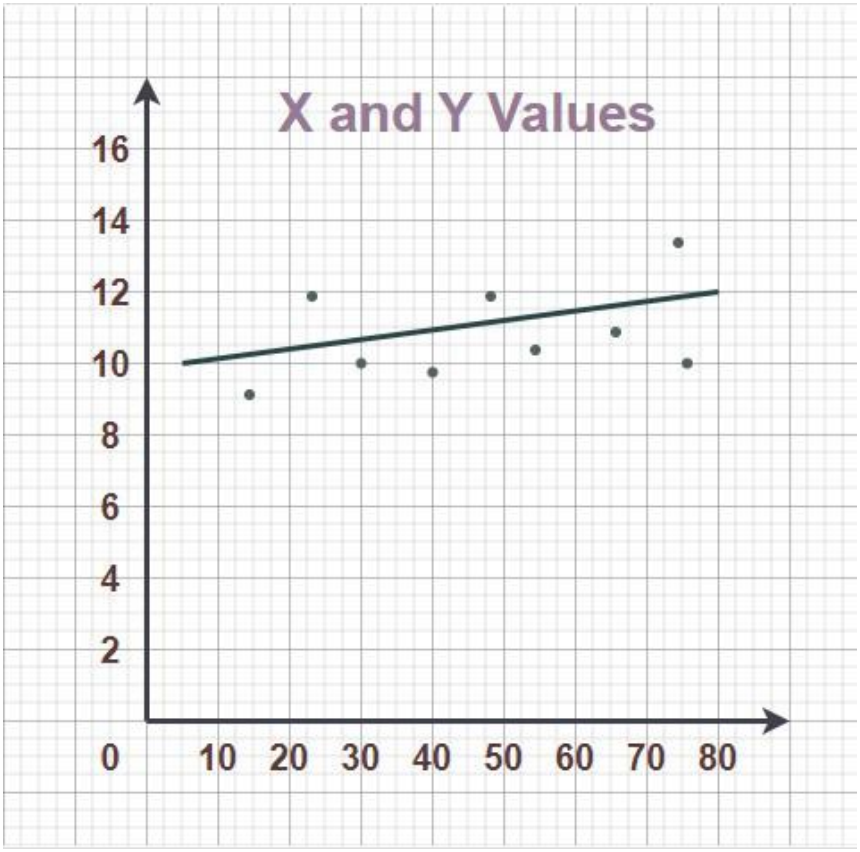


Рисунок 2.1.1 Лінія лінійної регресії[16]

Математичне представлення лінії лінійної регресії:

$$y = B_0 + B_1 X \ , \qquad (2.1.4)$$

де B_0 – це константа;

B_1 – коефіцієнт регресії.

Коефіцієнт регресії B_1 має такий вигляд:

$$B_1 = b_1 = \frac{\sum(x_i - x) - \sum(y_i - y)}{\sum(x_i - x)^2} \ , \qquad (2.1.5)$$

де $x_i$and y_i є наборами спостережуваних даних;

x and y є середніми значеннями.

Стандартна похибка лінії регресії визначається як міра середньої частки, яку прогнозує рівняння регресії. Стандартна помилка в цьому випадку позначається як «*SE*». Чим вищий коефіцієнт детермінації, тим нижча стандартна похибка і, відповідно, точніший результат.

Отже, лінійна регресія є важливим і широко використовуваним статистичним методом у прогнозному моделюванні та аналізі даних. Використовуючи формулу лінійної регресії та розуміючи її компоненти, такі як нахил, перехоплення та коефіцієнти регресії, можна ефективно моделювати зв'язок між незалежними та залежними змінними. Оволодіння формулою лінійної регресії дає можливість аналізувати тенденції даних, прогнозувати результати та отримувати значущі висновки, розширюючи можливості прийняття рішень у різних сферах.

2.2 Математичний опис методу Random Forest

Щоб зрозуміти принцип роботи методу випадкового лісу, спершу потрібно мати розуміння, як функціонує дерево рішень. Дерево рішень - це алгоритм з області машинного навчання, який використовує послідовність логічних умов для прийняття рішень. Воно будується у вигляді ієрархічної структури, де на кожному вузлі відбувається перевірка певної ознаки, а на основі результату обирається один з напрямків подальшого аналізу.

Основне завдання при побудові дерева - обрати, яку саме ознаку та порогове значення використовувати для розгалуження. Цей вибір базується на метриках якості, які оцінюють, наскільки добре поділ розділяє дані (наприклад, за допомогою ентропії або індексу Джині).

Процес триває до тих пір, поки не буде досягнуто певної умови зупинки: всі об'єкти у вузлі належать до одного класу, вичерпано допустиму глибину дерева, або кількість об'єктів у вузлі стає меншою за заданий поріг.Таким чином,

дерево рішень формує набір умов, що дозволяє класифікувати або прогнозувати значення для нових об'єктів.

Ентропія — це ключове поняття теорії інформації, яке ввів Клод Шеннон. Її можна уявити як міру невизначеності або хаотичності в системі: чим більш передбачуваним є стан системи, тим нижчий рівень ентропії. При побудові дерев рішень основна ідея полягає у зменшенні цієї невизначеності, тобто у впорядкуванні значень цільової змінної.

Формально ентропія для системи, що може перебувати в N різних станах, обчислюється за формулою Шеннона:

$$s = -\sum_{i=1}^{N} p_i \, log_2 p_i \, , \qquad (2.2.1)$$

де p_i— ймовірність того, що система перебуває в i-му стані.

Це поняття має зворотний зв’язок з інформацією: чим нижча ентропія, тим більше ми знаємо про систему. Тому вводиться поняття приросту інформації (Information Gain) як різниці між початковою ентропією та ентропією після поділу:

$$IG = S_0 - S_1, \qquad (2.2.2)$$

де S_0- ентропія до поділу, а S_1 - сумарна ентропія після розбиття вибірки. Крім приросту інформації, існують й інші метрики, що використовуються для вибору оптимального поділу у вузлах дерева:

- Індекс Джині (Gini impurity): вимірює ймовірність того, що випадково обраний елемент буде неправильно класифікований, якщо його класифікація базується на розподілі міток у підмножині. Він спрямований на формування максимально чистих піддерев.
- Помилка класифікації (misclassification error): оцінює частку неправильно класифікованих об’єктів. Проте, через низьку чутливість до змін, цей критерій використовується рідко.

У випадках регресії найчастіше застосовують дисперсію як метрику для поділу: ідея полягає в тому, щоб розбити вибірку так, аби всередині кожного піддерева варіація цільової змінної була мінімальною.

Побудова дерева реалізується як жадібний алгоритм, що приймає локально оптимальні рішення на кожному кроці. Алгоритм працює так:

1. Якщо всі елементи у вибірці U належать до одного класу — повертається листова вершина з цією міткою.
2. Інакше — обирається найкращий поділ за деяким критерієм, який розбиває U на дві частини: $U = U_1 \cup U_2$
3. Якщо хоча б одна з частин виявилась порожньою — створюється лист із міткою найпоширенішого класу в U.
4. В іншому разі формується внутрішня вершина, і рекурсивно викликається алгоритм для U_1 і U_2

Дерева рішень можуть дуже точно запам'ятовувати тренувальні дані, що призводить до перенавчання. Щоб цього уникнути, застосовують:

- Обмеження глибини дерева: наприклад, встановити максимальну глибину на рівні 5.
- Мінімальна кількість об'єктів у листі: якщо в листі, скажімо, 5 або менше прикладів, дерево більше не розгалужується.
- Ансамблеві методи, де багато дерев будуються на різних вибірках, а потім їх відповіді агрегуються.

Метод бутстреп (bootstrap), запропонований Бредлі Ефроном у 1977 році, базується на повторному випадковому виборі елементів з наявної вибірки з поверненням.

Нехай X — вибірка з N елементів. Створюється нова вибірка X_1, вибравши N разів елементи з X, кожного разу з однаковою ймовірністю $\frac{1}{N}$, дозволяючи повторення.

Повторюючи цей процес M разів, утворюється множина вибірок X_1,X_2,...,X_M. На кожній з них навчається свій класифікатор $a_i(x)$. Підсумкова модель $a(x)$. об'єднує їх відповіді, наприклад, методом голосування або усереднення:

$$a(x) = \frac{1}{M} \sum_{i=1}^{M} a_i(x) \qquad (2.2.3)$$

Такий підхід називається бегінгом (bagging — скорочення від *bootstrap aggregating*), і дозволяє зменшити дисперсію моделі та знизити ризик перенавчання[16].

Ідею методу випадкового лісу можна представити як бегінг серед дерев рішень та полягає в тому, щоб побудувати кілька дерев рішень, кожне з яких тренується на окремій підвибірці даних(рис.2.2.1).

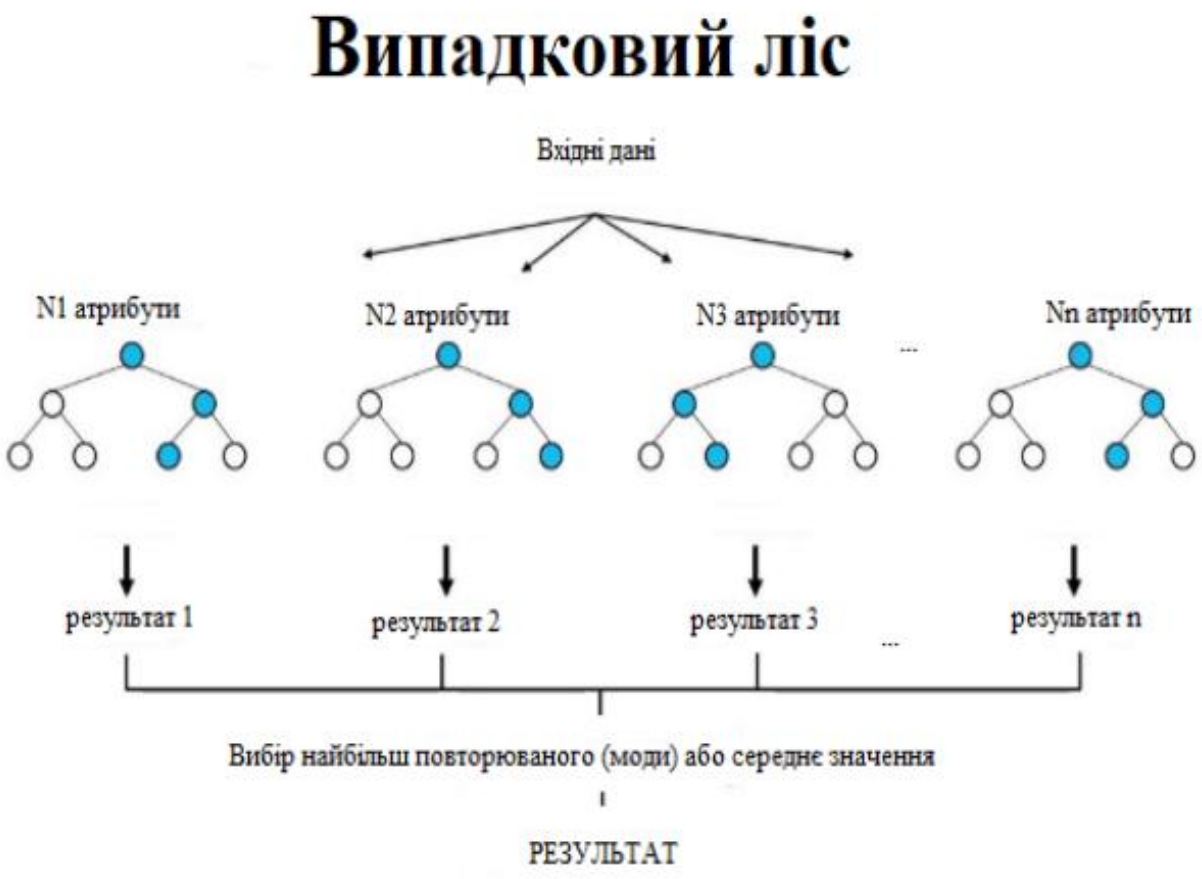


Рисунок 2.2.1 Алгортим випадкового лісу[18]

Нехай є навчальний набір даних X, що складається з l об'єктів, кожен з яких описується D ознаками. Обирається N кількість дерев у моделі N. Далі для кожного з N дерев:

- Формується нова навчальна підвибірка $\overline{X_n(n=1,N)}$ завдяки бутстрепу - тобто випадкового вибору об'єктів з повторенням.

- Для побудови дерева на цій підвибірці використовуються не всі D ознак, а випадкова підмножина з d ознак $(d < D)$. Значення d добирається з урахуванням задачі:
  - для класифікації зазвичай беруть $d \approx \sqrt{D}$
  - для задач регресії — $d \approx \frac{D}{3}$.

Фінальний прогноз моделі формується шляхом агрегування результатів усіх дерев:

- у класифікації за допомогою голосування (обирається клас, який частіше за все прогнозували дерева),
- у регресії ,як середнє або медіана прогнозів усіх дерев.

Переваги методу випадкового лісу:

- Забезпечує високу точність за рахунок об'єднання багатьох дерев у стабільну ансамблеву модель.
- Стійкий до викидів і шуму завдяки бутстреп-семплюванню.
- Не потребує нормалізації або стандартизації ознак.
- Працює однаково добре з числовими й категоріальними даними.
- Часто не вимагає складного налаштування параметрів.
- Може визначати, які ознаки найбільше впливають на результат.
- Добре масштабується та підтримує паралельну обробку.
- Має низький ризик перенавчання навіть при великій кількості дерев

Недоліки методу випадкового лісу:

- Результати важко інтерпретувати, особливо порівняно з одним деревом.
- Може працювати гірше на даних з великою кількістю розріджених ознак (наприклад, тексти).
- Не здатен робити екстраполяцію за межі наявних даних.
- Має схильність до переваги ознак з багатьма категоріями.
- Створює моделі великого розміру, що потребують багато пам’яті.
- Час навчання може бути довгим при великому обсязі даних.

- У порівнянні з простими моделями, складніше в реалізації й відлагодженні.

2.3 Математичний опис методу XGBoost

XGBoost (скорочена назва від eXtreme Gradient Boosting**)** - це вдосконалений варінт градієнтного бустингу і є типом методу ансамблевого навчання. Ансамблеве навчання поєднує кілька слабких моделей, щоб сформувати сильнішу модель.

XGBoost використовує дерева рішень як базові елементи навчання, поєднуючи їх послідовно для покращення продуктивності моделі. Кожне нове дерево навчається виправляти помилки, зроблені попереднім деревом, і цей процес називається бустингом. У цього методу є паралельна обробка для швидкого навчання моделей на великих наборах даних. Також є можливість налаштування гіперпараметрів моделі задля оптимізації продуктивності на основі конкретної проблеми.

Ця модель працює за принципом послідовного створення дерева рішень, в якому кожне дерево має спроби виправити помилки, отримані попереднім. Цей процес складається з таких етапів:

1. Початок з базового учня: перша модель дерева рішень навчається на даних, у задачах регресії базова модель просто передбачає середнє значення цільової змінної.
2. Обчислення помилок: після навчання першого дерева обчислюються помилки між прогнозованими та фактичними значеннями.
3. Навчання наступного дерева : наступне дерево навчається на помилках попереднього дерева. Цей крок намагається враховувати неточності, зроблені першим деревом.
4. Повторення процесу: цей процес продовжується з кожним новим деревом, намагаючись виправити помилки попередніх дерев, доки не буде виконано критерій зупинки.

5. Об’єднання прогнозів : Остаточний прогноз є сумою прогнозів усіх дерев.

Метод можна представити як ітеративний процес, який має початковий прогноз, який часто прирівнбється до нуля. Після цього кожне дерево додається для зменшення помилок. Математично модель можна представити так:

$$\hat{y}_i(x) = \sum_{i=1}^{n} f_k(x_i), \quad (2.1.1)$$

де $\hat{y}_i$ – кінцеве кінцеве прогнозоване значення для i-ї точки даних, K – кількість дерев в ансамблі та $f_k(x_i)$ являє собою передбачення K-го дерева для i -ї точки даних.

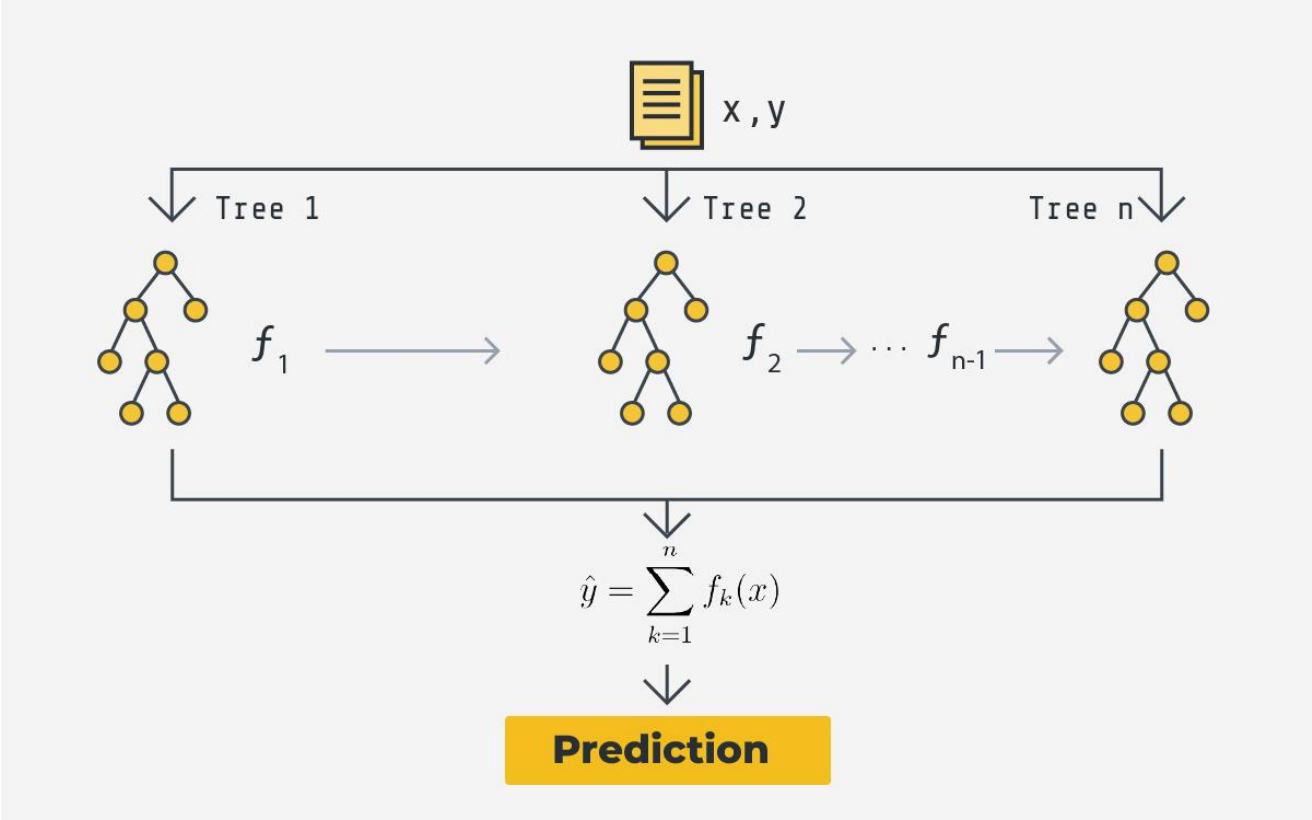


Рисунок 2.1 Математична модель XGboost[19]

Цільова функція в XGBoost складається з двох частин: функції втрат і члена регуляризації . Функція втрат вимірює, наскільки добре модель відповідає даним, а термін регуляризації спрощує складні дерева. Загальний вигляд функції втрат:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i), + \Sigma_{k=1}^{k} \Omega(f_k), \quad (2.1.2)$$

де $l(y_i, \hat{y}_i)$ це функція втрат, яка обчислює різницю між справжнім значенням y_i та прогнозованим значенням $\hat{y}_i$,

$\Omega(f_k)$ – це регуляризація, яка перешкоджає надто складним деревам.

Тепер замість того, щоб адаптувати модель відразу, модель оптимізовано ітераційно. З початкового прогнозу $\hat{y}_i^{(0)} = 0$ на кожному кроці додається нове дерево для покращення моделі. Оновлені прогнози після додавання *t*-го дерева можна записати так:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i), \tag{2.1.3}$$

де $\hat{y}_i^{(t-1)}$ є прогнозом з попередньої ітерації та $f_t(x_i)$ є прогнозом *t*-го дерева для і- точки даних.

Термін регуляризації спрощує $\Omega(f_t)$ складні дерева, обмежуючи кількість листків у дереві та розмір листків. Він визначається наступним чином:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum T_{j=1}\omega_j^2, \tag{2.1.4}$$

де *T* - це кількість листя на дерев;

γ є параметром регуляризації, який контролює складність дерева;

λ це параметр, який обмежує вагу листків ω_j^2.

Нарешті, вирішуючи, як розбити вершини дерева, обчислюється інформаційний приріст для кожного можливого розбиття. інформаційний приріст для розбиття розраховується так:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma, \tag{2.1.5}$$

де G_L, G_R — суми градієнтів у лівому та правому дочірніх вузлах;

H_L, H_R — суми матриці других частинних похідних у лівому та правому дочірніх вузлах.

Обчислюючи приріст інформації для кожного можливого розбиття на кожному вузлі, XGB вибирає розбиття, яке призводить до найбільшого приросту,

що ефективно зменшує помилки та покращує продуктивність моделі. XGBoost є потужним розширенням традиційного градієнтного бустингу, що робить його “екстремальним”. Його ключова відмінність полягає у введенні регуляризації безпосередньо в цільову функцію, яка має значний вплив на процес навчання моделі, сприяючи кращому узагальненню та запобігає перенавчанню.

Екстремальний градієнтний бустинг будує дерева рівень за рівнем, оцінюючи на кожному рівні, чи покращує додавання нового вузла (розбиття) цільову функцію в цілому. Якщо ні, розбиття обрізається. Таке рівневе зростання разом з обрізанням робить дерева легшими для розуміння та простішими для створення.

XGBoost оцінює кожне розбиття, яке можна зробити для кожної ознаки на кожному рівні, і вибирає те, яке мінімізує цільову функцію настільки, наскільки це можливо (наприклад, мінімізує середньоквадратичну помилку для задач регресії або перехресну ентропію для задач класифікації. На противагу цьому, для розбиття на кожному рівні вибирається одна ознака при розширенні в глибину. Витрати, пов'язані з вибором найкращого розбиття для кожної ознаки на кожному рівні, зменшуються з ростом рівня. Градієнтний бустинг усуває необхідність переглядати та оцінювати один і той самий об'єкт більше одного разу під час побудови дерева, оскільки всі об'єкти беруться до уваги одночасно. Це особливо корисно у випадку складних взаємозв'язків між елементами, оскільки алгоритм може адаптуватися до тонкощів даних.

Переваги екстремального градієнтного бустингу:

- зберігає продуктивність та ефективність навіть за великої кількості даних, оскільки він призначений для обробки великих наборів даних з мільйонами або навіть мільярдами екземплярів та об'єктів;
- реалізує методи паралельної обробки та використовує апаратну оптимізацію, таку як прискорення графічного процесора, для прискорення процесу навчання. Масштабованість та ефективність роблять цього алгоритму придатним для роботи з великими обсягами даних та прогнозування в реальному часі;

- він надає широкий спектр налаштовуваних параметрів і методів регуляризації, що надає змогу користувачам точно підбирати параметри моделі відповідно до свої цілей. Швидкість навчання, відома як коефіцієнт стискання, є новим параметром, представленим XGB. Він кількісно визначає внесок кожного дерева в загальний прогноз. Оскільки кожне дерево має менший вплив, процес оптимізації з нижчою швидкістю навчання є більш стійким. Роблячи модель більш консервативною, терміни регулярізації в поєднанні з низькою швидкістю навчання допомагають уникнути перенавчання;
- пропонує вбудований аналіз важливості функцій, який допомагає визначити найбільш впливові функції в наборі даних. Ця інформація може бути цінною для відбору функцій, зменшення розмірності та отримання уявлення про основні закономірності даних;
- добре працює навіть з неповними наборами даних завдяки потужному механізму обробки пропущених даних під час навчання. Для ефективної обробки відсутніх значень використовує алгоритм, що розглядає відсутні значення як окреме значення і оцінює потенційні розбиття відповідно до них при визначенні оптимального розбиття в кожному вузлі. Якщо під час побудови дерева в точці даних відсутнє значення певної ознаки, вона спускається вниз по іншій гілці дерева.

Недоліки методу:

- може вимагати значних обчислень, особливо при навчанні складних моделей, що робить його менш придатним для систем з обмеженими ресурсами;
- незважаючи на свою надійність, може бути чутливим до зашумлених даних або викидів, що вимагає ретельної попередньої обробки даних для оптимальної продуктивності;
- схильний до надмірного перенавчання на невеликих наборах даних або коли в моделі використовується занадто багато дерев;

- незважаючи на наявність оцінок важливості ознак, загальну модель може бути складно інтерпретувати порівняно з не такими примітивними методами, наприклад як лінійна регресія або дерева рішень. Така недостатня прозорість може бути недоліком у таких галузях, як охорона здоров'я або фінанси, де інтерпретованість є критично важливою.

2.4 Порівнняня моделей

Лінійна регресія, випадковий ліс та XGBoost - широко використовувані алгоритми машинного навчання, кожен з яких має свої особливості, що підходять для різних типів задач. У той час як Random Forest і XGBoost використовують ансамбль методів для надійних прогнозів, лінійна регресія пропонує простіший і зрозуміліший підхід для моделювання лінійних залежностей.

Лінійна регресія - це параметрична модель, яка припускає лінійний зв'язок між вхідними характеристиками і цільовою змінною, оцінюючи коефіцієнти для мінімізації суми квадратів помилок. Вона є простою, але обмежується відображенням лінійних закономірностей. Random Forest, ансамблевий метод, використовує бегінс для побудови декількох дерев рішень незалежно, кожне з яких навчається на випадкових підмножинах даних та ознак, і об'єднує їхні прогнози для покращення стабільності. XGBoost, ще один ансамблевий метод, використовує градієнтний бустинг, послідовно будуючи дерева, де кожне з них виправляє помилки попередніх, оптимізуючи функцію втрат за допомогою градієнтного спуску для більшої точності.

Лінійна регресія схильна до надмірного перенавчання, коли має справу з даними високої розмірності або зашумленими даними, особливо без регуляризації. Алгортим випадкового лісу зменшує надмірне перенавчання, усереднюючи прогнози з різних дерев, при цьому випадковість у виборі ознак і даних покращує узагальнення. XGBoost включає вбудовану регуляризацію L1 і

L2, а також такі параметри, як max_depth і min_child_weight, для контролю складності моделі, що робить його високоефективним для запобігання перенавчання, особливо в складних наборах даних.

Лінійна регресія є обчислювально ефективною, з швидким часом навчання та прогнозування, але її продуктивність страждає, коли дані відхиляються від лінійних припущень. Random Forest може бути повільним у навчанні на великих наборах даних через побудову численних незалежних дерев, хоча прогнозування є відносно швидким. XGBoost оптимізовано для швидкості та масштабованості, використовуючи паралельну обробку та ефективне використання пам'яті, часто перевершуючи Random Forest як за швидкістю навчання, так і за точністю прогнозування, особливо на структурованих даних.

Головні відмінності представлено в таблиці:

Таблиця 2.1.1 – Відмінності методів за ознаками

Ознака	Лінійна регресія	Випадковий ліс	XGBoost
Створення моделі	Підбирає лінійне лінійне рівняння для мінімізації середньоквадратичної похибки.	Будує незалежні дерева рішень з використанням бегінгу.	Послідовно будує дерева, виправляючи помилки за допомогою градієнтного підсилення.
Оптимізація	Аналітично мінімізує суму квадратичних похибок	Усереднює прогнози з декількох дерев.	Мінімізує функцію втрат ітеративно, використовуючи градієнтний спуск.
Перенавчання	Схильна до перенавчання без регуляризації.	Зменшує перенавчання завдяки усередненню та випадковості.	Використовує регуляризацію та обмеження на дерева.

Продовження таблиці 2.1.1

Ознака	Лінійна регресія	Випадковий ліс	XGBoost
Робота з нерівномірними наборами даних	Обмежена; потребує попередньої обробки.	може вимагати методів вирівнювання.	ефективно справляється з нерівномірністю.
Налаштування	Потребує мінімального налаштування;	Переважно налаштування кількості дерев та глибини	Вимагає налаштування багатьох параметрів
Пристосованість до розподілених обчислень	Обмежена; зазвичай однопотокова	Підтримує паралельну побудову дерев.	Добре оптимізовано для розподілених систем.
Обробка великих наборів даних	Швидка, але обмежена вимогами до лінійних залежностей.	Навчання сповільнюється на великих обсягах даних.	Ефективна та масштабована для великих наборів даних.
Точність прогнозу	Добре підходить для лінійних даних; неефективна для складних моделей.	Стійкий до помилок, але не завжди забезпечує найвищу точність.	Високоефективний, особливо для складних задач.
Інтерпретованість	Коефіцієнти з високим рівнем інтерпретації.	Помірний рівень інтерпретації; доступна оцінка важливості ознак.	Низький рівень інтерпретації; вимагає використання додаткових інструментів.

2.5 Мова програмування, основні бібліотеки та інструменти

Мовою програмування для побудови моделей оцінки у сфері нерухомості було обрано Python. Python - це мова програмування високого рівня, яка підтримує об'єктно-орієнтований підхід і працює в інтерпретованому режимі. Вона має динамічну семантику, що робить її гнучкою та зручною для швидкої розробки програм. Мова програмування вирізняється високорівневими вбудованими структурами даних, які разом із динамічною типізацією та гнучким механізмом зв’язування забезпечують зручність для швидкої розробки програм. Завдяки простому та зрозумілому синтаксису, мова легко вивчається й сприяє кращій читабельності коду, що, в свою чергу, знижує витрати на його супровід. Python підтримує використання модулів і пакетів, що сприяє модульному підходу до проєктування та повторному використанню коду. Інтерпретатор мови та обширна стандартна бібліотека доступні безкоштовно у вигляді вихідного коду або готових виконуваних файлів для більшості популярних операційних систем, що робить Python ще більш доступним для широкого кола користувачів.

Розробники часто обирають саме цю мову через її значну продуктивність, яку вона забезпечує. Оскільки немає етапу компіляції, прискорюється цикл редагування, тестування та налагодження. Python забезпечує простоту налагодження: помилки в коді чи некоректні дані не викликають критичних збоїв, таких як помилка сегментації. Натомість інтерпретатор генерує інформативні винятки з детальним трасуванням стеку, що полегшує пошук і виправлення проблем. Вбудований налагоджувач, реалізований на Python, дозволяє інспектувати змінні, виконувати приклади, встановлювати точки переривання та покроково аналізувати код, демонструючи потужні інтроспективні можливості мови.

Для реалізації моделей оцінки нерухомості використано мову програмування Python у поєднанні з набором спеціалізованих бібліотек і інструментів, які забезпечують повний цикл обробки даних: від збору інформації, її аналізу та моделювання до візуалізації результатів і створення

зручного користувацького інтерфейсу. Ці інструменти враховують специфіку ринку нерухомості, дозволяючи створювати точні, масштабовані та практично застосовні рішення[20].

NumPy є основою для обробки числових даних, таких як ціни нерухомості, площі об’єктів, кількість кімнат чи відстань до інфраструктури. Бібліотека підтримує роботу з багатовимірними масивами та забезпечує швидкі математичні обчислення, що критично важливо для підготовки даних до моделювання. Pandas спрощує роботу з табличними даними, дозволяючи виконувати попередню обробку, включаючи очищення пропущених значень, видалення аномалій, фільтрацію та об’єднання наборів даних із різних джерел.

Scikit-learn застосовується для створення моделей машинного навчання, зокрема, в бібліотеці реалізовано алгоритми лінійної регресії та випадкових лісів. Бібліотека XGBoost, яка реалізує градієнтний бустинг, вирізняється високою точністю прогнозів завдяки ефективно оптимізованим обчислювальним процесам.Її можливості тонкого налаштування гіперпараметрів роблять її ефективною для роботи з великими та складними наборами ринкових даних.

Matplotlib і Seaborn відповідають за візуалізацію даних. Matplotlib дозволяє створювати гнучкі графіки та діаграми. Seaborn полегшує побудову естетичних статистичних візуалізацій, таких як теплові карти кореляцій чи графіки залежностей між параметрами, що допомагає виявляти ключові фактори, які впливають на вартість. Selenium WebDriver автоматизує збір даних із вебсайтів, взаємодіючи з динамічними сторінками для отримання актуальної інформації. Це забезпечує регулярне оновлення наборів даних для аналізу.

Також застосовувалися знання JavaScript, CSS і HTML для аналізу структури веб-сторінок і створення селекторів під час парсингу даних із сайту. JavaScript - мова програмування, що підтримує динамічну обробку вмісту веб-сторінок, CSS - стилі для визначення структури та відображення елементів, а HTML - мова розмітки для створення структури веб-сторінок, що разом дозволили ефективно витягувати дані з оголошень.

Flask - вебфреймворк,що застосовується для створення інтерактивного вебдодатку, де користувачі зможуть вводити дані та отримувати результати обробки. Jupyter Notebook забезпечує інтерактивне середовище для розробки, тестування та документування коду. Воно дозволяє поєднувати код, візуалізації та текстові пояснення в єдиному документі, що спрощує експерименти з даними, оцінку моделей і презентацію результатів.

Цей набір інструментів формує комплексний підхід до вирішення задачі оцінки нерухомості. Selenium забезпечує збір даних, NumPy і Pandas – їхню підготовку, Scikit-learn і XGBoost – побудову точних моделей, Matplotlib і Seaborn – наочне представлення результатів, а Flask і Jupyter Notebook – практичне використання та документування. Така інтеграція гарантує високу точність, масштабованість і зручність застосування розроблених моделей у реальних умовах ринку нерухомості.

Висновки

У цьому розділі представлено аналіз і математичне обґрунтування вибору моделей, що застосовують для розв’язання задачі прогнозування. Розглянуто три підходи: лінійну регресію, метод випадкового лісу та алгоритм градієнтного бустингу. Для кожної з моделей наведено формалізований опис алгоритмів, охарактеризовано їхню функціональність, обчислювальну складність та особливості реалізації. Було здійснено порівняння зазначених методів з огляду на їхню точність, інтерпретованость, стійкість до шумів і здатность до узагальнення.

Лінійна регресія відзначається простотою побудови та високим рівнем пояснювальної сили в умовах лінійних залежностей. Метод випадкового лісу продемонстрував ефективність у виявленні складних взаємозв’язків між змінними та в обробці нелінійних структур. XGBoost, у свою чергу, забезпечує високу прогностичну здатність завдяки використанню механізмів регуляризації, бустингу та ефективної оптимізації.

Окрему увагу приділено вибору інструментів реалізації. Як базовий засіб програмування використано мову Python, що надає великий спектр можливостей завдяки своїм бібліотекам для збору і обробки даних, побудови моделей, обробки даних та їх візуалізації. Програмне середовище Jupyter Notebook обране як зручна платформа для інтерактивної розробки, валідації результатів та документування процесу.

РОЗДІЛ 3. РОЗРОБКА МОДЕЛІ ПРОГНОЗУВАННЯ ВАРТОСТІ НЕРУХОМОСТІ МЕТОДАМИ НАУКИ ПРО ДАНІ

3.1 Збір та обробка даних

Дані для дослідження було зібрано з веб-сайту Lun.ua [15], який містить актуальні оголошення про продаж квартир у місті Київ. Збір даних здійснювався на мові Python з використанням бібліотеки Selenium WebDriver . Було створено скрипт у вигляді класу RieltorSpider, що автоматизує процес навігації по вебсайту, отримує дані із карток оголошень і зберігає їх у структурованому форматі. Основні етапи збору даних включають:

1. Ініціалізація вебдрайвера:
   - Через Selenium WebDriver було використано браузер Google Chrome для застосування драйвера.
   - Завдяки налаштуванню драйвера відбувався запуск браузера у повноекранному режимі при роботі зі сторнікою.
2. Навігація по сторінці:
   - Скрипт відкриває цільову сторінку LUN.ua та очікує появи карток оголошень (елементи з класом realty-preview), використовуючи механізм явного очікування (WebDriverWait) для забезпечення коректного завантаження динамічного контенту.
3. Парсинг даних із карток оголошень:
   - Картка з оголошеннями містила в собі таку інформацію: ціна, площа, кількість кімнат, поверх, адреса, назва житлового комплексу, мікрорайон, район, місто, рік будівництва та тип конструкції будинку, детальний опис квартири, дати останнього оновлення та створення оголошення (last_updated, created), URL зображення квартири (image_url).
   - CSS-селектори було застосовано для вилучення даних,приклад реалізації показано на рисунку 3.1.1.

```
# ID карточки
data['id'] = card.get_attribute('id') or "0"

# Ціна та ціна за м²
data['price'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-price--main').text.strip()
if card.find_elements(By.CSS_SELECTOR, '.realty-preview-price--main') else "0"
data['price_per_sqm'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-price--sqm').text.strip()
if card.find_elements(By.CSS_SELECTOR, '.realty-preview-price--sqm') else "0"
```

Рисунок 3.1.1 Використання CSS-селекторів

4. Обробка пагінації:
    - Використовуючи кнопку пагінації скрипт автоматично перенаправляє на наступні сторінки до тих пір поки кнопка стане не доступна.
    - Для кожного переходу перевіряється завантаження нової сторінки, а поточна сторінка відображається у консолі для моніторингу.
5. Уникнення дублювання:
    - Здійснюється перевірка однакових оголошень за унікальним ідентифікаторм в наборі даних для запобіганню задубльованих значень.
6. Збереження даних:
    - Зібрані дані зберігаються у структурованому вигляді у CSV-файлі apartments_data.csv за допомогою бібліотеки Pandas.

Отримано такий датасет завдяки парсингу інформації:

id	price	price_per_sqm	address	complex	microdistrict	district	rooms	area	floor	condition	constructio	year_	description
390198	$ 1 190 000	$ 5 409 за м²	Рильський провулок, 5/5	Старий Київ	Шевченківський	Київ	3 кімнати	220 / 150 / 30 м²	6 з 8	з ремонтом	цегляний будинок	1902	Рыльский пе
391137	$ 54 000	$ 1 588 за м²	вулиця Миколи Закревського, 101-А	ЖК Вигурівщина-20	Троєщина	Деснянський	1 кімната	34 / 16 / 9 м²	2 з 22	з ремонтом	панельні	2018	Продажа одн
392071	$ 419 900	$ 3 359 за м²	вулиця Іоанна Павла II, 12	ЖК Taryan Towers	Нижній Печерськ	Печерський	3 кімнати	125 / 60 / 25 м²	8 з 35	без ремонту	2017	0	Предлагаетс
392964	$ 69 000	$ 908 за м²	вулиця Олександра Махова, 4-Б	Південна Борщагівка	Святошинський	Київ	3 кімнати	76 / 42 / 8 м²	11 з 16	з ремонтом	утеплена панель	1982	Продаж: Без
393302	$ 156 000	$ 1 642 за м²	Павлівська вулиця, 18	Солдатська слобідка	Шевченківський	Київ	3 кімнати	95 / 70 / 12 м²	15 з 17	з ремонтом	цегляний будинок	2001	ІЦентр Киев
397699	$ 1 400 000	$ 6 863 за м²	вулиця Саперне Поле, 5	ЖК Бульвар Фонтанів	Нижній Печерськ	Печерський	4 кімнати	204 / 115 / 72 м²	11 з 11	з ремонтом	монолітно-каркасний	2019	Description ir
409362	$ 2 499 000	$ 8 061 за м²	вулиця Івана Мазепи, 11-Б	ЖК Diamond Hill	Печерськ	Печерський	4 кімнати	310 / 180 / 19 м²	7 з 20	з ремонтом	монолітно-каркасний	2011	Мазепи Івана
416790	$ 119 900	$ 1 071 за м²	вулиця Олександри Екстер, 9	Троєщина	Деснянський	Київ	4 кімнати	112 / 60 / 12 м²	2 з 22	з ремонтом	утеплена панель	2010	Предлагаетс

3.1.2 Отриманий датасет

Далі отриманий датасет треба обробити для коректного подальшого аналізу. Необхідно ціну всього, ціну за квадратний метр та кількість кімнат представити у числовому форматі. Поле з площею квартри розділити на три колонки: загальна площа, житлова та нежитлова площа, додати колонки з поверхом квартири і кількістю поверхів всього в будинку. Також через відсутність деяких параметрів при заповненні оголошення, деяка інформаця, як адреса, назва комплексу, мікрорайон, місто, рік будівництва та тип конструкції змістилась в сусідні стовпці, що потребує правильного розподілу.

id	address	complex	microdistrict	district	city	construction	year_built	price	price_per_sqm	rooms
39019	Рильський провулок, 5/5	без комплексу	Старий Київ	Шевченківський	Київ	цегляний будинок	1902	1 190 000	5 409	3
39113	вулиця Миколи Закревського, 101-А	ЖК Вигурівщина-20	Троєщина	Деснянський	Київ	панельні	2018	54 000	1 588	1
39207	вулиця Іоанна Павла II, 12	ЖК Taryan Towers	Нижній Печерськ	Печерський	Київ	монолітно-каркасний	2017	419 900	3 359	3
39296	вулиця Олександра Махова, 4-Б	без комплексу	Південна Борщагівка	Святошинський	Київ	утеплена панель	1982	69 000	908	3
39330	Павлівська вулиця, 18	без комплексу	Солдатська слобідка	Шевченківський	Київ	цегляний будинок	2001	156 000	1 642	3
39769	вулиця Саперне Поле, 5	ЖК Бульвар Фонтанів	Нижній Печерськ	Печерський	Київ	монолітно-каркасний	2019	1 400 000	6 863	4
40936	вулиця Івана Мазепи, 11-Б	ЖК Diamond Hill	Печерськ	Печерський	Київ	монолітно-каркасний	2011	2 499 000	8 061	4
41679	вулиця Олександри Екстер, 9	без комплексу	Троєщина	Деснянський	Київ	утеплена панель	2010	119 900	1 071	4
41769	Киянівський провулок	ЖК Liberty Estate	Татарка	Шевченківський	Київ	монолітно-каркасний	2013	2 400 000	5 783	4
41901	Оболонський проспект, 26	ЖК Obolon Residences	Оболонь	Оболонський	Київ	монолітно-каркасний	2020	240 000	2 637	2

3.1.3 Перша частина обробленого датасету

Для підвищення точності прогнозування цін до датасету додано географічні ознаки: довготу (longitude) і широту (latitude) квартири, відстань до найближчої станції метро (distance_to_metro, у кілометрах) та назву найближчої станції метро (nearest_metro). Ці дані отримано шляхом геокодування адрес завдяки бібліотеці geopy з використанням сервісу Nominatim. Для кожної адреси виконувався запит до Nominatim для отримання координат. На основі отриманих координат обчислювалася відстань до найближчої станції метро з попередньо визначеного списку станцій Київського метрополітену за допомогою методу geodesic з бібліотеки geopy.distance. Для стабільності запитів до Nominatim

використовувалася затримка 1,2 секунди між запитами та кешування результатів для уникнення повторних запитів для однакових адрес.

id	address	total_area	living_area	not_living_area	flat_floor	max_floor	distance_to_metro	nearest_metro	latitude	longitude
39019	Рильський провулок, 5/5	220	150	30	6	8	0,6346631165	Золоті ворота	50,4543735	30,5142446
39113	вулиця Миколи Закревського, 101-А	34	16	9	2	22	7,699924529	Лісова	50,5321023	30,6234497
39207	вулиця Іоанна Павла II, 12	125	60	25	8	35	0,8977601677	Палац Україна	50,4197996	30,5333534
39296	вулиця Олександра Махова, 4-Б	76	42	8	11	16	4,739384912	Святошин	50,415214	30,3971707
39330	Павлівська вулиця, 18	95	70	12	15	17	1,20752883	Лук'янівська	50,4528059	30,4901445
39769	вулиця Саперне Поле, 5	204	115	72	11	11	0,7480467148	Палац Україна	50,4189445	30,5309606
40936	вулиця Івана Мазепи, 11-Б	310	180	19	7	20	0,4268794089	Арсенальна	50,441572	30,5505622
41679	вулиця Олександри Екстер, 9	112	60	12	2	22	6,908080142	Лісова	50,5228417	30,6128129
41769	Киянівський провулок	415	125	82	5	5	1,014998918	Золоті ворота	50,4569616	30,5072183
41901	Оболонський проспект, 26	91	39	27	13	24	0,1505649176	Оболонь	50,5134219	30,5000854

3.1.4 Друга частина обробленого датасету

У результаті отримано датасет на 2377 рядків та 21 поле, опис яких подано нижче:

id: Унікальний ідентифікатор оголошення.

- price: Ціна квартири в гривнях.
- price_per_sqm: Ціна за квадратний метр у гривнях.
- address: Адреса квартири.
- complex: Назва житлового комплексу.
- microdistrict: Мікрорайон розташування квартири.
- district: Район міста.
- city: Місто розташування квартири.
- rooms: Кількість кімнат у квартирі.
- floor: Поверх розташування квартири.
- max_floor: Загальна кількість поверхів у будинку.
- condition: Стан квартири.
- construction_type: Тип конструкції будинку.

- year_built: Рік будівництва будинку.
- total_area: площа квартири всього в квадратних метрах.
- living_area: Житлова площа квартири в квадратних метрах.
- non_living_area: Нежитлова площа квартири в квадратних метрах.
- distance_to_metro: Відстань до найближчої станції метро в кілометрах.
- nearest_metro: Назва найближчої станції метро.
- latitude: Широта розташування квартири.
- longitude: Довгота розташування квартири**.**

3.2 Аналіз даних для моделі

Гістограма розподілу цін на квартири показує, як розподілені ціни на квартири у вибірці (рис.3.2.1). По осі X відкладено ціни квартир від 0 до 1.5 млн доларів, а на осі Y – кількість квартир. З графіку видно, що більшість квартир мають ціну від 50 до 100 тис. доларів США, і кількість оголошень різко зменшується після 200 тис. Доларів. Це вказує на те, що на ринку переважає середні і нижчий ціновий сегмент, а дорогі квартири зустрічаються рідко й утворюють довгий "хвіст" розподілу.

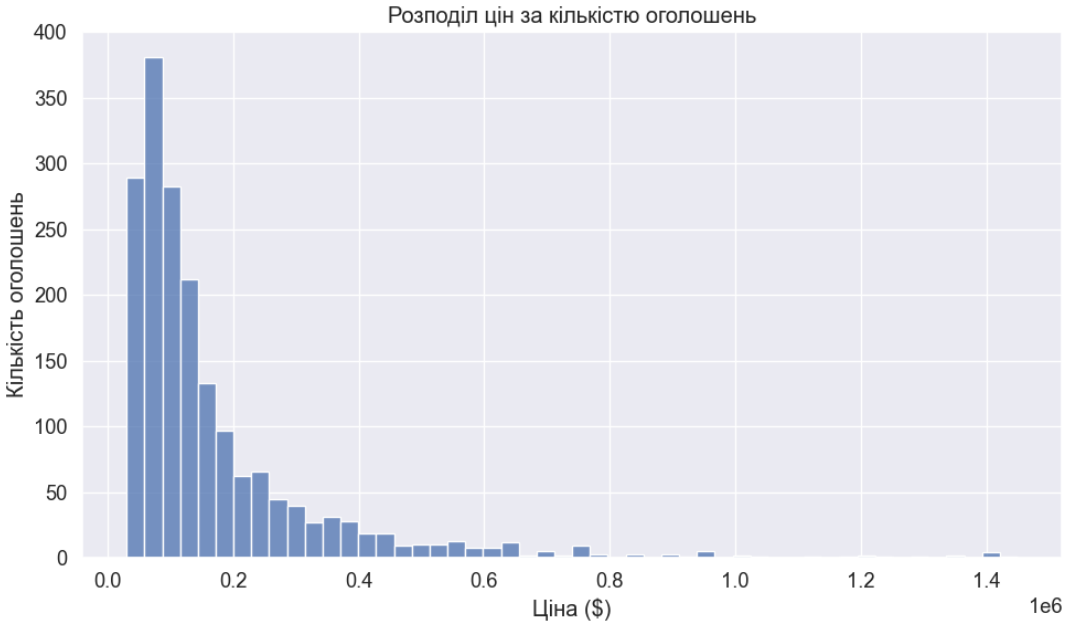


Рисунок 3.2.1 Розподіл цін за кількість оголошень

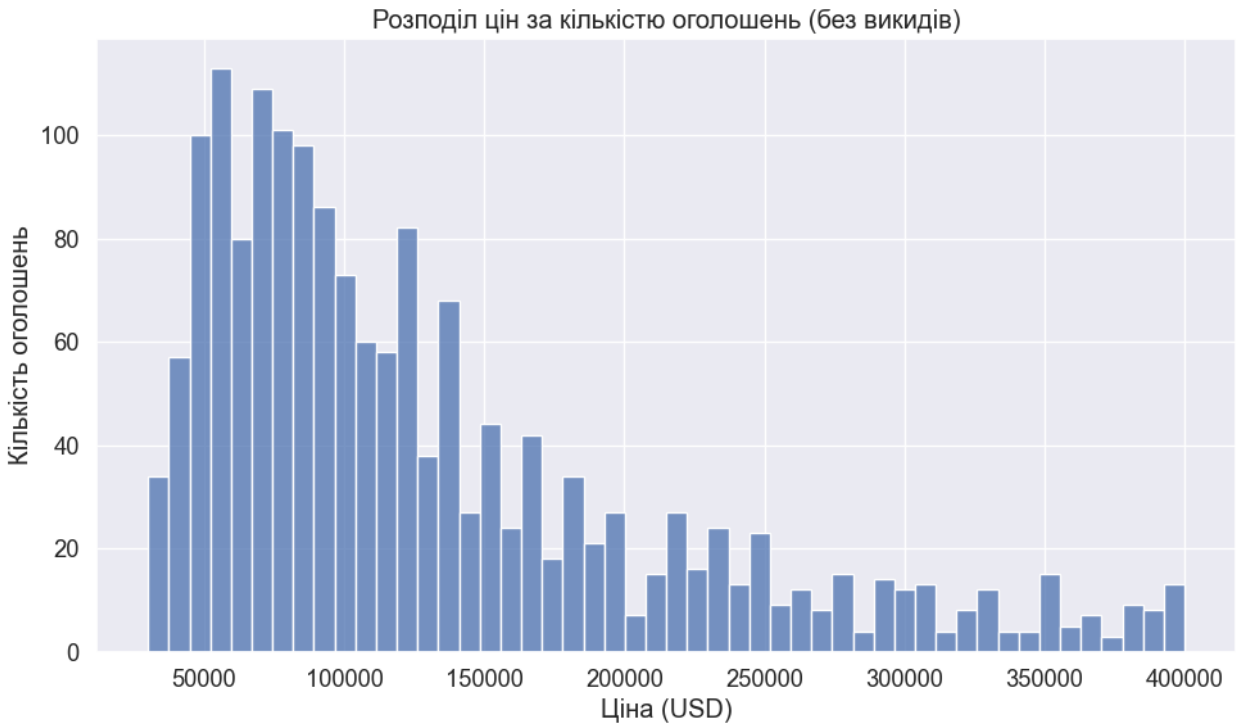


Рисунок 3.2.2 Розподіл цін за кількість оголошень

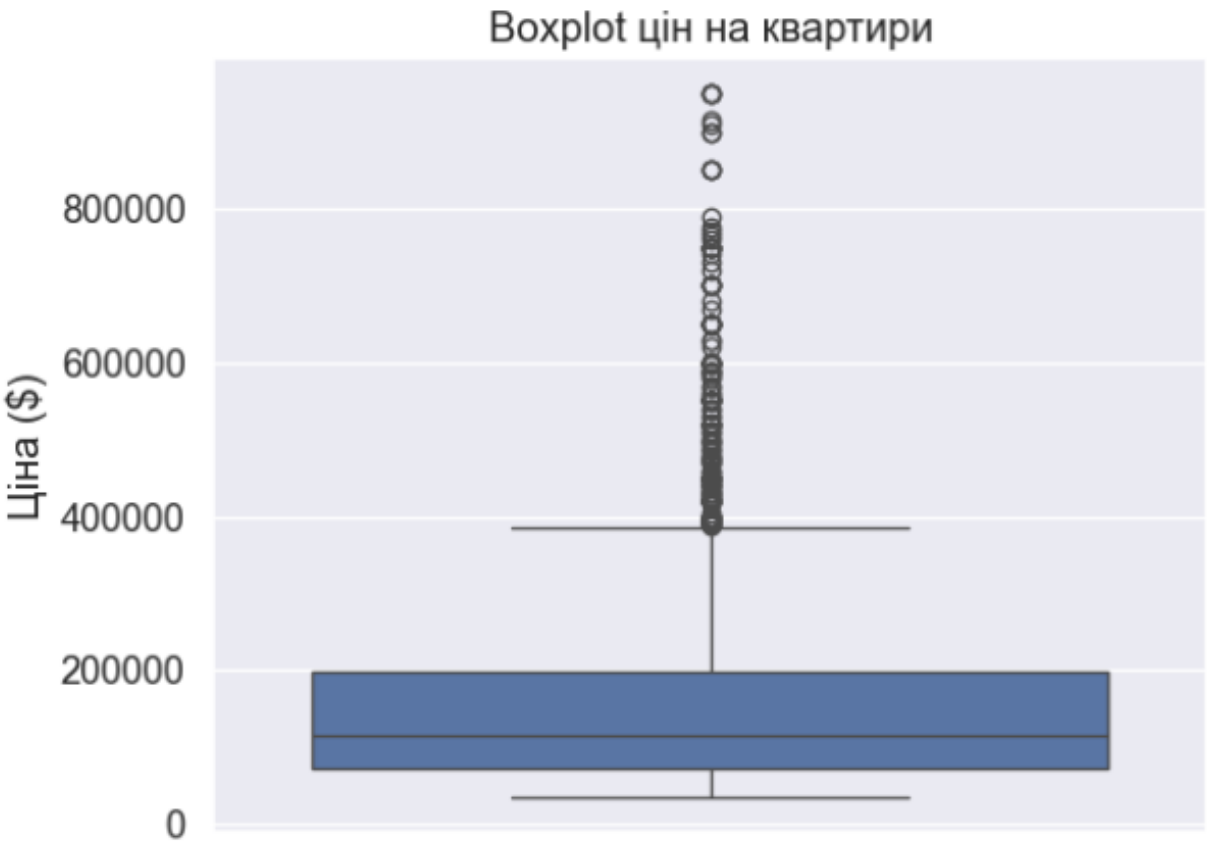


Рисунок 3.2.3 Боксплот цін на квартири

Боксплот цін на квартири показує основні характеристики розподілу цін. Медіана знаходиться приблизно на рівні 150 000 доларів, а 50% квартир мають

ціну в межах від 100 000 до 250 000. Боксплот також показує викиди: елітне житло із вартістю більше 40000 доларів. Основна частина квартир має вартість між 50 тис. і 400 тис. доларів. Як і на гістограмі, видно, що розподіл є скошеним вправо: більшість квартир мають помірну вартість, тоді як дуже дорогі об'єкти є рідкістю.

Кореляційна матриця числових ознак у сфері нерухомості показує (рис.3.2.4), що найбільший зв'язок із ціною мають загальна площа 0.75 і житлова площа 0.68. Сильна кореляція між total_area та living_area (0.89) свідчить про те, що збільшення загальної площі зазвичай супроводжується збільшенням житлової. Кількість кімнат також суттєво пов'язана із площею, проте має слабший прямий вплив на ціну - 0.52. Відстань до метро негативно корелює з ціною (-0.30), що логічно: чим ближче об'єкт до метро, тим вища його вартість. Інші змінні, такі як рік побудови чи поверх, мають помірний або слабкий вплив.

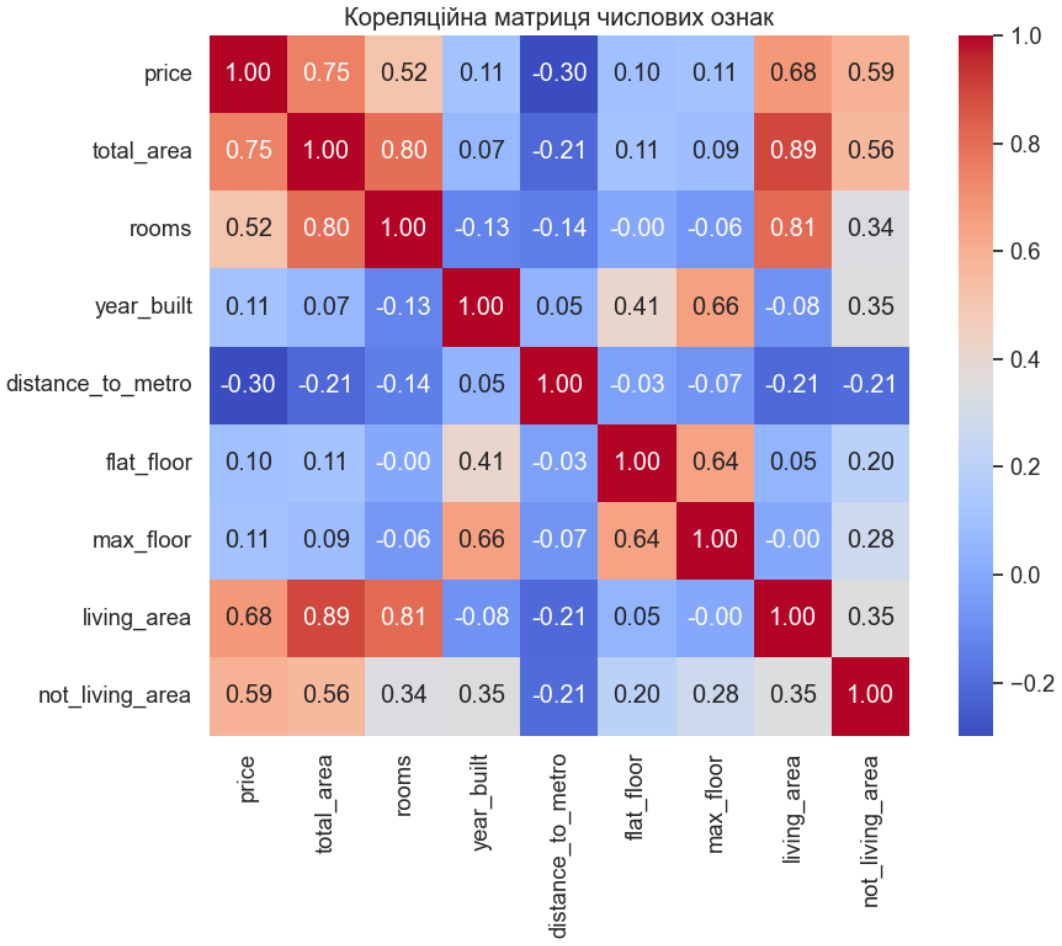


Рис. 3.2.4 Кореляційна матриця числових ознак

Карту було побудовано використовуючи бібліотеку contextily, яка дає змогу додавати фонові картографічні зображення на основі відкритих даних.На рисунку 3.2.5 зображено розподіл середньої ціни за квадратний метр на ринку нерухості в місті Києві. Карта чітко ілюструє, що найвищі ціни, починаючи від 2500 доларів і більше за квадратний метр зосереджені в районах центру міста, а саме Печерський та Шевченківський райони. Натомість нижчі ціни, до 2500 доларів переважають у спальних районах та на околицях, наприклад, у Деснянському чи Святошинському районах, що відображає типову тенденцію для великих міст.

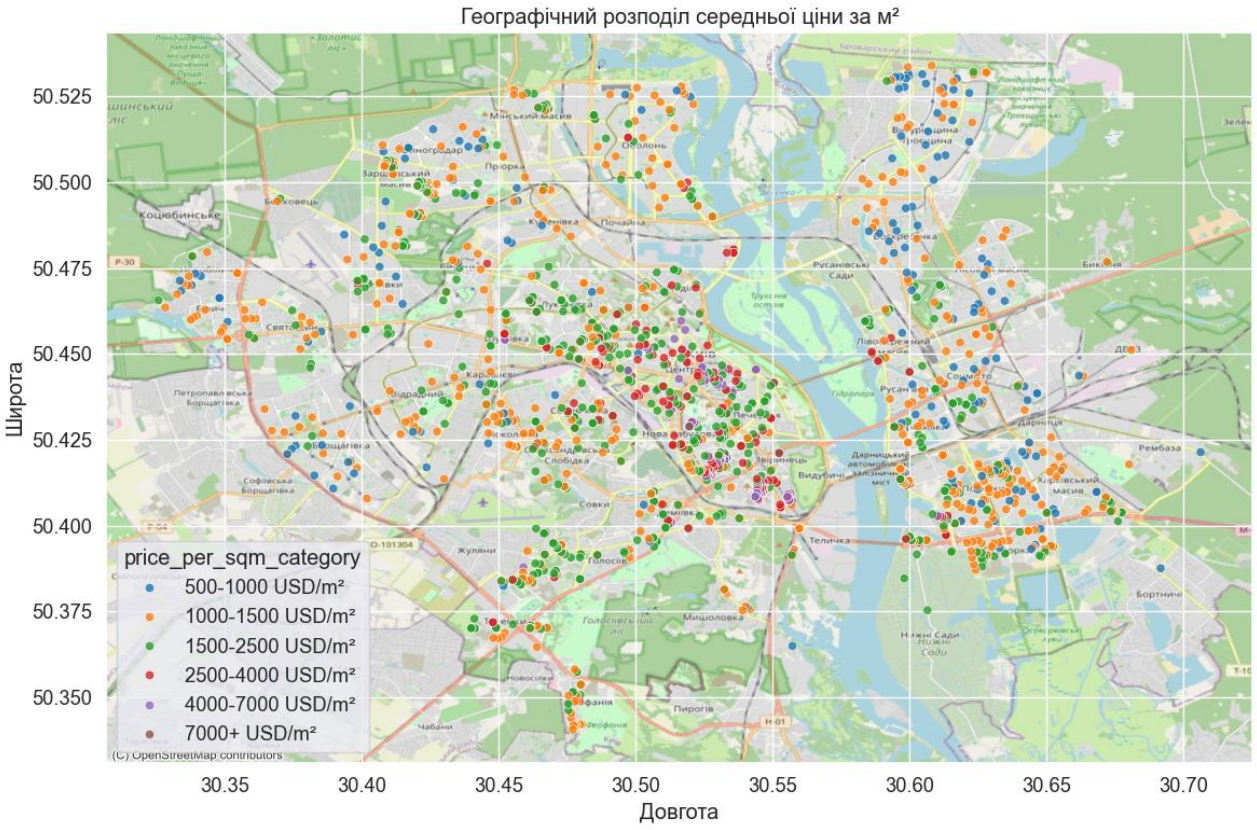


Рисунок 3.2.5 Географічний розподіл середньої ціни за квадратний метр

Далі представлено огляд на середні ціни за квадратний метр за кількістю кімнат, станом ремонту та районами (рис.3.2.6, рис. 3.2.7) . Дані з власної вибірки порівняно з результатами щоквартального звіту у сфері нерухомості на Лун за 1 квартал 2025 року станом на квітень (рис. 3.2.8) .

Порівняння даних з ваших графіків із статистикою ЛУН за квітень 2025 року показало високу відповідність основних тенденцій на ринку нерухомості

Києва. В обох джерелах найвищі ціни спостерігаються в центральних районах: Печерському та Шевченківському, а найнижчі у спальних районах, таких як Святошинський і Деснянський.

Виявлено цікаве спостереження: середня ціна за квадратний метр в однокімнатних квартирах вища, ніж у дво- та трикімнатних, що може бути пов'язано із вищим попитом на менші за площею об'єкти серед покупців.

Графіки також наочно демонструють різницю у вартості житла залежно від стану ремонту: квартири з ремонтом мають вищу середню ціну за квадратний метр на $100–$300 порівняно з об'єктами без ремонту. Це відповідає очікуваній ринковій логіці, оскільки готові до заселення квартири є більш привабливими для покупців.

Окрему увагу привертає Дарницький район, де у власних даних спостерігається деяке відхилення порівняно зі статистикою ЛУН. Це може бути пов'язано з тим, що дані збиралися у березні 2025 року, тоді як дані ЛУН відображають ситуацію за перший квартал 2025 року, або іншими факторами, такими як відмінності у складі вибірки чи короткострокові коливання ринку.

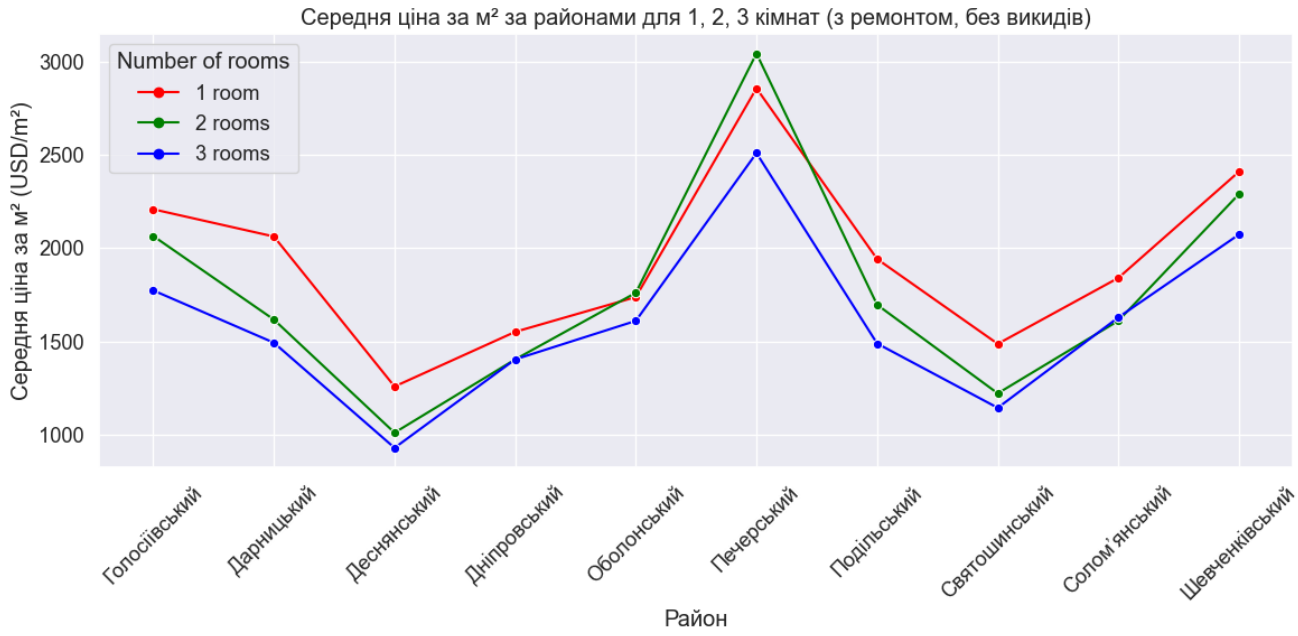


Рисунок 3.2.6 Середня ціна за квадртаний метр за районами, кількістю кімнат і з ремонтом

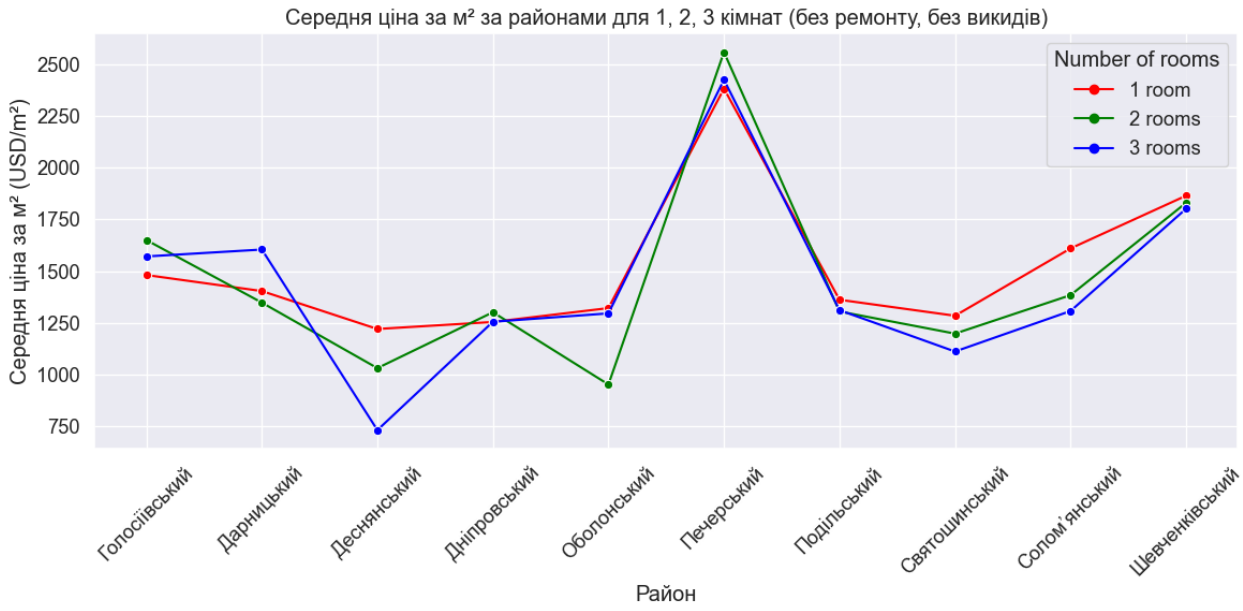
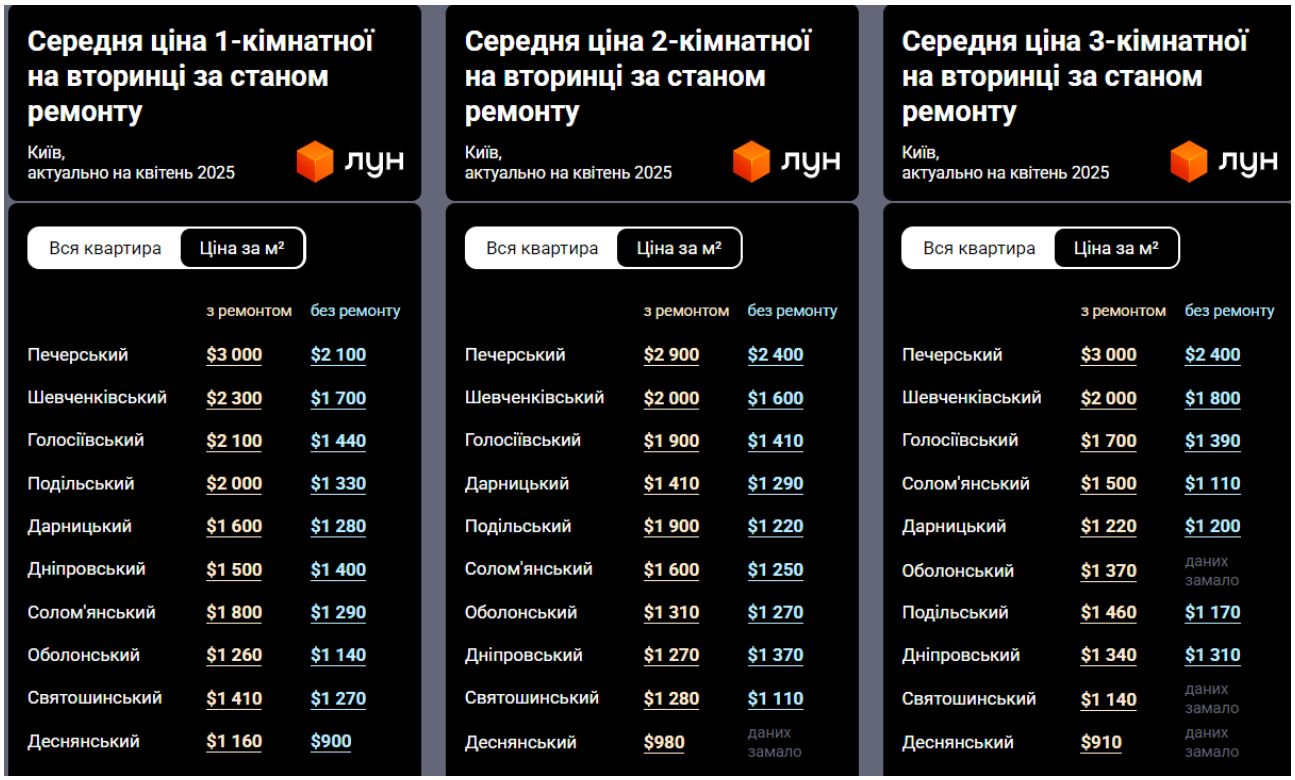


Рисунок 3.2.7 Середня ціна за кв.метр за районами, кількістю кімнат і без ремонту



Середня ціна 1-кімнатної на вторинці за станом ремонту

Київ, актуально на квітень 2025

лун

Вся квартира | Ціна за м²

	з ремонтом	без ремонту
Печерський	$3 000	$2 100
Шевченківський	$2 300	$1 700
Голосіївський	$2 100	$1 440
Подільський	$2 000	$1 330
Дарницький	$1 600	$1 280
Дніпровський	$1 500	$1 400
Солом'янський	$1 800	$1 290
Оболонський	$1 260	$1 140
Святошинський	$1 410	$1 270
Деснянський	$1 160	$900

Середня ціна 2-кімнатної на вторинці за станом ремонту

Київ, актуально на квітень 2025

лун

Вся квартира | Ціна за м²

	з ремонтом	без ремонту
Печерський	$2 900	$2 400
Шевченківський	$2 000	$1 600
Голосіївський	$1 900	$1 410
Дарницький	$1 410	$1 290
Подільський	$1 900	$1 220
Солом'янський	$1 600	$1 250
Оболонський	$1 310	$1 270
Дніпровський	$1 270	$1 370
Святошинський	$1 280	$1 110
Деснянський	$980	даних замало

Середня ціна 3-кімнатної на вторинці за станом ремонту

Київ, актуально на квітень 2025

лун

Вся квартира | Ціна за м²

	з ремонтом	без ремонту
Печерський	$3 000	$2 400
Шевченківський	$2 000	$1 800
Голосіївський	$1 700	$1 390
Солом'янський	$1 500	$1 110
Дарницький	$1 220	$1 200
Оболонський	$1 370	даних замало
Подільський	$1 460	$1 170
Дніпровський	$1 340	$1 310
Святошинський	$1 140	даних замало
Деснянський	$910	даних замало

Рисунок 3.2.8 Середні ціни за кв.метр [21]

Аналіз розподілу загальної ціни на нерухомість у Києві за кількістю кімнат та районами підтверджує попередньо виявлені тенденції щодо різниці цін між районами, навності реионту та кількості кімнат. Проте, варто зазначити, що середня ціна, розрахована на основі власних даних, є вищою за показники,

наведені у звіті веб-сайту нерухомості ЛУН. Таке розходження може бути зумовлене кількома факторами, включаючи відмінності у вибірках об'єктів нерухомості, різний період збору даних, а також потенційні особливості методології розрахунку середньої ціни в обох джерелах.

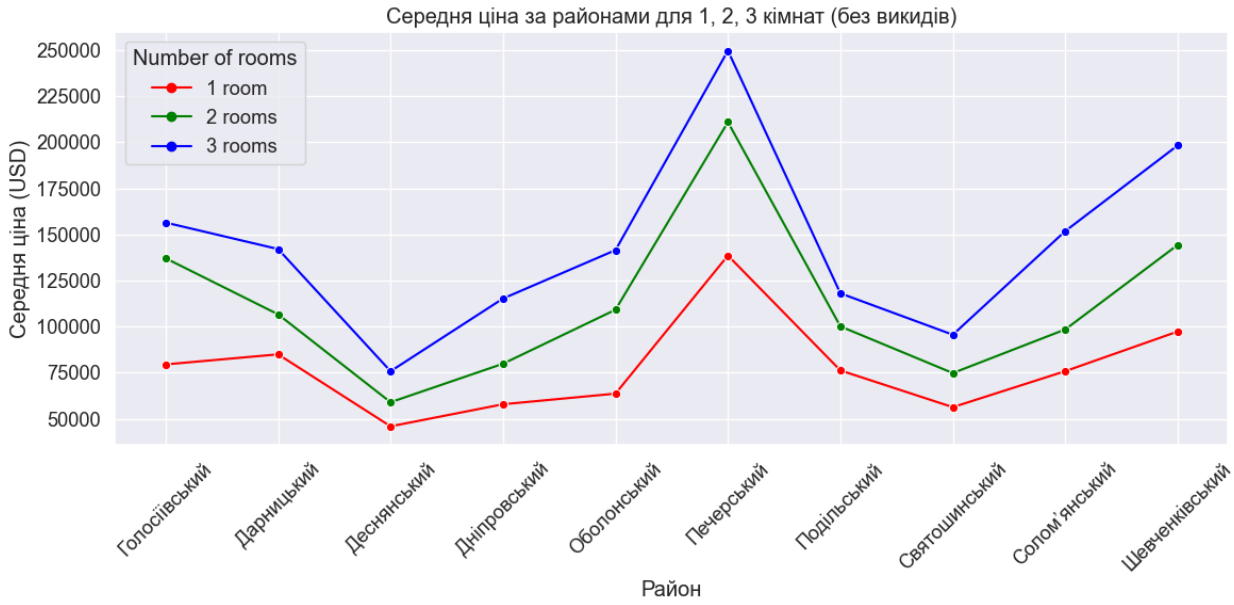


Рисунок 3.2.9 Середня загальна ціна за районми та кількістю кімнат

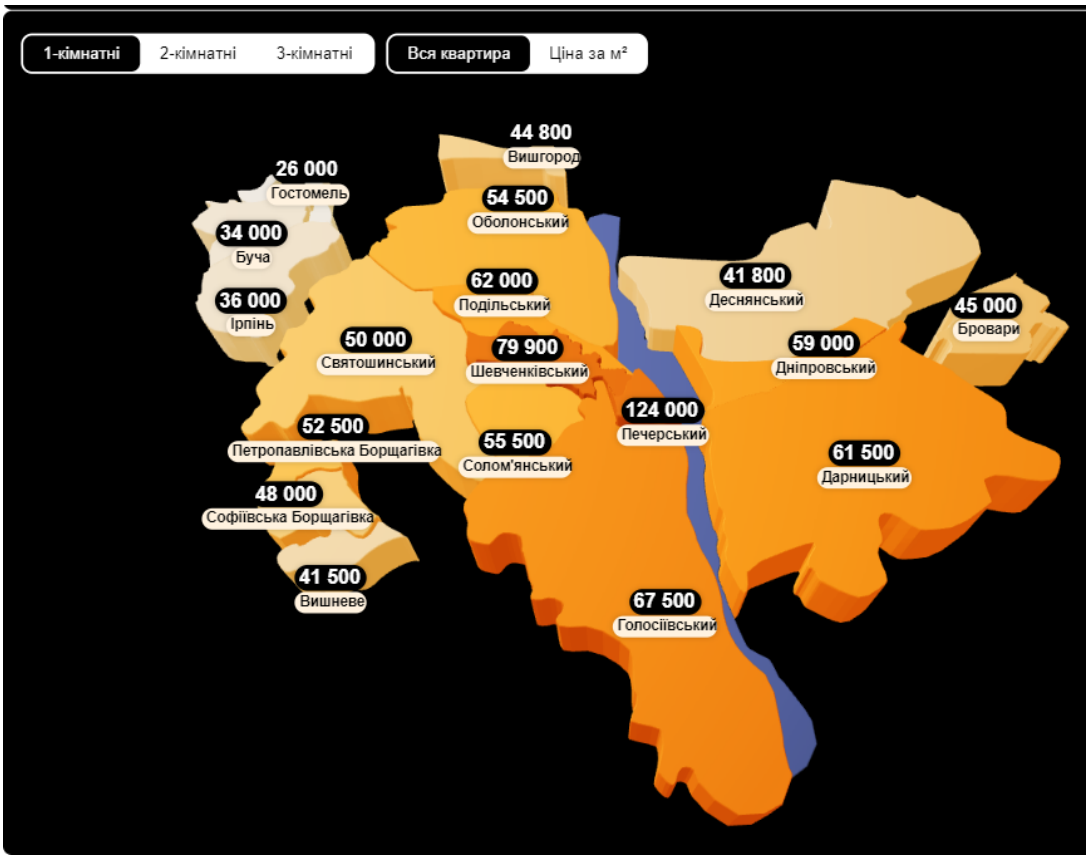


Рисунок 3.2.10 Середня загальна ціна на однокімнатні квартири в Києві [20]

Як і у прикладі з районами ціни на житло залежать від найближчих станцій метро через їх розташування в центрі чи на околицях міста. Також спотерігається різниця між графіками з викидами (усі дані) та без викидів(обмеження до 400 тис. доларів).

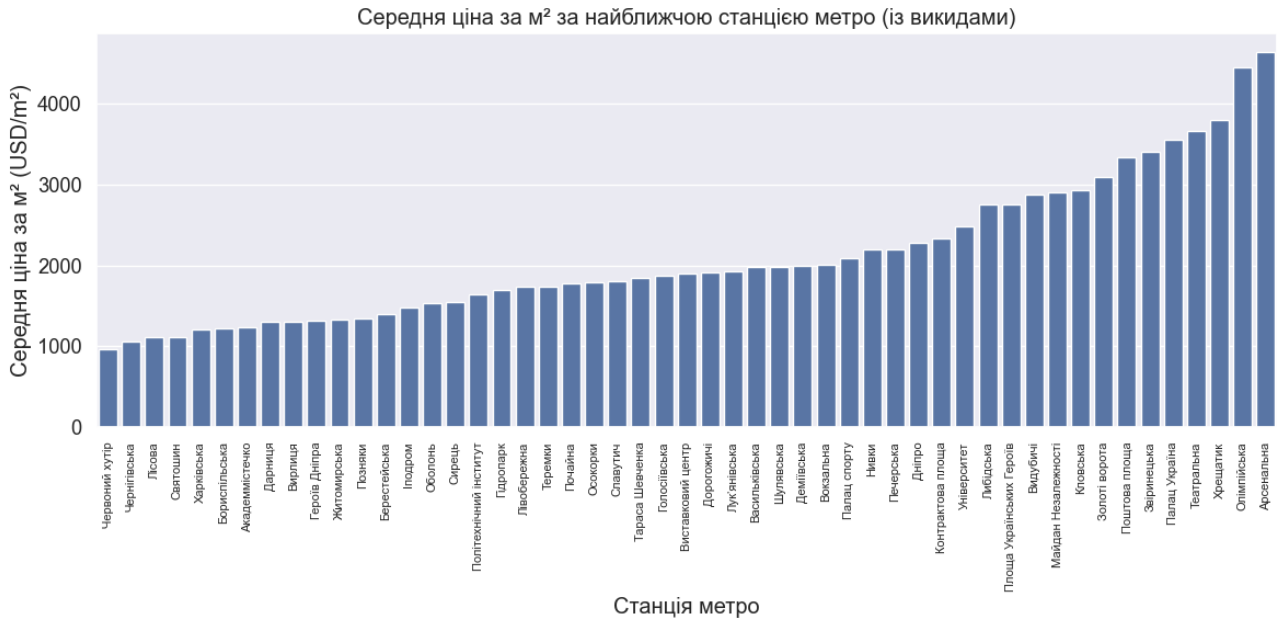


Рисунок 3.2.11 Загальна середня ціна за кв. метр за станціями метро

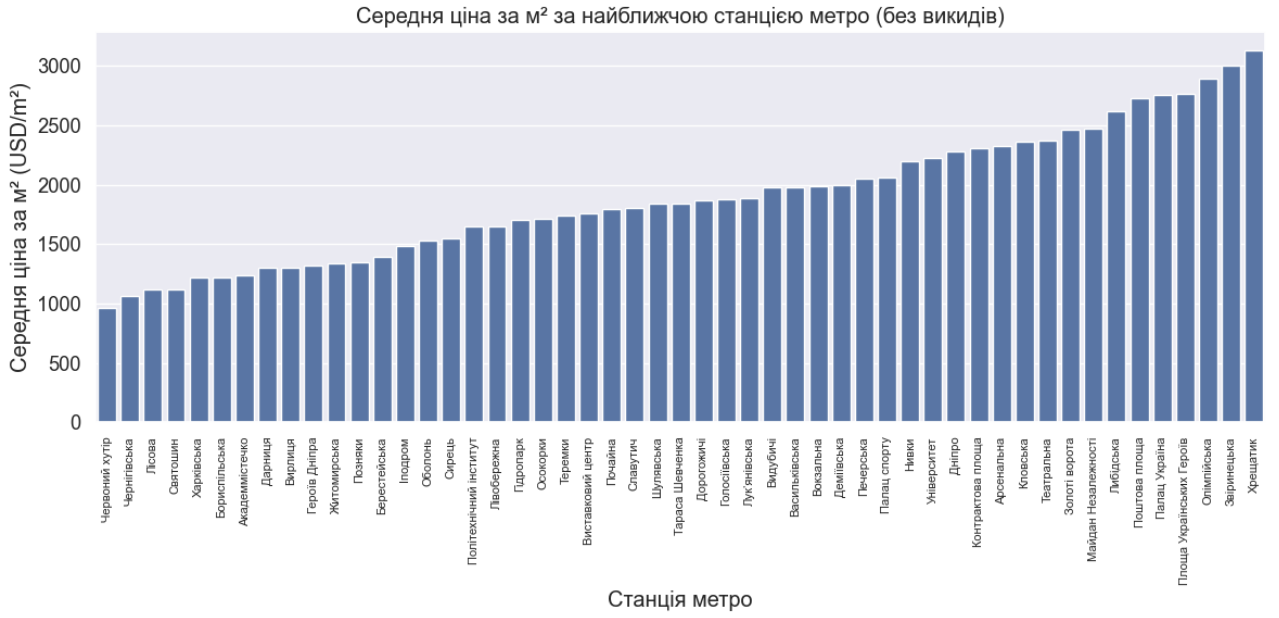


Рисунок 3.2.12 Середня ціна за кв. метр за станціями метро без викидів

На графіку зображно залежність середньої ціни за квадратний метр від відстані до найближчого метро середня ціна за квадратний метр нерухомості знижується зі збільшенням відстані до найближчої станції метро, що підкреслює

значний позитивний вплив близькості до метрополітену на вартість житла через зручність транспортної інфраструктури.

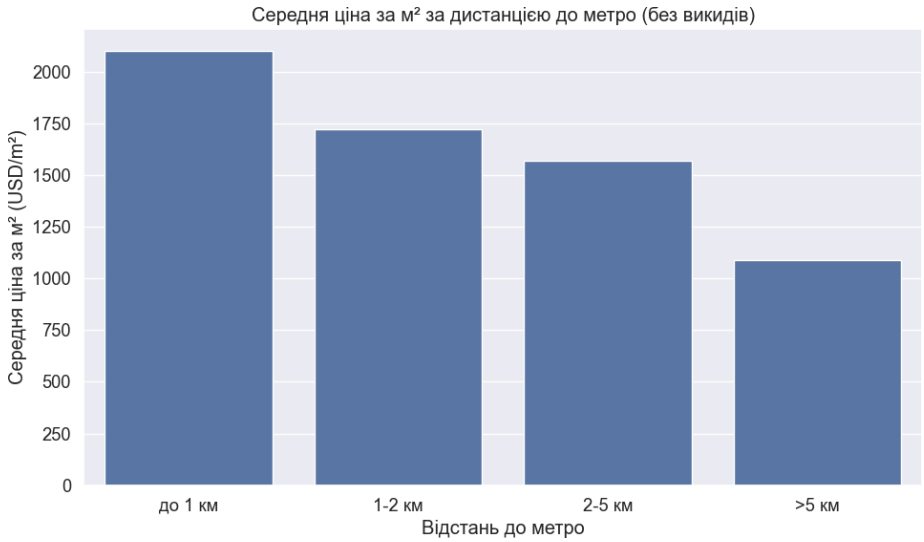


Рисунок 3.2.13 Середня ціна за кв. метр за відстанню до метро

На рисунку 3.2.14 відображено залежність середньої ціни за квадратний метр нерухомості від типу конструкції будівлі. Різниця в ціні зумовлена тим, що монолітно-каркасні будинки, як правило, зводяться за сучаснішими технологіями, забезпечують кращу звуко- та теплоізоляцію, мають гнучкіші планування та часто розташовані в новіших або престижніших районах, що робить їх більш привабливими для покупців порівняно з панельними будинками,

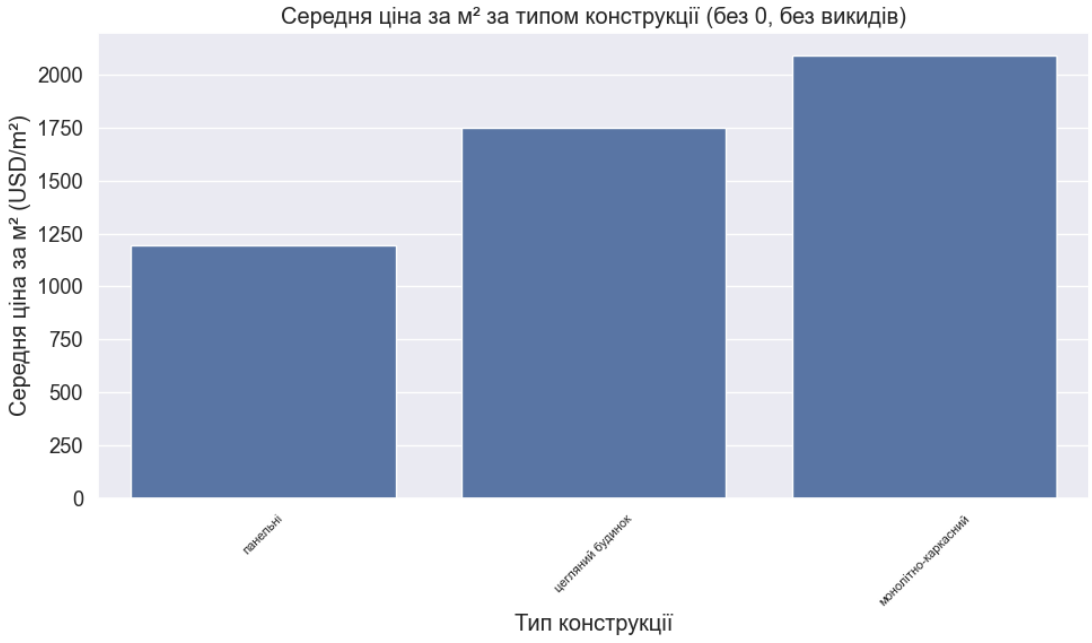


Рисунок 3.2.14 Середня ціна за кв. метр за типом конструкції

На рисунку 3.2.15 представлено розсіяний графік (scatter plot), який ілюструє залежність ціни за квадратний метр житла (у доларах США) від року побудови будівлі. Дані вказують, що більшість будівель, представлених у графіку, були зведені після 1980 року, з особливою концентрацією в період з 2000 по 2020 роки. У цей період ціни за м² переважно коливаються в діапазоні від 1000 до 6000 доларів, із значною щільністю точок у нижній частині цього діапазону. Проте є кілька викидів, де ціна перевищує 10000 доларів за м², зокрема в районі 2000-х і 2010-х років. Для будівель, зведених до 1980 року, дані менш щільні, а ціни зазвичай нижчі, рідко перевищуючи 4000 доларів за м².Графік підкреслює тенденцію зростання цін на житло в новіших будівлях, хоча значна варіативність цін зберігається навіть у межах одного періоду.

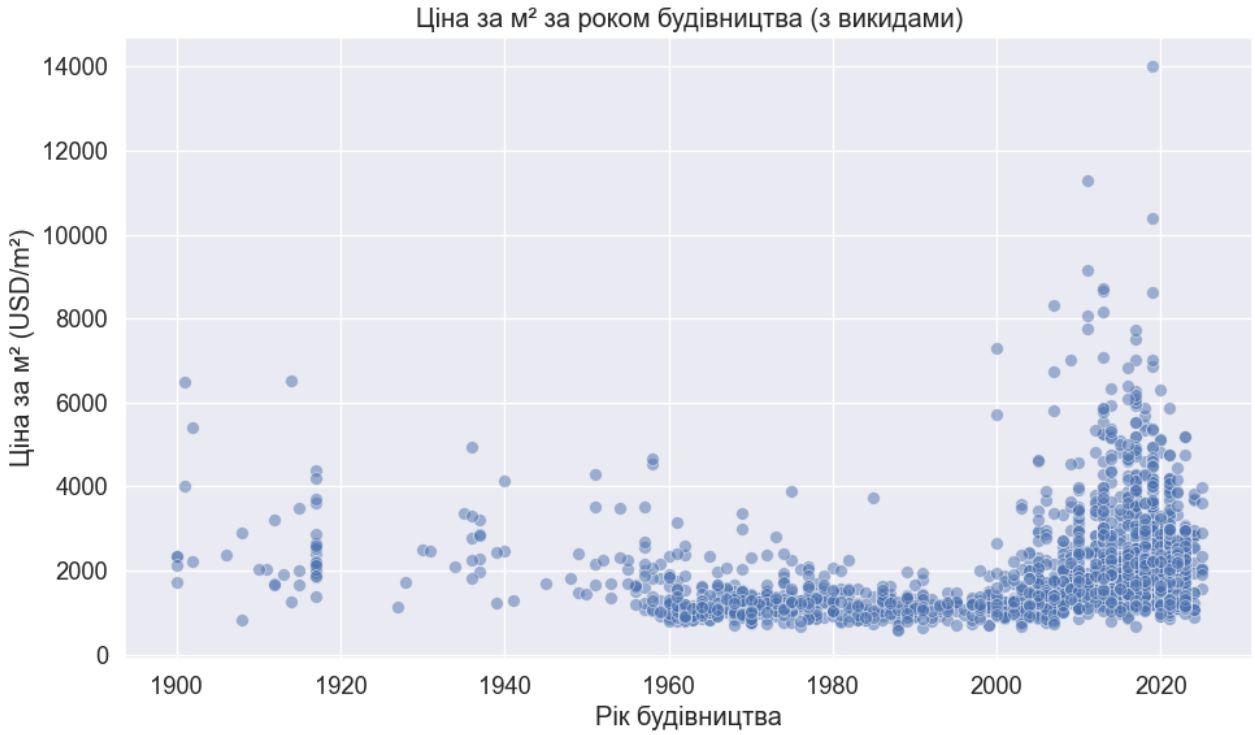


Рисунок 3.2.15 Середня ціна за кв. метр за типом конструкції

3.3 Побудова моделей прогнозування цін на житло

Для створення моделі використано датасет Apartments_final.csv, з якого обрано релевантні ознаки для прогнозування. Датасет очищено від пропусків і викидів: ціни обмежено діапазоном 50,000–400,000 USD, а для числових ознак

(total_area, living_area, distance_to_metro) застосовано правило 1.5*IQR. Створено нові ознаки: age_of_building, living_area_ratio, floor_ratio, а також категоріальні distance_category (на основі відстані до метро в кілометрах) і age_category.

Датасет розділено на тренувальну (80%) та тестову (20%) вибірки за допомогою з використанням стратифікації за ціною, що забезпечує однаковий розподіл цін у обох вибірках, підвищуючи надійність оцінки моделі на нових даних (рис. Такий розподіл забезпечує достатню кількість даних для навчання та дозволяє оцінити здатність моделі до узагальнення на невидимих даних.

Для обробки категоріальних ознак застосовано One-Hot Encoding і Target Encoding. One-Hot Encoding перетворює категоріальні ознаки з низькою кількістю унікальних значень (наприклад, район чи тип будинку) у бінарні стовпці (0 або 1), дозволяючи моделі ефективно враховувати їхній вплив. Target Encoding використовується для ознак із великою кількістю категорій (наприклад, мікрорайон чи комплекс), замінюючи категорії середньою ціною з урахуванням згладжування, щоб уникнути перенавчання та витоку даних. Додатково створено взаємодії між площею та ключовими ознаками (стан ремонту, престижний район) для врахування нелінійних ефектів.

```
Розмір датасету: (1677, 84)
Унікальні значення (до One-Hot Encoding): complex          267
address          1224
microdistrict      88
dtype: int64
Розподіл цін (тренувальна): count      1341.000000
mean     122017.517524
std       58247.998343
min       50000.000000
25%       75500.000000
50%      106000.000000
75%      152000.000000
max      295000.000000
Name: price, dtype: float64
Розподіл цін (тестова): count       336.000000
mean     122205.288690
std       57808.518339
min       50000.000000
25%       76874.250000
50%      107000.000000
75%      155250.000000
max      295000.000000
Name: price, dtype: float64
```

Рисунок 3.3.1 Розподіл цін у вибірках

Модель лінійної регресії була побудована з використанням бібліотеки sklearn.linear_model.LinearRegression для прогнозування цін на квартири на основі датасету Apartments_final_metro.csv. Дані пройшли ретельну передобробку: видалено рядки з пропусками або нульовими значеннями ціни, обмежено ціновий діапазон. Числові ознаки, такі як total_area, year_built, заповнено медіанами, а категоріальні, як complex чи condition, — значеннями 'no_complex' або 'unknown'. Для зменшення кількості рідкісних категорій у address і complex застосовано поріг < 5, у microdistrict < 15, замінюючи їх на 'Other'. Також видалено викиди за міжквартильним розмахом для total_area, living_area, distance_to_metro та price.

Для врахування нелінійних ефектів додано взаємодії ознак, такі як area_x_district, condition_x_district, area_x_condition та distance_x_district. Низькокардинальні категоріальні ознаки (district, condition, nearest_metro) закодовано через One-Hot Encoding, а висококардинальні (address, microdistrict, complex) — через Target Encoding, яке виконано після розбиття даних, щоб уникнути витоку інформації. Ознаку distance_to_metro дискретизовано на чотири категорії (<0.5km, 0.5-1km, 1-2km, 2+km). Дані розбито на тренувальну (80%, 1341 рядків) і тестову (20%, 336 рядків) вибірки зі стратифікацією за ціною для забезпечення однакового розподілу.

Оскільки лінійна регресія чутлива до масштабів ознак, числові змінні стандартизовано за допомогою StandardScaler, застосовуючи fit_transform до тренувальних даних і transform до тестових. Модель навчена на тренувальній вибірці без додаткового налаштування гіперпараметрів, оскільки лінійна регресія є простою моделлю. Після навчання отримано прогнози для тестової вибірки, виведено важливість ознак (на основі абсолютних значень коефіцієнтів регресії) та візуалізовано розподіл цін і графік розсіювання прогнозів проти фактичних значень. Модель забезпечує базовий рівень прогнозування, хоча може бути менш точною для складних нелінійних залежностей порівняно з іншими методами, такими як Random Forest.

Модель випадкового лісу була побудована з використанням бібліотеки `sklearn.ensemble.RandomForestRegressor` для прогнозування цін на квартири на основі датасету Apartments_final_metro.csv. Дані пройшли ретельну передобробку: видалено пропуски та викиди, числові ознаки заповнено медіаною, категоріальні — значеннями unknown або no_complex для відсутніх назв комплексу. Рідкісні категорії (address, complex з порогом мен ше 5, microdistrict менше 15) замінено на Other. Додано взаємодії ознак (area_x_district, area_x_condition). Низькокардинальні ознаки закодовано через One-Hot Encoding, що перетворює категорії на бінарні стовпці, а висококардинальні — через Target Encoding, що замінює категорії середніми значеннями цільової змінної, виконаний після розбиття даних для уникнення витоку.

Гіперпараметри моделі налаштовано через GridSearchCV (5 фолдів, 80 фітів) , зображені на рисунку 3.3.2:

```
# Random Forest with GridSearchCV
estimator = RandomForestRegressor(random_state=42)
parameters = {
    'n_estimators': [100, 200],
    'max_depth': [5, 7],
    'min_samples_split': [5, 10],
    'min_samples_leaf': [2, 4],
    'max_features': ['sqrt']
}
```

Рисунок 3.3.2 Налаштування гпараметрів для випадкового лісу

Випадковість у побудові дерев (через `max_features='sqrt'`) зменшує кореляцію між деревами, сприяючи узагальненню.

Для прогнозування цін на нерухомість використано бібліотеку XGBoost, зокрема модель XGBRegressor із параметром tree_method='hist', який забезпечує швидке навчання за допомогою гістограмного методу побудови дерев. Цей підхід оптимізує обчислення, що особливо важливо для датасетів із великою кількістю ознак і зразків.

Оптимізацію гіперпараметрів проведено за допомогою GridSearchCV із 5-кратною крос-валідацією, що передбачає поділ тренувальних даних на п’ять частин, де модель навчається на чотирьох частинах і оцінюється на п’ятій, повторюючи процес для всіх комбінацій(рисунок 3.3.2) . Це забезпечило надійну оцінку узагальнюючої здатності моделі та допомогло уникнути перенавчання. Через GridSearchCV протестовано численні комбінації параметрів, включаючи швидкість навчання (learning_rate), максимальну глибину дерев (max_depth), мінімальну вагу дочірніх вузлів (min_child_weight), кількість дерев (n_estimators), параметри регуляризації (reg_lambda, reg_alpha), частку вибірки (subsample) та ознак (colsample_bytree), а також параметр gamma для контролю складності дерев(рис.3.3.3). Найкраща комбінація параметрів визначалася за критерієм мінімальної середньої відсоткової абсолютної помилки (MAPE), що є ключовою метрикою для оцінки точності прогнозів цін.

```
# XGBoost with GridSearchCV
estimator = xgb.XGBRegressor(random_state=42, enable_categorical=False)
parameters = {
    'learning_rate': [0.01, 0.05],
    'max_depth': [2, 3],
    'min_child_weight': [20, 30],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8],
    'n_estimators': [75, 100],
    'reg_lambda': [30.0, 40.0],
    'reg_alpha': [0.1, 1.0],
    'gamma': [0.5, 1.0]
}
```

Рисунок 3.3.3 Налаштування рпараметрів для XGBoost

Модель із найкращими параметрами навчалася на тренувальній вибірці (fit) та прогнозувала ціни на тренувальній і тестовій вибірках (predict).

3.4 Оцінка якості моделі

Для оцінки ефективності моделей прогнозування цін на нерухомість використано метрики:

- MAE (Mean Absolute Error): Середня абсолютна помилка, що відображає середнє відхилення між фактичною та прогнозованою вартістю, виміряне в одиницях валюти.
- MAPE (Mean Absolute Percentage Error): Середня відсоткова абсолютна помилка, що відображає середню відсоткову різницю між фактичними та прогнозованими цінами.
- MdAE (Median Absolute Error): Медіанна абсолютна помилка, що показує медіанну різницю між фактичними та прогнозованими цінами. Менш чутлива до великих відхилень, ніж MAE.
- MdAPE (Median Absolute Percentage Error): Медіанна відсоткова абсолютна помилка, що відображає медіанну відсоткову різницю. Допомагає оцінити типову відносну помилку, ігноруючи екстремальні відхилення. Приклад, яким чином ввдено цю метрику, зображено на рисунку 3.4.1 :

```python
# Custom metric: Median Absolute Percentage Error
def median_absolute_percentage_error(actual: np.ndarray, predicted: np.ndarray):
    EPSILON = 1e-10
    return np.median(np.abs((actual - predicted) / (actual + EPSILON)))
```

Рисунок 3.4.1 Медіанна відсоткова абсолютна помилка

Ці метрики обчислено для тренувальної (1341 рядок) і тестової (336 рядків) вибірок, що становлять загалом 1677 рядків, і представлено в таблиці для порівняння. Додатково проаналізовано результати на тренувальних даних для виявлення ознак перенавчання, а також при крос-валідації. Крос-валідація - метод оцінки, при якому тренувальні дані діляться на кілька частин (фолдів), модель навчається на одних частинах і оцінюється на інших, що забезпечує надійну оцінку узагальнюючої здатності. Для випадкового лісу застосовано 5-кратну крос-валідацію з 8 комбінаціями параметрів, для градієнтного бустінгу — 3-кратну з 256 комбінаціями.

Лінійна регресія продемонструвала помірну точність (рис.3.4.1). На тестовій вибірці MAE становить 22575.91 доларів , що вказує на середню абсолютну різницю, а MAPE 18.91% відображає відсоткову похибку. MdAE

16753.91 доларів і MdAPE 14.71% підтверджують стабільність, але нижчу точність через лінійну природу моделі. Близькість тренувальних (MAE: 20061.39, MAPE: 0.1784) і тестових метрик свідчить про відсутність перенавчання. Для візуалізації побудовано графіки розсіювання (рис.3.4.2), де фактичні ціни відображено на осі X, прогнозовані — на осі Y, з ідеальною лінією відповідності (y=x).

```
Training Metrics:
Train MAE: 20061.38717, MdAE: 15257.05143, MAPE: 0.17839, MdAPE: 0.13830

Test Metrics:
Test MAE: 22575.91243, MdAE: 16753.91495, MAPE: 0.18910, MdAPE: 0.14707
```

Рисунок 3.4.1 Оцінки метрик лінійної регресії

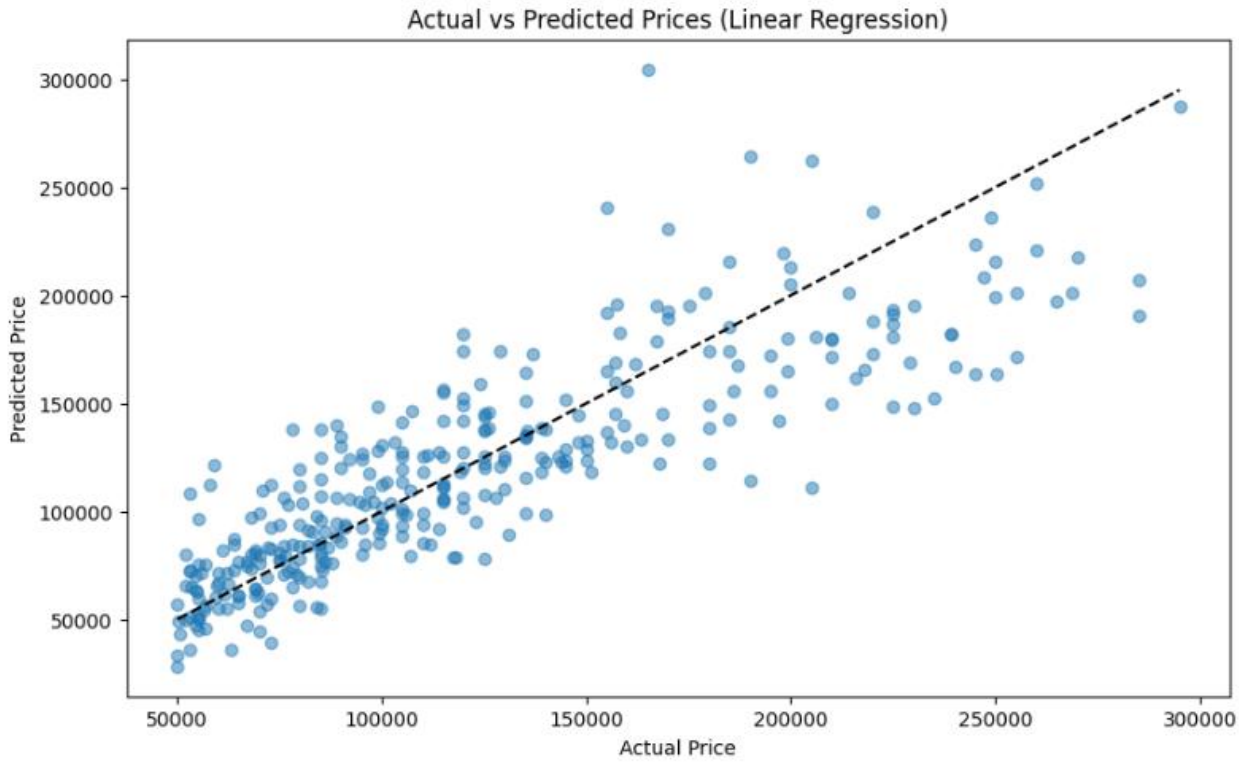


Рисунок 3.4.2 Прогноз ціни лінійної регресією

Модель випадкового лісу показала нижчу точність (рис.3.4.3, рис.3.4.4). Крос-валідація виявила MAE 29378.69 USD і MAPE 27.15%, що вказує на більші помилки порівняно з іншими моделями. На тестовій вибірці MAE склало 29066.77 доларів, MAPE — 26.96%, а MdAE 24971.92 і MdAPE 22.72%

підтверджують помірні відхилення. Тренувальні метрики (MAE: 27569.12 , MAPE: 0.2546) близькі до тестових, що свідчить про відсутність перенавчання, але модель поступається за точністю через обмежену здатність до моделювання складних залежностей.

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
CV Best neg_MAE: -24901.16262
CV Best neg_MAPE: -0.22335
CV Best params: {'max_depth': 7, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'n_estimators': 200}

Training Metrics:
Train MAE: 20769.83669, MdAE: 17866.56904, MAPE: 0.19073, MdAPE: 0.16311

Test Metrics:
Test MAE: 24289.80457. MdAE: 20460.19718. MAPE: 0.22079. MdAPE: 0.18801
```

Рисунок 3.4.3 Оцінки метрик методу випадкового лісу

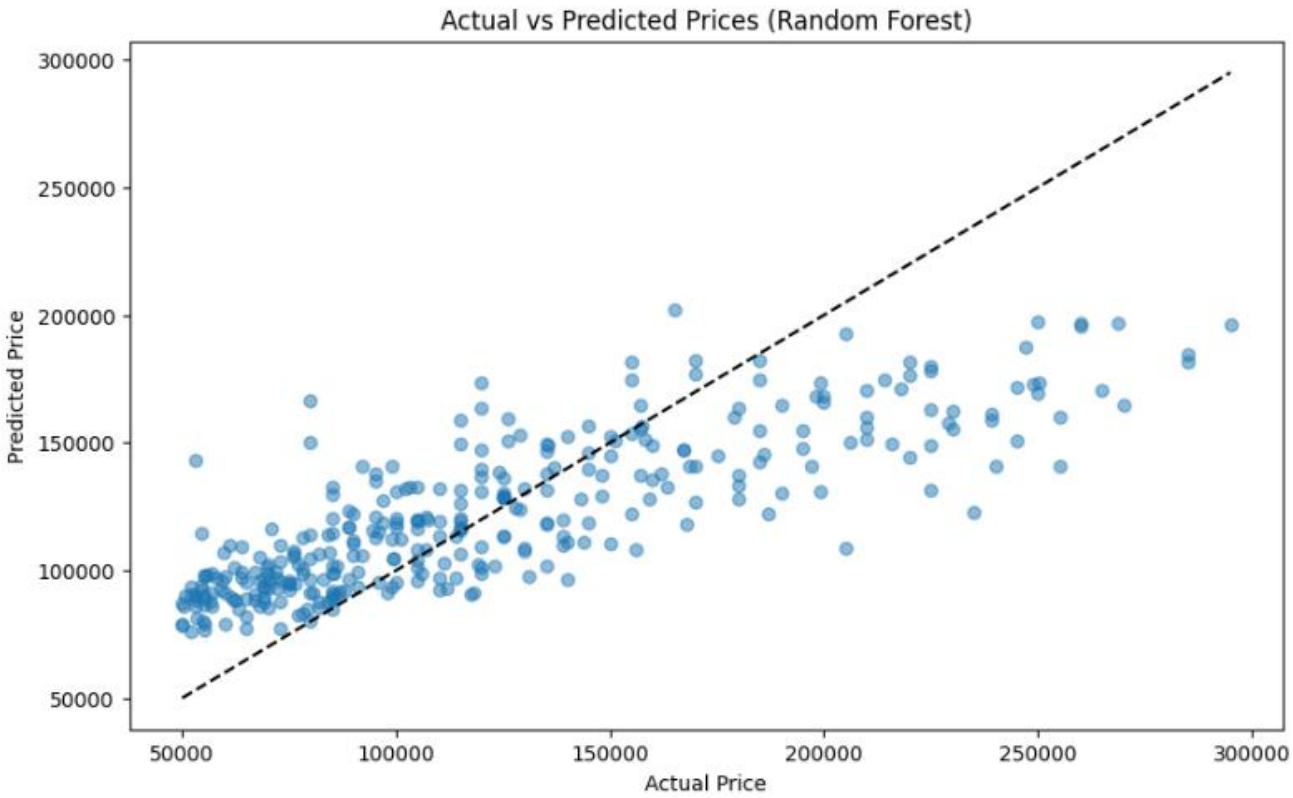


Рисунок 3.4.4 Прогноз методом випадкового лісу

Модель градієнтного бустінгу досягла найкращих результатів (рис.3.4.5,рис.3.4.6). Крос-валідація показала MAE 22548.29 і MAPE 0.1832 (18.32%), що підтверджує високу узагальнюючу здатність. На тестовій вибірці MAE становить 22924.06 доларів, MAPE —18.69%, а MdAE (16528.39) і MdAPE (16.06%) вказують на стабільність і точність. Тренувальні метрики (MAE:

20193.97, MAPE: 0.1640) близькі до тестових, що підтверджує відсутність перенавчання та високу адаптивність моделі до складних даних.

```
Fitting 3 folds for each of 256 candidates, totalling 768 fits
CV Best neg_MAE: -22548.28741
CV Best neg_MAPE: -0.18321
CV Best params: {'colsample_bytree': 0.8, 'gamma': 0.5, 'learning_rate': 0.05, 'max_depth': 3, 'min_child_weight': 20,
'n_estimators': 100, 'reg_alpha': 0.1, 'reg_lambda': 30.0, 'subsample': 1.0}

Training Metrics:
Train MAE: 20193.97491, MdAE: 13853.64844, MAPE: 0.16396, MdAPE: 0.13662

Test Metrics:
Test MAE: 22924.05635, MdAE: 16528.38867, MAPE: 0.18692, MdAPE: 0.16064
```

Рисунок 3.4.5 Прогноз методом XGBoost

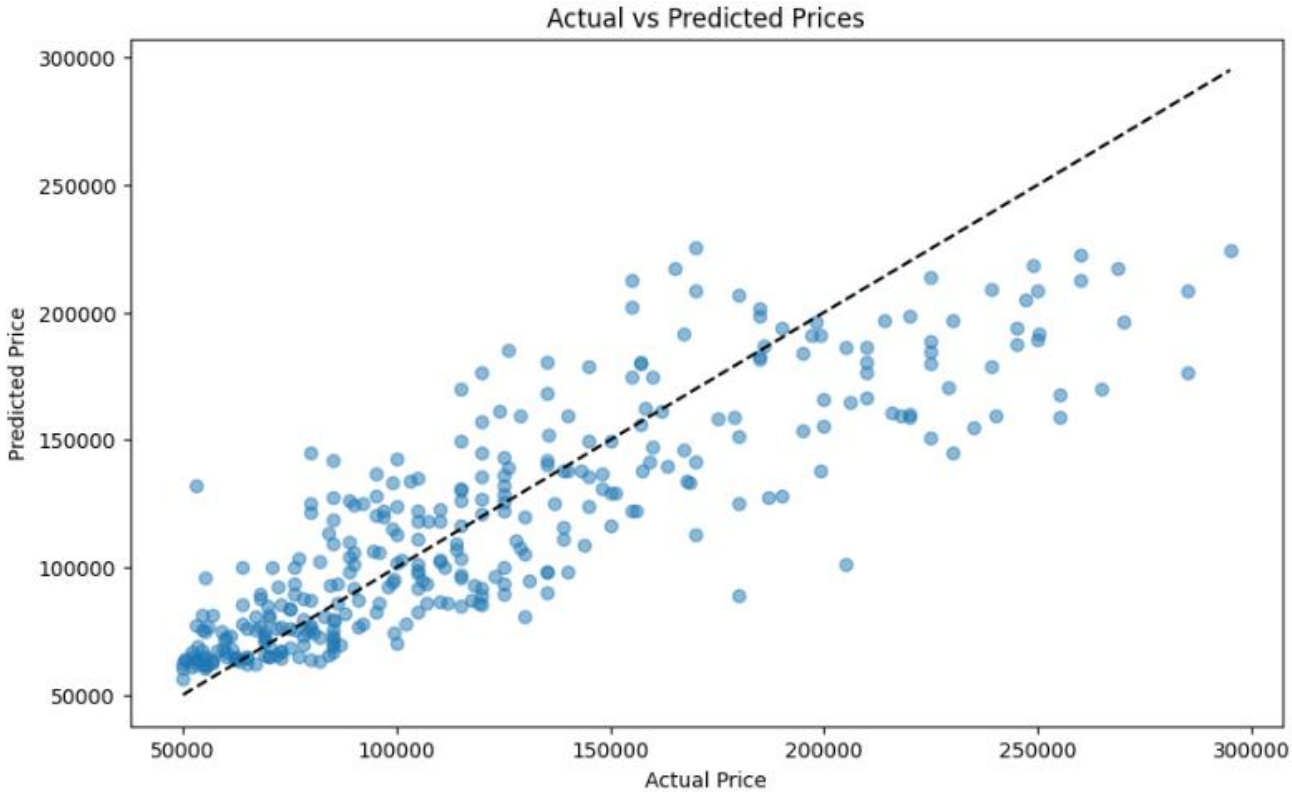


Рисунок 3.4.3 Прогноз XGboost

Графіки дозволяють візуально оцінити точність моделей, підтверджуючи, що XGBoost має найменші відхилення, тоді як лінійна регресія та випадковий ліс показують більші розбіжності.

Метрики оцінки моделей на різних вибірках відображен у таблиці 3.1:

Таблиця 3.1 - Оцінки якості побудованих моделей

Модель	**Вибірка**	**MAE**	**MAPE**	**MdAE**	**MdAPE**
Лінійна регресія	Навчальна	20061.39	0.1784	15257.05	0.1383
	Тестова	22575.91	0.1891	16753.91	0.1471
Випадковий ліс	Крос-валідація	29378.69	0.2715	-	-
	Навчальна	27569.12	0.2546	23395.99	0.2167
	Тестова	29066.77	0.2696	24971.92	0.2272
XGBoost	Крос-валідація	22548.29	0.1832	-	-
	Навчальна	20193.97	0.1640	13853.65	0.1366
	Тестова	22924.06	0.1869	16528.39	0.1606

Аналіз важливості ознак для моделей випадкового лісу, градієнтного бустінгу і лінійної регресії показав, що ключовими факторами, які впливають на прогнозування цін на нерухомість, є загальна площа квартири та її розташування. У моделях випадкового лісу та градієнтного бустингу ознака total_area має найвищу важливість (16.98% і 23.77% відповідно), що підтверджує прямий зв’язок між розміром квартири та її вартістю. Ознаки розташування, такі як microdistrict_target_encoded (10.54% для випадкового лісу, 10.17% для XGBoost) і district_Печерський (3.77% і 7.19%), також входять до топ-5, підкреслюючи значущість географічного фактора, особливо в престижних районах. Взаємодії, такі як area_x_condition/area_condition_remont (9.31% і 6.70%) та area_x_district/area_district_pecherskyi (5.25% і 3.92%), додатково підвищують точність прогнозів, відображаючи нелінійний вплив ремонту та престижного розташування на ціну залежно від площі.

Модель XGBoost демонструє більшу концентрацію важливості на ключових ознаках, таких як total_area і district_Печерський, що вказує на її здатність ефективно виділяти найвпливовіші фактори, тоді як випадковий ліс розподіляє вагу рівномірніше, включаючи додаткові ознаки, як living_area

(7.46%) і rooms (3.52%). Лінійна регресія фокусується виключно на близькості до центральних станцій метро (наприклад, nearest_metro_Хрещатик з коефіцієнтом 137612.12), ігноруючи площу та взаємодії, що обмежує її здатність враховувати комплексні залежності. Некорисна площа (not_living_area) і тип будинку (construction_type_панельні, construction_type_цегляний будинок) також мають помітний вплив у деревних моделях, відображаючи важливість планування та матеріалів.

Feature Importance (Top 10, Absolute Coefficients):

	coefficient
nearest_metro_Хрещатик	137612.117747
nearest_metro_Площа Українських Героїв	132683.031945
nearest_metro_Поштова площа	97939.476785
nearest_metro_Майдан Незалежності	96784.940892
nearest_metro_Золоті ворота	93256.630691
nearest_metro_Палац спорту	71006.842238
nearest_metro_Олімпійська	68927.733959
nearest_metro_Університет	61795.265017
nearest_metro_Контрактова площа	58149.277656
nearest_metro_Дорогожичі	56161.204723

Рисунок 3.4.4 Важливість ознак для лінійної регресії

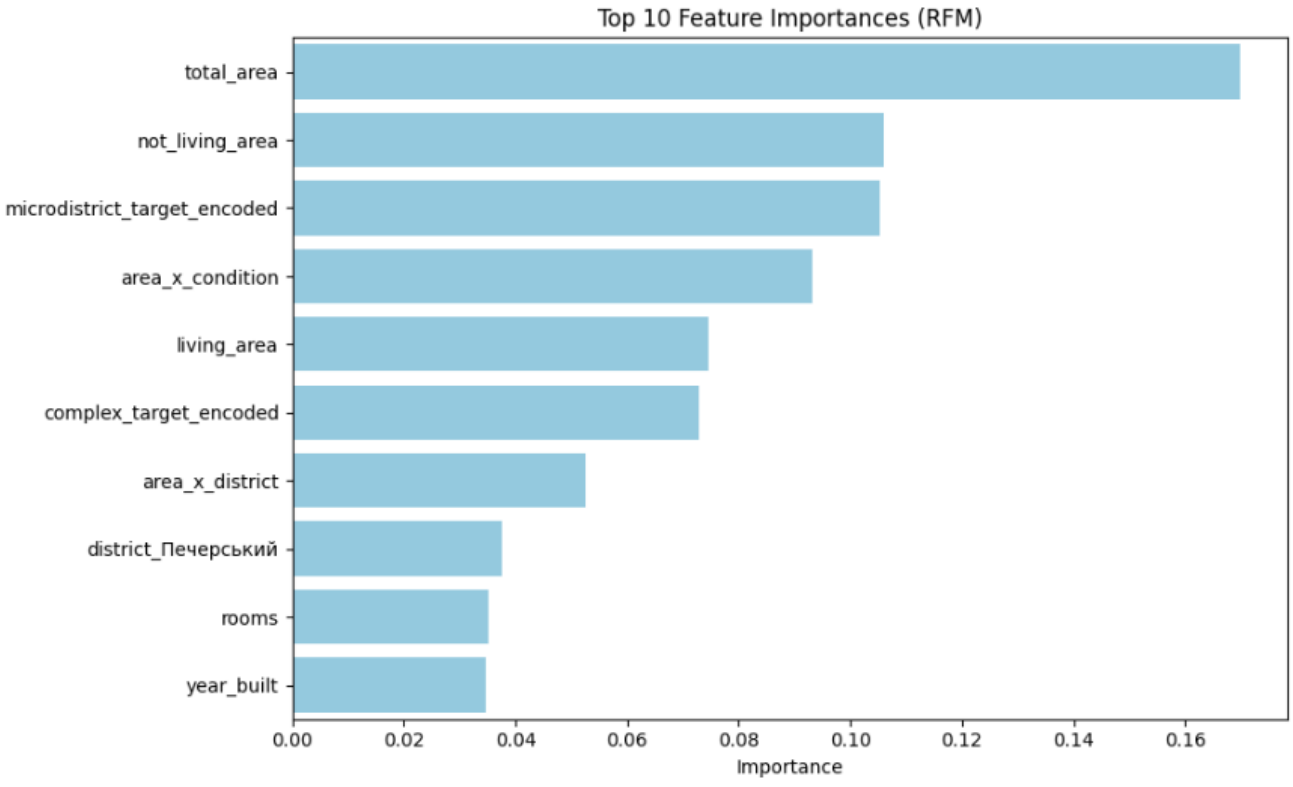


Рисунок 3.4.5 Важливість ознак для випадкового лісу

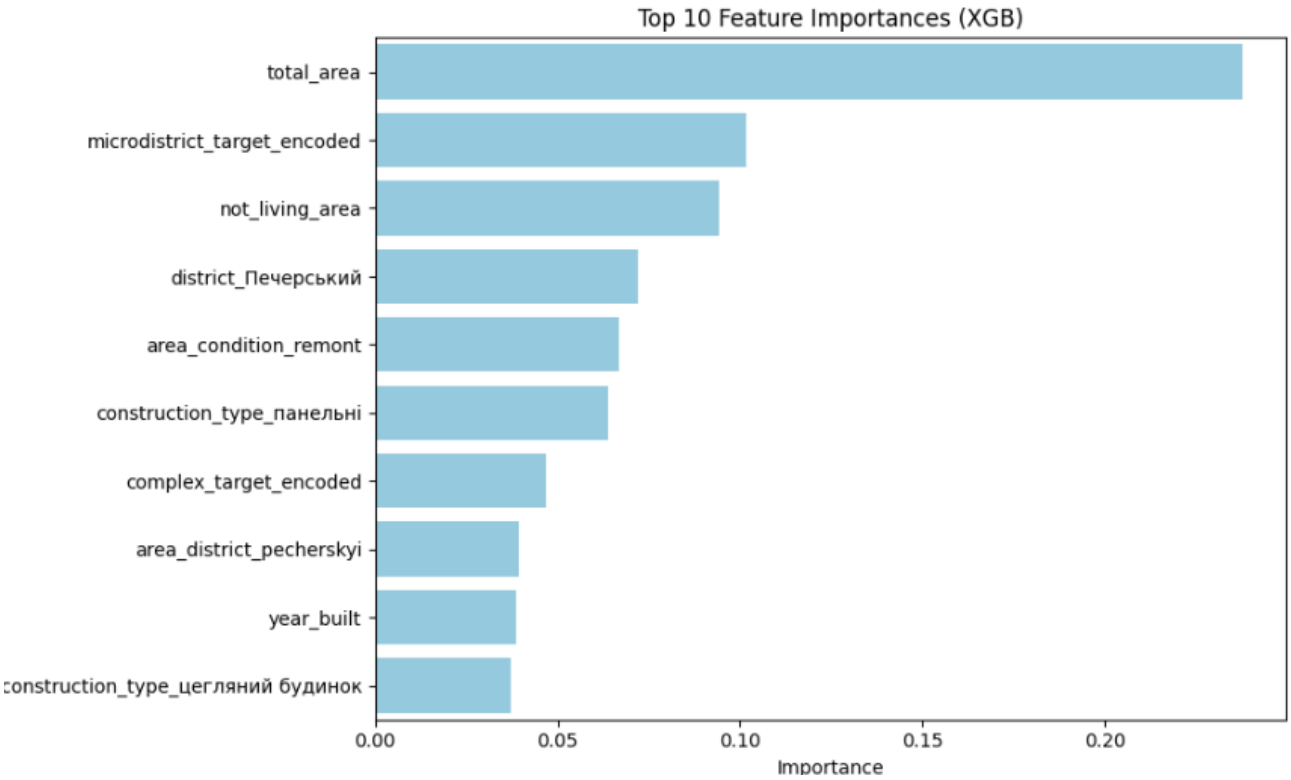


Рисунок 3.4.6 Важливість ознак для випадкового лісу

Таким чином, XGBoost забезпечує найточнішу ідентифікацію ключових драйверів цін завдяки чіткому виділенню площі, розташування та їхніх взаємодій, що узгоджується з його вищою прогнозною точністю. Випадковий ліс і лінійна регресія, хоч і підкреслюють схожі фактори, менш ефективно враховують складні взаємозв'язки, що робить їх менш інформативними для цієї задачі. Загалом, загальна площа, розташування (особливо престижні райони), стан ремонту та тип будинку є основними факторами, що визначають ціни на нерухомість.

Висновки

У розділі здійснено повноцінний аналіз даних про продаж квартир на вторинному ринку нерухомості Києва. Датасет, що включає дані про ціну, площу, розташування, стан ремонту та інші характеристики, було оброблено: очищено від викидів, заповнено пропуски, додано географічні ознаки (відстань до метро, координати) та створено нові ознаки, зокрема взаємодії між площею, ремонтом і престижними районами. Аналіз виявив ключові тенденції: вищі ціни

в центральних районах, значний вплив площі та близькості до метро, а також вищу ціну за квадратний метр для квартир із ремонтом і однокімнатних об’єктів.

Для прогнозування цін побудовано три моделі: лінійну регресію, випадковий ліс і градієнтний бустінг. Моделі навчалися на підготовлених даних, а їхні гіперпараметри оптимізовано для підвищення точності. Оцінка якості проводилася за метриками середньої та медіанної абсолютної і відсоткової помилок, а також аналізом важливості ознак.

XGBoost продемонстрував найкращі результати, значно перевершуючи випадковий ліс і лінійну регресію завдяки здатності ефективно моделювати складні залежності. Основними факторами ціноутворення визначено площу, розташування (особливо престижні райони) і стан ремонту. Випадковий ліс був менш точним, а лінійна регресія виявилася обмеженою через нездатність враховувати нелінійні ефекти.

Розроблений метод, що охоплює обробку, аналіз і моделювання даних, підтвердив свою ефективність для прогнозування цін на нерухомість. Результати можуть бути використані для прийняття рішень на ринку, але потребують подальшого покращення і розширення даних для врахування додаткових ринкових факторів.

РОЗДІЛ 4. ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ ОЦІНКИ У СФЕРІ НЕРУХОМОСТІ

4.1 Застосування побудованих моделей на практиці

Моделі оцінки нерухомості, подібні до розробленої моделі на основі екстремального градієнтного бустингу для прогнозування цін на квартири, мають значний потенціал для використання в різних секторах економіки, де потрібна швидка, точна та об’єктивна оцінка вартості об’єктів. У ріелторській діяльності такі моделі автоматизують процес визначення ринкової вартості житла, дозволяючи агентствам надавати клієнтам обґрунтовані цінові пропозиції, скорочувати час підготовки угод та підвищувати конкурентоспроможність на ринку. У страхових компаніях моделі оцінки застосовуються для визначення страхової вартості об’єктів, що є основою для розрахунку страхових премій, оцінки ризиків та формування страхових резервів. Державні установи, такі як податкові органи чи муніципалітети, можуть використовувати ці моделі для оцінки нерухомості з метою нарахування податків на майно, планування міського розвитку або управління державними активами. Інвестиційні фонди та девелоперські компанії застосовують моделі для аналізу прибутковості проєктів, оцінки доцільності інвестування в будівництво, реконструкцію чи придбання об’єктів нерухомості.

У сфері фінансових технологій моделі оцінки інтегруються в онлайн-платформи, які надають користувачам миттєві оцінки вартості житла, сприяючи цифровізації ринку нерухомості. Наприклад, такі платформи можуть пропонувати інструменти для порівняння цін, прогнозування зростання вартості чи оцінки інвестиційної привабливості об’єктів. Крім того, моделі можуть бути використані в юридичній практиці для оцінки майна під час судових спорів, спадкових справ чи поділу активів. Загалом, застосування таких моделей сприяє підвищенню прозорості ринку, зменшенню суб’єктивізму в оцінці, оптимізації бізнес-процесів і здійсненню обдуманих рішень.

На базі практики у банківській сфері розроблена модель оцінки нерухомості може бути ефективно впроваджена для автоматизації та підвищення точності оцінки заставного майна, що є значною складовою кредитних процесів. У банку модель має широкий спектр застосувань, які охоплюють як операційні, так і стратегічні задачі. По-перше, під час скорингу клієнтів модель може бути інтегрована в загальну скорингову систему банку, де дані про об’єкт нерухомості (район розташування, мікрорайон, загальна площа, стан квартири, кількість кімнат, рік побудови тощо) використовуються як додаткові змінні для оцінки кредитоспроможності позичальника. Це дозволяє точніше визначити ризик неповернення кредиту, особливо коли нерухомість виступає заставою, і зменшити ймовірність дефолту. По-друге, під час видачі іпотечних кредитів модель забезпечує автоматичну оцінку ринкової вартості заставного майна, що знижує залежність від зовнішніх оцінювачів, скорочує час обробки кредитних заявок і підвищує об’єктивність рішень. Наприклад, менеджер банку може ввести дані про квартиру в веб-додаток, подібний до розробленого, і отримати прогнозовану ціну, яка використовується для визначення максимальної суми кредиту. По-третє, модель може застосовуватися в межах державних програм підтримки населення, таких як «єОселя» чи «Доступна іпотека 5-7-9%», де швидка перевірка відповідності нерухомості встановленим критеріям вартості є критично важливою. По-четверте, для управління кредитним портфелем модель дає змогу проводити регулярну переоцінку заставної нерухомості на основі актуальних ринкових даних. Це допомагає виявляти об’єкти, ринкова вартість яких знизилася, і вживати заходів для мінімізації ризиків, таких як перегляд умов кредиту чи вимога додаткового забезпечення. Наприклад, банк може щоквартально запускати модель для переоцінки всіх об’єктів у портфелі, що особливо актуально в умовах економічної нестабільності чи коливань цін на ринку нерухомості.

Крім того, модель може бути використана для стратегічної аналітики в банку. Агреговані результати оцінки дозволяють досліджувати ринкові тренди, такі як зростання цін у певних мікрорайонах чи залежність вартості від типу

будинку. Це дає змогу банку розробляти нові кредитні продукти, адаптовані до регіональних особливостей, або оптимізувати умови іпотечного кредитування для різних сегментів клієнтів. Наприклад, банк може запропонувати знижені ставки для квартир у мікрорайонах із високим попитом або розробити спеціальні програми для новобудов. Інтеграція моделі з цифровими системами банку, такими як CRM чи онлайн-банкінг, дозволяє створити зручний інтерфейс для клієнтів, наприклад, онлайн-калькулятор іпотечної вартості, який надає попередню оцінку ще на етапі подання заявки. Такий інструмент підвищує клієнтський досвід і сприяє залученню нових позичальників.

4.2 Концепція впровадження моделі прогнозування вартості нерухомості

Розроблена модель оцінки нерухомості на основі алгоритму XGBoost була успішно впроваджена як веб-додаток з використанням фреймворку Flask для автоматизації оцінки вартості квартир у таких сферах, як ріелторські послуги. Концепція реалізації передбачала створення зручного інтерфейсу для введення даних про об’єкт нерухомості, обробку цих даних моделлю та надання прогнозованої вартості користувачу. Вибір Flask як основи розробки обґрунтований його простотою, можливістю легкої адаптації та інтеграції з іншими технологіями, що зробило цей фреймворк оптимальним для даного проєкту.

Архітектура рішення. Реалізований веб-додаток складається з трьох основних компонентів: фронтенду, бекенду та моделі машинного навчання. Фронтенд, створений на основі HTML-шаблонів Flask, включає форму з полями для введення характеристик квартири (загальна площа, кількість кімнат, поверх, рік побудови, тип будинку, стан, відстань до метро, район, найближче метро, мікрорайон). Поля розміщені в логічному порядку, а значення випадаючих списків, таких як райони чи мікрорайони, відсортовані за алфавітом для зручності користувача. Бекенд, побудований на Flask, обробляє введені дані, виконує їх валідацію (наприклад, перевіряє, чи значення площі та кількості

кімнат додатні) і передає підготовлені дані до моделі. Модель XGBoost, збережена у файлі xgb_model.pkl, генерує прогноз ціни на основі оброблених даних. Додаткові файли, такі як categories.pkl (зберігає категорії для випадаючих списків) і target_encoding_maps.pkl (містить мапи таргет-енкодінгу для мікрорайонів), забезпечують коректну обробку категоріальних ознак.

Алгоритм роботи додатку був реалізований наступним чином (рис.4.2.1). Користувач, наприклад, ріелтор,вводить дані про квартиру через веб-форму. Після натискання кнопки "Прогнозувати ціну" дані надсилаються на сервер через POST-запит. Flask-сервер приймає дані, перевіряє їх коректність (наприклад, чи всі поля заповнені) і трансформує їх: числові ознаки (площа, поверх) конвертуються в числа, а категоріальні (район, стан) кодуються за допомогою one-hot або таргет-енкодінгу. Модель XGBoost обробляє підготовлені дані та повертає прогнозовану ціну, яка відображається на веб-сторінці у зрозумілому форматі.

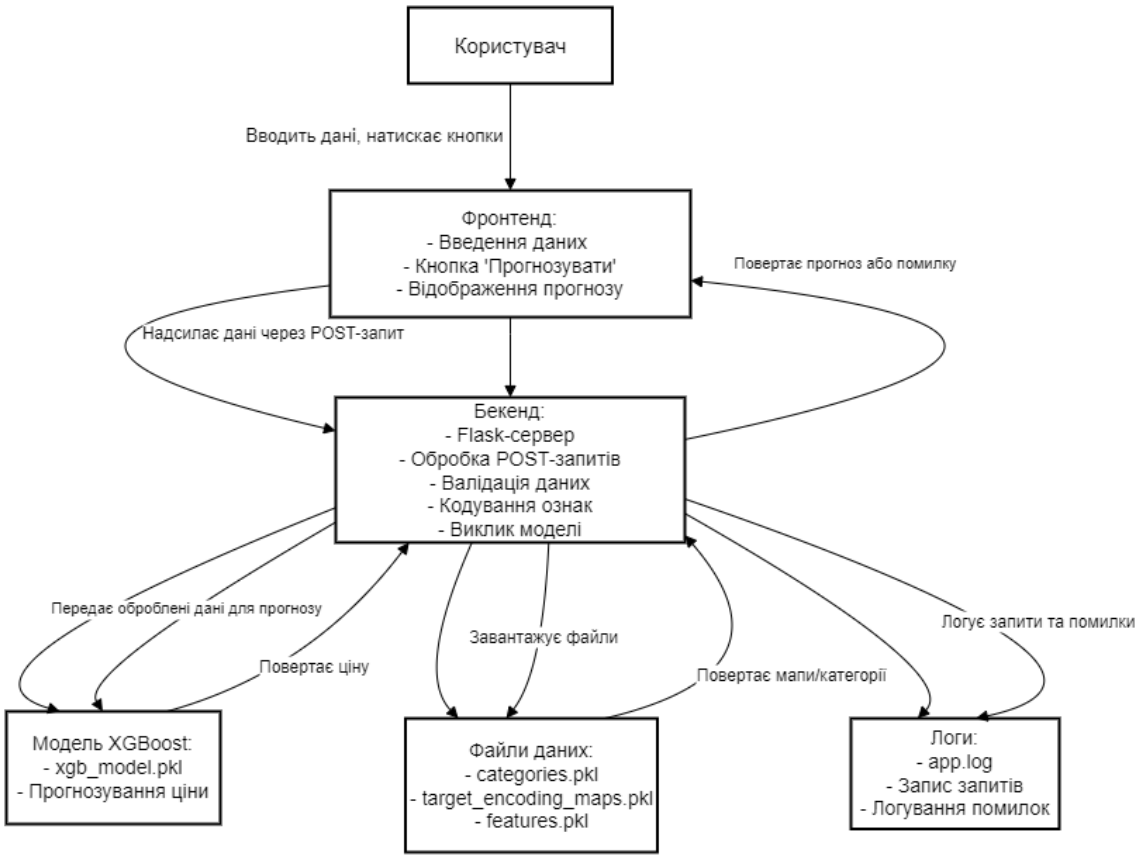


Рисунок 4.2.1 Алгоритм роботи застосунку

Для забезпечення надійності та контролю роботи системи було реалізовано логування всіх запитів і помилок у файл app.log (рис.4.2.2). Це дозволяє відстежувати дії користувачів, виявляти помилки (наприклад, некоректні значення в полях, зображені на рис.4.2.3) аналізувати роботу додатку для подальшого вдосконалення. Логування стало важливим інструментом для моніторингу продуктивності та стабільності системи під час її експлуатації.

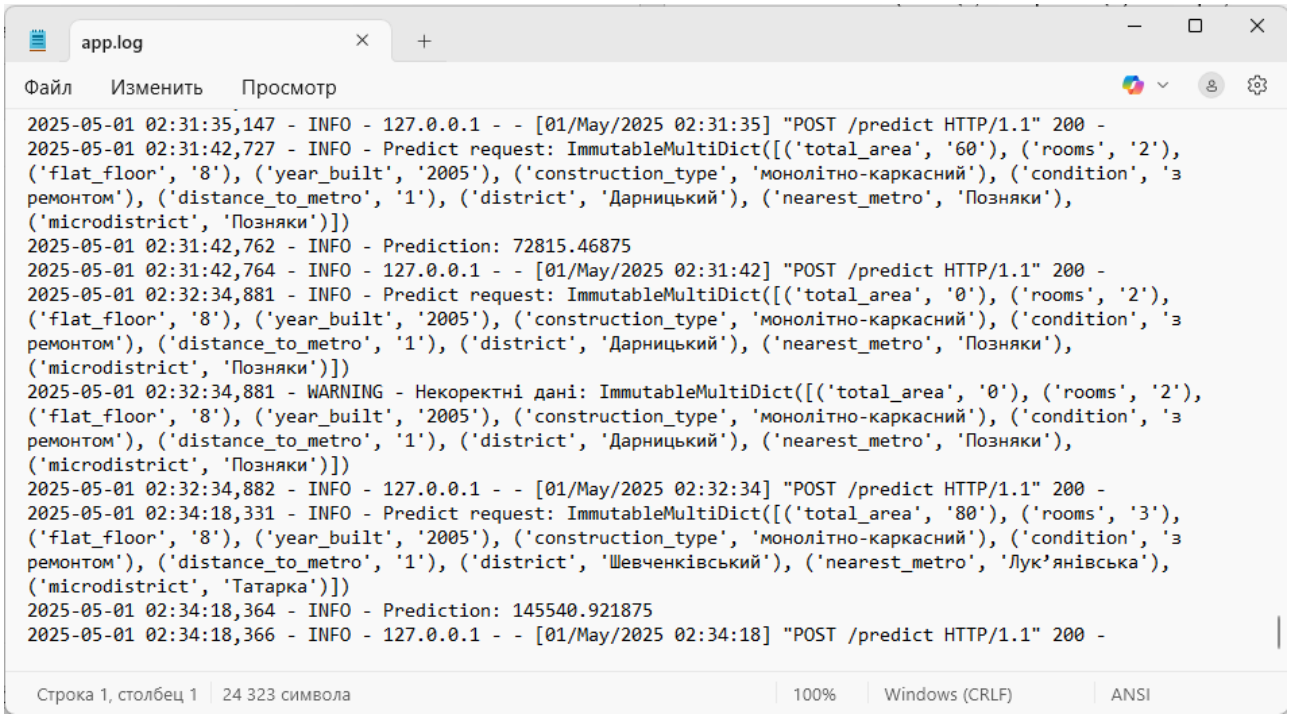


Рисунок 4.2.2 Логування запитів і помилок

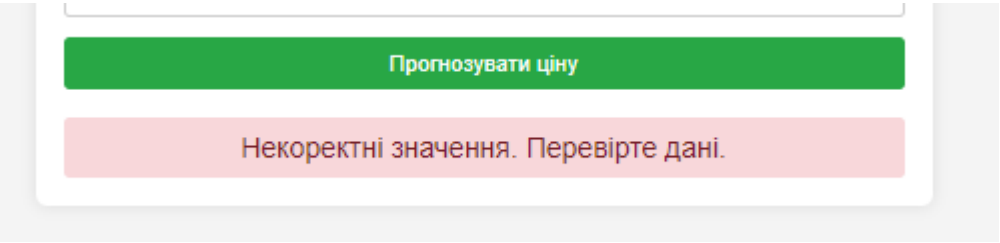


Рисунок 4.2.3 Відображення помилки

Переваги впровадження. Веб-додаток на Flask із моделлю XGBoost дозволяє швидко та точно оцінювати вартість нерухомості, зменшуючи залежність від суб'єктивних оцінок. Інтуїтивний інтерфейс і автоматизація підвищують зручність для користувачів, а можливість інтеграції через API забезпечує гнучкість для різних платформ. Таке рішення оптимізує процеси

оцінки, підтримує аналіз ринкових трендів і сприяє розвитку цифрових сервісів у сфері нерухомості.

Результати власного додатку. Розроблений веб-додаток на основі моделі XGBoost оцінив вартість 2-кімнатної квартири площею 60 м² у 72 815.47 USD, враховуючи параметри, такі як поверх (8), рік побудови (2005), тип будинку (панельний), стан (з ремонтом), відстань до метро (1 км), район, найближче метро та мікрорайон (рис.4.2.4).

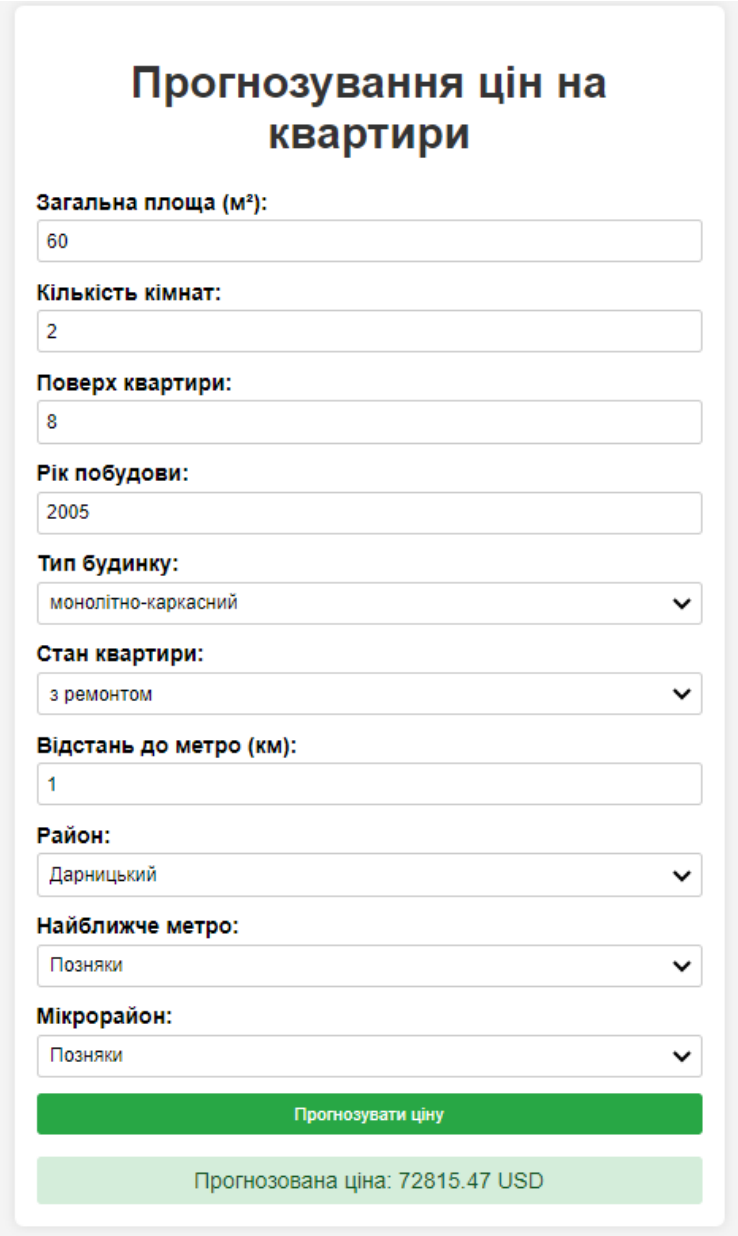


Рисунок 4.2.4 Прогнозування ціни на квартиру веб-додатком

Калькулятор DOM.RIA [22] проаналізувавши 94 пропозиції зі схожими параметрами, визначив середню ціну такої квартири на рівні 74165 USD. Різниця

між результатами становить близько 1350 USD, що свідчить про близькість прогнозів і підтверджує адекватність розробленої моделі.

4.3 Перспективи майбутніх досліджень

Перспективи майбутніх досліджень для розробленої моделі оцінки нерухомості на основі XGBoost та веб-додатку на Flask зосереджені на вдосконаленні функціональності, підвищенні точності прогнозів і розширенні сфер застосування. Одним із ключових напрямів є інтеграція моделі з різноманітними цифровими екосистемами, такими як платформи для оцінки нерухомості, фінансові сервіси чи аналітичні інструменти. Це дозволить використовувати модель як універсальний сервіс, що обробляє запити у форматі JSON і повертає прогнозовану ціну, автоматизуючи оцінку для користувачів, які аналізують ринкові дані або готують комерційні пропозиції.

Важливим напрямом розвитку є покращення самої моделі XGBoost шляхом експериментів із новими алгоритмами та підходами до обробки даних. Наприклад, можна дослідити використання ансамблевих методів, таких як стекінг або блендінг із моделями на основі нейронних мереж, для збільшення рівня правильності прогнозів у складних ринкових умовах. Також планується додавання нових джерел даних, зокрема економічних показників (інфляція, рівень доходів), індексів попиту, сезонності чи геопросторових даних (близькість до інфраструктури), що дозволить моделі краще враховувати зовнішні фактори.

Ще одним перспективним напрямом є використання інструментів природномовної обробки (NLP) для роботи з текстовими даними, пов’язаних із нерухомістю. NLP-технології дадуть змогу автоматично витягувати додаткові характеристики з описів об’єктів, таких як оголошення чи відгуки (наприклад, наявність дизайнерського ремонту, вид із вікна чи рівень шуму в районі), збагачуючи модель новими ознаками. Це не лише підвищить точність прогнозів,

а й зробить модель більш адаптивною до реальних ринкових умов і потреб користувачів.

Використання штучного інтелекту відкриває нові можливості для вдосконалення моделі оцінки нерухомості. Зокрема, ШІ може бути застосовано для створення адаптивних систем рекомендацій, які пропонуватимуть користувачам оптимальні об’єкти на основі їхніх вподобань і ринкових трендів. Крім того, методи глибинного навчання, такі як згорткові нейронні мережі, можуть використовуватися для аналізу візуальних даних (наприклад, фотографій об’єктів), що дозволить оцінювати стан ремонту чи естетичну привабливість автоматично. Інтеграція ШІ також сприятиме розробці чат-ботів для автоматизації спілкування з клієнтами, надаючи миттєві оцінки вартості та консультації в реальному часі.

Розвиток веб-додатку передбачає його масштабування для роботи з знчаними об’ємами запитів. Для цього планується розгортання на сервері з використанням сервера і контейнеризації через Docker, що забезпечить стабільність і доступність при високому навантаженні. Також розглядається створення мобільної версії додатку для iOS та Android, що зробить його зручнішим для користувачів, які працюють віддалено або в польових умовах.

Подальше вдосконалення користувацького досвіду передбачає додавання інтерактивних функцій, таких як візуалізація ринкових трендів, порівняння цін у різних регіонах чи збереження історії запитів. Це дозволить користувачам не лише отримувати прогнози, а й аналізувати динаміку цін для прийняття обґрунтованих рішень. Наприклад, інтеграція графіків зростання вартості нерухомості у певних районах може стати корисним інструментом для інвесторів.

Висновки

Розроблена модель оцінки нерухомості на основі алгоритму XGBoost та її реалізація у вигляді веб-додатку на фреймворку Flask демонструють значний потенціал для практичного застосування в різних секторах, включаючи

ріелторську діяльність, банківську сферу, страхування, державне управління та фінансові технології. Модель забезпечує швидку, точну та об’єктивну оцінку вартості квартир, що сприяє автоматизації бізнес-процесів, підвищенню прозорості ринку та оптимізації формувань вердиктів на основі даних. У банківській сфері модель ефективно застосовується для оцінки заставного майна, інтеграції в скорингові системи, підтримки іпотечного кредитування, управління кредитним портфелем та аналізу ринкових трендів, що знижує ризики та підвищує клієнтський досвід.

Впровадження моделі у веб-додаток із зручним інтерфейсом, валідацією даних, логуванням запитів і підтримкою категоріальних ознак забезпечує надійність і гнучкість рішення. Порівняння результатів додатку з ринковими даними, наприклад, калькулятором DOM.RIA, підтверджує високу достовірність прогнозів (різниця становить лише 1350 доларів США), що свідчить про адекватність моделі.

Перспективи майбутніх досліджень включають інтеграцію моделі з цифровими екосистемами, покращення алгоритмів за допомогою ансамблевих методів і нейронних мереж, додавання нових джерел даних (економічні показники, геопросторові дані), застосування NLP для аналізу текстових описів, штучного інтелекту,а також масштабування додатку через контейнеризацію та створення мобільної версії. Додаткові інтерактивні функції, такі як візуалізація ринкових трендів і порівняння цін, посилять аналітичні можливості системи, роблячи її універсальним інструментом для користувачів у сфері нерухомості.

ВИСНОВКИ

Розробка технології оцінки нерухомості методами науки про дані, представлена в цій кваліфікаційній роботі, довела свою актуальність і практичну цінність у контексті сучасних викликів ринку нерухомості. Дослідження зосереджено на автоматизації оцінки вартості житлової нерухомості вторинного ринку міста Києва з використанням методів науки про дані , зокрема алгоритму XGBoost, та подальшій інтеграції моделі у веб-додаток на базі Flask. Здійснений аналіз і результати дослідження підтверджують, що мету було досягнуто: підвищення якості прогнозу вартості об’єктів на ринку нерухомості шляхом розробки технології оцінки методами науки про дані.

У першому розділі здійснено аналіз теоретико-методологічних основ, який показав, що оцінка нерухомості є комплексним процесом, на який впливають різні фактори, як місцезнаходження, технічні характеристики, інфраструктура, економічні та політичні умови. Традиційні методи оцінки, попри їх поширеність, мають обмеження через суб’єктивність і недостатню здатність враховувати нелінійні залежності. Натомість методи науки про дані, зокрема машинне навчання, дозволяють обробляти великі обсяги даних і виявляти складні закономірності, що забезпечує вищу точність прогнозів. Методологія CRISP-DM, застосована в роботі, надала структурований підхід до реалізації проєкту, охоплюючи етапи збору даних, їх підготовки, моделювання, оцінки та розгортання.

Другий розділ було присвячено формалізації методів аналізу даних. Детально описано математичні основи лінійної регресії, випадкового лісу та XGBoost, а також проведено їх порівняння за інтерпретованістю, здатністю обробляти нелінійні залежності та обчислювальною ефективністю. Градієнтний бустинг виявився оптимальним завдяки адаптивності до складних даних. Описано технічні аспекти реалізації, зокрема використання Python, бібліотек scikit-learn, XGBoost і Flask, що забезпечило гнучкість і масштабованість системи.

У третьому розділі здійснено повний цикл аналізу даних. У процесі дослідження було зібрано та оброблено датасет із даними про квартири на вторинному ринку Києва, який включав характеристики об’єктів, географічні ознаки та ринкові показники. Проведено ретельну передобробку даних: очищення від викидів, заповнення пропусків, кодування категоріальних змінних і створення нових ознак, таких як взаємодії між площею та розташуванням. Для прогнозування цін використано три моделі: лінійну регресію, випадковий ліс і градієнтний бустінг (XGBoost). Оцінка якості моделей за метриками показала, що XGBoost має найвищу точність (MAPE 18.69% на тестовій вибірці), значно перевершуючи випадковий ліс (MAPE 26.96%) і лінійну регресію (MAPE 18.91%). Ключовими факторами, що впливають на ціноутворення, визначено загальну площу, розташування (зокрема престижні райони) і стан ремонту, що підтверджує важливість геопросторових і технічних характеристик.

Четвертий розділ представив розроблену модель XGBoost, що була інтегрована у веб-додаток на базі Flask, який забезпечує зручний інтерфейс для введення даних про квартиру та отримання прогнозованої вартості. Додаток включає валідацію даних, логування запитів і обробку категоріальних ознак, що робить його надійним і гнучким інструментом. Порівняння результатів додатку з калькулятором DOM.RIA показало різницю лише в 1350 доларів США для тестового прикладу, що підтверджує точність прогнозів. Впровадження моделі має широкий потенціал для використання в ріелторській діяльності, банківській сфері,страхуванні, державному управлінні та фінансових технологіях, сприяючи автоматизації процесів, зниженню суб’єктивізму та підвищенню прозорості ринку.

Перспективи подальших досліджень включають інтеграцію моделі з цифровими екосистемами, експерименти з ансамблевими методами та нейронними мережами, додавання нових джерел даних (економічні показники, геопросторові дані), застосування підходів NLP для аналізу текстових описів, штучного інтелекту, а також масштабування додатку через контейнеризацію та створення мобільної версії. Впровадження інтерактивних функцій, таких як

візуалізація ринкових трендів, посилить аналітичні можливості системи, роблячи її універсальним інструментом для учасників ринку нерухомості.

Таким чином, розроблена технологія підтвердила свою ефективність і створює підґрунтя для подальшого розвитку автоматизованих систем оцінки нерухомості, сприяючи цифровізації та підвищенню ефективності ринкових процесів.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Cross industry standard process for data mining [Електронний ресурс] // Вікіпедія вільна енциклопедія. – 2025. URL: https://en.wikipedia.org/wiki/Cross industry_standard_process_for_data_mining

2. Büyükkaracığan, Naci. MODERN METHODS APPROACH IN REAL ESTATE VALUATION, 2021. URL:https://iksadyayinevi.com/wp-content/uploads/2021/11/MODERN-METHODS-APPROACH-IN-REAL-ESTATE-VALUATION.pdf

3. Joey Hernandez, Danny Chang, Santiago Gutierrez, Paul Huggins. Predictive Analysis of Local House Prices: Leveraging Machine Learning for Real Estate Valuation. *SMU Data Science Review*. 2024. Vol. 8: No. 1, Article 12. URL: https://scholar.smu.edu/datasciencereview/vol8/iss1/12

4. Hoxha, V. Comparative analysis of machine learning models in predicting housing prices: a case study of Prishtina's real estate market. International Journal of Housing Markets and Analysis. 2025. Vol. 18 No. 3, pp. 694-711. DOI: https://doi.org/10.1108/IJHMA-09-2023-0120

5. Chuang Hu, Rui Huang, Haijian Li. Prediction and Analysis of Rental Price using Random Forest Machine Learning Technique Take Shanghai and Wuhan for example. 2023. DOI: 10.2991/978-94-6463-042-8_84

6. Ayten Yağmur1, Mehmet Kayakuş, Mustafa Terzioğlu. House price prediction modeling using machine learning techniques: A comparative study. 2022. DOI:10.36253/aestim-13703

7. Ahmet Yagmur. Real Estate Valuation Decision-Making System Using Machine Learning and Geospatial Data. 2025. URL: https://www.researchgate.net/publication/388272795_Real_Estate_Valuation_Decision-Making_System_Using_Machine_Learning_and_Geospatial_Data

8. Мірошниченко І. В., Крашеніннікова О. В., Прогнозування ціни на нерухомість з використанням алгоритмів машинного навчання. 2022. DOI: 10.32702/2307-2105-2022.1.81

9. AI in real estate property valuation. mDevelopers: вебсайт. URL: https://mdevelopers.com/blog/ai-real-estate-property-valuation

10. Thomas H. Root, Troy J. Strader, Yu-Hsiang (John) Huang. A Review of Machine Learning Approaches for Real Estate Valuation.2023.URL: https://jmwais.org/wp-content/uploads/sites/8/2023/07/V2023.I2.A2.pdf

11. RedFin.com: веб-сайт, URL: https://www.redfin.com/redfin-estimate

12. Zillow.com: веб-сайт, URL: https://www.zillow.com/

13. DOM.ria: веб-сайт, URL: https://dom.ria.com/uk/

14. Olx.ua: веб-сайт,URL: https://www.olx.ua/uk/

15. Lun.ua: веб-сайт,URL: https://lun.ua/uk/продаж-квартир-київ

16. Linear Regression in Machine learning. GeeksforGeeks: веб-сайт. URL: https://www.geeksforgeeks.org/linear-regression-formula/

17. Decision Trees and Random Forest (UA). Kaggle: веб-сайт. URL: https://www.kaggle.com/code/emstrakhov/decision-trees-and-random-forest-ua

18. Прохніцький Владислав. Побудова нейронної мережі на основі моделі випадкового лісу.2020. URL: https://archer.chnu.edu.ua/bitstream/handle/123456789/9808/math_2020_081%20%D0%9F%D1%80%D0%BE%D1%85%D0%BD%D1%96%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1

19. XGBoost - GeeksforGeeks: веб-сайт. URL: https://www.geeksforgeeks.org/xgboost/

20. What is Python? Executive Summary. Python: веб-сайт. URL: https://www.python.org/doc/essays/blurb/

21. ЛУН Статистика. Lun.ua: веб-сайт. URL: https://lun.ua/misto/stat/sale/kyiv?srsltid=AfmBOorHFgl61J5HwiTz68rOmfRxAkXNhaNJgnteaDPA-zvxg8M9UQL

22. Оцінка нерухомості. DIM.ria: веб-сайт. URL: https://dom.ria.com/uk/kalkuljator-stoimosti/prodazha kvartir/?res_id=6f41e50001dfc3fd09b3be93c5fb977b3ff0b069e57b9601

ДОДАТКИ

#ПАРСИНГ ДАНИХ

```
from typing import Any, Optional
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException, TimeoutException
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
import pandas as pd
import time

class RieltorSpider:
    CHROME_DRIVER_PATH = r'C:\Users\krolik\Untitled Folder\chromedriver.exe'
    URL = 'https://lun.ua/uk/продаж-квартир-київ'

    def __init__(self) -> None:
        self.driver = self._create_driver()
        self.wait = WebDriverWait(self.driver, 15)
        self.data = []
        self.parsed_ids = set()

    def _create_driver(self) -> webdriver.Chrome:
        service = Service(self.CHROME_DRIVER_PATH)
        options = Options()
        options.add_argument("--start-maximized")  # Відкривати браузер у повноекранному режимі
        return webdriver.Chrome(service=service, options=options)

    def run(self) -> None:
        self.start_request()
        self.collect_all_pages()
        self.save()
        self.driver.quit()

    def start_request(self) -> None:
        self.driver.get(self.URL)
        self.wait.until(EC.presence_of_element_located((By.CLASS_NAME, "realty-preview")))  # Чекаємо
першу карточку

    def collect_all_pages(self) -> None:
        while True:
            self.collect_cards()
            if not self.go_to_next_page():
                break

    def collect_cards(self) -> None:
        try:
            # Чекаємо, поки з'являться всі карточки
            cards = self.wait.until(EC.presence_of_all_elements_located(
                (By.CLASS_NAME, "realty-preview")
            ))
            print(f"Знайдено {len(cards)} карточок на сторінці.")
```

```
            for card in cards:
                data = self.parse_card(card)
                card_id = data.get('id', '0')
                if card_id not in self.parsed_ids:
                    self.parsed_ids.add(card_id)
                    self.data.append(data)
                    print(f"Додано карточку з ID: {card_id}")
                else:tpYT
                    print(f"Пропущено дублікат карточки з ID: {card_id}")

        except TimeoutException:
            print("Час очікування минув, карточки не знайдено.")

    def parse_card(self, card) -> dict:
        """Парсить максимум інформації з карточки квартири."""
        data = {}
        try:
            # ID карточки
            data['id'] = card.get_attribute('id') or "0"

            # Ціна та ціна за м²
            data['price'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-price--main').text.strip() if
card.find_elements(By.CSS_SELECTOR, '.realty-preview-price--main') else "0"
            data['price_per_sqm'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-price--
sqm').text.strip() if card.find_elements(By.CSS_SELECTOR, '.realty-preview-price--sqm') else "0"

            # Адреса
            data['address'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-title__link').text.strip() if
card.find_elements(By.CSS_SELECTOR, '.realty-preview-title__link') else "0"

            # ЖК, мікрорайон, район, місто
            sub_titles = card.find_elements(By.CSS_SELECTOR, '.realty-preview-sub-title')
            data['complex'] = sub_titles[0].text.strip() if len(sub_titles) > 0 else "0"
            data['microdistrict'] = sub_titles[1].text.strip() if len(sub_titles) > 1 else "0"
            data['district'] = sub_titles[2].text.strip() if len(sub_titles) > 2 else "0"
            data['city'] = sub_titles[3].text.strip() if len(sub_titles) > 3 else "0"

            # Основні характеристики
            properties = card.find_elements(By.CSS_SELECTOR, '.realty-preview-properties-item .realty-
preview-info')
            data['rooms'] = properties[0].text.strip() if len(properties) > 0 else "0"
            data['area'] = properties[1].text.strip() if len(properties) > 1 else "0"
            data['floor'] = properties[2].text.strip() if len(properties) > 2 else "0"
            data['condition'] = properties[3].text.strip() if len(properties) > 3 else "0"
            data['construction_type'] = properties[4].text.strip() if len(properties) > 4 else "0"
            data['year_built'] = properties[5].text.strip() if len(properties) > 5 else "0"

            # Опис
            data['description'] = card.find_element(By.CSS_SELECTOR, '.realty-preview-
description__text').text.strip() if card.find_elements(By.CSS_SELECTOR, '.realty-preview-
description__text') else "0"

            # Дати
            dates = card.find_elements(By.CSS_SELECTOR, '.realty-preview-dates__value')
            data['last_updated'] = dates[0].text.strip().replace("сьогодні о ", "").replace("вчора о ", "") if
len(dates) > 0 else "0"
            data['created'] = dates[1].text.strip().replace("Створено ", "") if len(dates) > 1 else "0"
```

```
            # Зображення
            data['image_url'] = card.find_element(By.CSS_SELECTOR, '.realty-preview__image
img').get_attribute('src') if card.find_elements(By.CSS_SELECTOR, '.realty-preview__image img') else "0"

        except NoSuchElementException as e:
            print(f"Елемент не знайдено в карточці з ID {data.get('id', '0')}: {e}")
        except Exception as e:
            print(f"Помилка під час парсингу карточки з ID {data.get('id', '0')}: {e}")

        return data

    def go_to_next_page(self) -> bool:
        """Переходить на наступну сторінку, якщо вона доступна."""
        try:
            next_button = self.wait.until(EC.element_to_be_clickable(
                (By.CSS_SELECTOR, '.paging-nav--right.paging-button.paging-nav:not(.disabled)')
            ))
            current_page = self.driver.find_element(By.CSS_SELECTOR, '.paging-button.active-link').text
            print(f"Поточна сторінка: {current_page}")

            next_button.click()
            self.wait.until(EC.staleness_of(next_button))
            self.wait.until(EC.presence_of_element_located((By.CLASS_NAME, "realty-preview")))
            new_page = self.driver.find_element(By.CSS_SELECTOR, '.paging-button.active-link').text
            print(f"Перейшли на сторінку: {new_page}")
            return True
        except TimeoutException:
            print("Наступна сторінка недоступна або час очікування минув. Завершення пагінації.")
            return False
        except NoSuchElementException:
            print("Кнопка наступної сторінки не знайдена. Завершення пагінації.")
            return False

    def save(self) -> None:
        """Зберігає дані у CSV-файл."""
        columns = [
            'id', 'price', 'price_per_sqm', 'address', 'complex', 'microdistrict', 'district', 'city',
            'rooms', 'area', 'floor', 'condition', 'construction_type', 'year_built', 'description',
            'last_updated', 'created', 'image_url'
        ]
        df = pd.DataFrame(self.data, columns=columns)
        df.to_csv('apartments_data.csv', encoding='utf-8', index=False)
        print(f"Дані збережено у apartments_data.csv. Всього зібрано {len(self.data)} записів.")

if __name__ == '__main__':
    spider = RieltorSpider()
    spider.run()

#XGBoost
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
import xgboost as xgb
import matplotlib.pyplot as plt
```

```
# Custom metric: Median Absolute Percentage Error
def median_absolute_percentage_error(actual: np.ndarray, predicted: np.ndarray):
    EPSILON = 1e-10
    return np.median(np.abs((actual - predicted) / (actual + EPSILON)))

# Load dataset
df = pd.read_csv('Apartments_final_metro.csv')

# Data preprocessing
df = df[df['price'].notnull() & (df['price'] != 0)]
df = df[(df['price'] >= 50000) & (df['price'] <= 400000)]

# Handle missing values
numeric_cols = ['year_built', 'rooms', 'total_area', 'living_area', 'not_living_area',
                'flat_floor', 'max_floor', 'distance_to_metro']
for col in numeric_cols:
    df[col] = df[col].fillna(df[col].median())

categorical_cols = ['address', 'complex', 'microdistrict', 'district',
                    'construction_type', 'nearest_metro', 'condition']
df['complex'] = df['complex'].fillna('no_complex')
df['microdistrict'] = df['microdistrict'].fillna(df['district'])
df['construction_type'] = df['construction_type'].replace('0', 'unknown').fillna('unknown')
df['condition'] = df['condition'].fillna('unknown')
df['nearest_metro'] = df['nearest_metro'].fillna('unknown')
df['city'] = df['city'].fillna('unknown')

# Save categorical columns for diagnostics before One-Hot Encoding
df_diagnostic = df[['complex', 'address', 'microdistrict']].copy()

# Remove rare categories
high_cardinality_cols = ['address', 'microdistrict', 'complex']
for col in high_cardinality_cols:
    counts = df[col].value_counts()
    rare_categories = counts[counts < 20].index  # Збільшено поріг до <20
    df[col] = df[col].replace(rare_categories, 'Other')

# Remove outliers
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

for col in ['total_area', 'living_area', 'distance_to_metro', 'price']:
    df = remove_outliers(df, col)

# Discretize distance_to_metro
bins = [0, 0.5, 1, 2, float('inf')]
labels = ['<0.5km', '0.5-1km', '1-2km', '2+km']
df['distance_to_metro_binned'] = pd.cut(df['distance_to_metro'], bins=bins, labels=labels,
include_lowest=True)

# One-Hot Encoding for low-cardinality variables
one_hot_cols = ['district', 'construction_type', 'condition', 'nearest_metro',  'distance_to_metro_binned']
```

```
df = pd.get_dummies(df, columns=one_hot_cols, prefix=one_hot_cols, drop_first=True)

# Split data with stratification
X = df.drop('price', axis=1)
y = df['price']
y_bins = pd.qcut(y, q=10, duplicates='drop')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_bins, random_state=42)

# Print dataset diagnostics
print("Розмір датасету:", df.shape)
print("Унікальні значення (до One-Hot Encoding):", df_diagnostic.nunique())
print("Розподіл цін (тренувальна):", y_train.describe())
print("Розподіл цін (тестова):", y_test.describe())
for col in high_cardinality_cols:
    train_categories = set(X_train[col])
    test_categories = set(X_test[col])
    print(f"Категорії в {col} у тесті, але не в тренуванні:", test_categories - train_categories)

# Target Encoding with leakage prevention
def target_encode_train_test(train_df, test_df, column, target, smoothing=30):
    global_mean = target.mean()
    temp_df = pd.DataFrame({column: train_df[column], 'target': target})
    group_means = temp_df.groupby(column)['target'].mean()
    group_counts = temp_df.groupby(column)['target'].count()
    smoothed_mean = (group_counts * group_means + smoothing * global_mean) / (group_counts +
smoothing)

    train_df[column + '_target_encoded'] = train_df[column].map(smoothed_mean)
    test_df[column + '_target_encoded'] = test_df[column].map(smoothed_mean).fillna(global_mean)
    return train_df, test_df

X_train = X_train.copy()
X_test = X_test.copy()
for col in high_cardinality_cols:
    X_train, X_test = target_encode_train_test(X_train, X_test, col, y_train)

# Add interaction features
X_train['area_condition_remont'] = X_train['total_area'] * X_train.get('condition_з ремонтом', 0)
X_test['area_condition_remont'] = X_test['total_area'] * X_test.get('condition_з ремонтом', 0)
X_train['area_district_pecherskyi'] = X_train['total_area'] * X_train.get('district_Печерський', 0)
X_test['area_district_pecherskyi'] = X_test['total_area'] * X_test.get('district_Печерський', 0)

# Define features
encoded_cols = [col + '_target_encoded' for col in high_cardinality_cols]
features = [col for col in X_train.columns if col in numeric_cols + encoded_cols or
col.startswith(tuple(one_hot_cols)) or col in ['area_condition_remont', 'area_district_pecherskyi']]
# Exclude distance_to_metro, living_area, and rooms
features = [col for col in features if col not in ['distance_to_metro', 'living_area', 'rooms']]
X_train = X_train[features]
X_test = X_test[features]

# Add sample weights for low-price samples
sample_weights = np.ones(len(y_train))
sample_weights[y_train < 75500] = 2.0  # Подвоєна вага для цін <25-го перцентиля

# XGBoost with GridSearchCV
estimator = xgb.XGBRegressor(random_state=42, enable_categorical=False)
```

```
parameters = {
    'learning_rate': [0.01, 0.05],
    'max_depth': [2, 3],
    'min_child_weight': [20, 30],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8],
    'n_estimators': [75, 100],
    'reg_lambda': [30.0, 40.0],
    'reg_alpha': [0.1, 1.0],
    'gamma': [0.5, 1.0]
}

xgb_grid = GridSearchCV(
    estimator,
    parameters,
    cv=3,
    scoring=['neg_mean_absolute_error', 'neg_mean_absolute_percentage_error'],
    refit='neg_mean_absolute_percentage_error',
    verbose=1,
    n_jobs=-1
)

xgb_grid.fit(X_train, y_train, sample_weight=sample_weights)

# Output CV results
print(f"CV Best neg_MAE:
{xgb_grid.cv_results_['mean_test_neg_mean_absolute_error'][xgb_grid.best_index_]:.5f}")
print(f"CV Best neg_MAPE:
{xgb_grid.cv_results_['mean_test_neg_mean_absolute_percentage_error'][xgb_grid.best_index_]:.5f}")
print(f"CV Best params: {xgb_grid.best_params_}")

# Train final model with early stopping
best_params = xgb_grid.best_params_
xgb_model = xgb.XGBRegressor(**best_params, random_state=42, early_stopping_rounds=30)
xgb_model.fit(X_train, y_train, sample_weight=sample_weights, eval_set=[(X_test, y_test)], verbose=False)

# Predictions
y_train_pred = xgb_model.predict(X_train)
y_test_pred = xgb_model.predict(X_test)

# Evaluate
train_mae = mean_absolute_error(y_train, y_train_pred)
train_mape = mean_absolute_percentage_error(y_train, y_train_pred)
train_mdae = np.median(np.abs(y_train - y_train_pred))
train_mdape = median_absolute_percentage_error(y_train, y_train_pred)

test_mae = mean_absolute_error(y_test, y_test_pred)
test_mape = mean_absolute_percentage_error(y_test, y_test_pred)
test_mdae = np.median(np.abs(y_test - y_test_pred))
test_mdape = median_absolute_percentage_error(y_test, y_test_pred)

print("\nTraining Metrics:")
print(f"Train MAE: {train_mae:.5f}, MdAE: {train_mdae:.5f}, MAPE: {train_mape:.5f}, MdAPE:
{train_mdape:.5f}")
print("\nTest Metrics:")
print(f"Test MAE: {test_mae:.5f}, MdAE: {test_mdae:.5f}, MAPE: {test_mape:.5f}, MdAPE:
{test_mdape:.5f}")
```

```
# Feature importance
feature_importance = pd.DataFrame(
    data=xgb_model.feature_importances_,
    index=xgb_model.feature_names_in_,
    columns=['importance']
).sort_values(by='importance', ascending=False)
print("\nFeature Importance (Top 10):\n", feature_importance.head(10))

# Analyze large errors
output_validation = pd.DataFrame(
    data={'price_predicted': y_test_pred, 'price_actual': y_test},
    index=X_test.index
)
output_validation['abs_error'] = np.abs(output_validation['price_predicted'] -
output_validation['price_actual'])
output_validation['percent_error'] = output_validation['abs_error'] / output_validation['price_actual']
print("\nTop 5 Largest Errors:\n", output_validation.sort_values(by='percent_error',
ascending=False).head())

# Save predictions
print("\nSample Predictions:\n", output_validation.head())

# Visualize
plt.figure(figsize=(10, 6))
plt.scatter(output_validation['price_actual'], output_validation['price_predicted'], alpha=0.5)
plt.plot([output_validation['price_actual'].min(), output_validation['price_actual'].max()],
         [output_validation['price_actual'].min(), output_validation['price_actual'].max()],
         color='black', linestyle='--')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices')
plt.show()

# Print features for problematic samples
problematic_ids = [1984, 759, 389]
print("\nFeatures for Problematic Samples:")
for idx in problematic_ids:
    if idx in X_test.index:
        print(f"ID {idx}:")
        print(X_test.loc[idx][['total_area', 'microdistrict_target_encoded', 'complex_target_encoded',
'condition_з ремонтом', 'district_Печерський']])plt.scatter(output_validation['price_actual'],
output_validation['price_predicted'], alpha=0.5)
plt.plot([output_validation['price_actual'].min(), output_validation['price_actual'].max()],
         [output_validation['price_actual'].min(), output_validation['price_actual'].max()],
         color='black', linestyle='--')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices')
plt.show()

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
```

```
# Custom metric: Median Absolute Percentage Error
def median_absolute_percentage_error(actual: np.ndarray, predicted: np.ndarray):
    EPSILON = 1e-10
    return np.median(np.abs((actual - predicted) / (actual + EPSILON)))

# Load dataset
df = pd.read_csv('Apartments_final_metro.csv')

# Save a copy of the original DataFrame for error analysis
df_original = df.copy()

# Data preprocessing
df = df[df['price'].notnull() & (df['price'] != 0)]
df = df[(df['price'] >= 50000) & (df['price'] <= 400000)]

# Handle missing values
numeric_cols = ['year_built', 'rooms', 'total_area', 'living_area', 'not_living_area',
                'flat_floor', 'max_floor', 'distance_to_metro']
for col in numeric_cols:
    df[col] = df[col].fillna(df[col].median())

categorical_cols = ['address', 'complex', 'microdistrict', 'district', 'city',
                    'construction_type', 'nearest_metro', 'condition']
df['complex'] = df['complex'].fillna('no_complex')
df['microdistrict'] = df['microdistrict'].fillna(df['district'])
df['construction_type'] = df['construction_type'].replace('0', 'unknown').fillna('unknown')
df['condition'] = df['condition'].fillna('unknown')
df['nearest_metro'] = df['nearest_metro'].fillna('unknown')
df['city'] = df['city'].fillna('unknown')

# Remove rare categories
high_cardinality_cols = ['address', 'microdistrict', 'complex']
for col in high_cardinality_cols:
    counts = df[col].value_counts()
    if col == 'address' or col == 'complex':
        rare_categories = counts[counts < 5].index  # Зменшено до < 5
    else:
        rare_categories = counts[counts < 15].index  # Залишено < 15 для microdistrict
    df[col] = df[col].replace(rare_categories, 'Other')

# Remove outliers
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

for col in ['total_area', 'living_area', 'distance_to_metro', 'price']:
    df = remove_outliers(df, col)

# Discretize distance_to_metro
bins = [0, 0.5, 1, 2, float('inf')]
labels = ['<0.5km', '0.5-1km', '1-2km', '2+km']
```

```
df['distance_to_metro_binned'] = pd.cut(df['distance_to_metro'], bins=bins, labels=labels,
include_lowest=True)

# Add interaction features
df['area_x_district'] = df['total_area'] * df['district'].astype('category').cat.codes
df['condition_x_district'] = df['condition'].astype('category').cat.codes *
df['district'].astype('category').cat.codes
df['area_x_condition'] = df['total_area'] * df['condition'].astype('category').cat.codes
df['distance__district'] = df['distance_to_metro'] * df['district'].astype('category').cat.codes

# One-Hot Encoding for low-cardinality variables
one_hot_cols = ['district', 'construction_type', 'condition', 'nearest_metro', 'city', 'distance_to_metro_binned']
df = pd.get_dummies(df, columns=one_hot_cols, prefix=one_hot_cols, drop_first=True)

# Split data with stratification
X = df.drop('price', axis=1)
y = df['price']
y_bins = pd.qcut(y, q=10, duplicates='drop')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_bins, random_state=42)

# Target Encoding with leakage prevention
def target_encode_train_test(train_df, test_df, column, target, smoothing=10):
    global_mean = target.mean()
    temp_df = pd.DataFrame({column: train_df[column], 'target': target})
    group_means = temp_df.groupby(column)['target'].mean()
    group_counts = temp_df.groupby(column)['target'].count()
    smoothed_mean = (group_counts * group_means + smoothing * global_mean) / (group_counts +
smoothing)

    train_df[column + '_target_encoded'] = train_df[column].map(smoothed_mean)
    test_df[column + '_target_encoded'] = test_df[column].map(smoothed_mean).fillna(global_mean)
    return train_df, test_df

X_train = X_train.copy()
X_test = X_test.copy()
for col in high_cardinality_cols:
    X_train, X_test = target_encode_train_test(X_train, X_test, col, y_train)

# Define features
encoded_cols = [col + '_target_encoded' for col in high_cardinality_cols]
features = [col for col in X_train.columns if col in numeric_cols + encoded_cols + ['area_x_district',
'condition_x_district', 'area_x_condition', 'distance_x_district'] or col.startswith(tuple(one_hot_cols))]
X_train = X_train[features]
X_test = X_test[features]

# Check data size
print(f"Кількість рядків для навчання: {len(X_train)}")
print(f"Кількість рядків для тестування: {len(X_test)}")
print(f"Загальна кількість рядків: {len(df)}")

# Random Forest with GridSearchCV
estimator = RandomForestRegressor(random_state=42)
parameters = {
    'n_estimators': [100, 200],
    'max_depth': [5, 7],
    'min_samples_split': [5, 10],
    'min_samples_leaf': [2, 4],
```

```
    'max_features': ['sqrt']
}

rf_grid = GridSearchCV(
    estimator,
    parameters,
    cv=5,
    scoring=['neg_mean_absolute_error', 'neg_mean_absolute_percentage_error'],
    refit='neg_mean_absolute_percentage_error',
    verbose=1,
    n_jobs=-1
)

rf_grid.fit(X_train, y_train)

# Output CV results
print(f"CV Best neg_MAE:
{rf_grid.cv_results_['mean_test_neg_mean_absolute_error'][rf_grid.best_index_]:.5f}")
print(f"CV Best neg_MAPE:
{rf_grid.cv_results_['mean_test_neg_mean_absolute_percentage_error'][rf_grid.best_index_]:.5f}")
print(f"CV Best params: {rf_grid.best_params_}")

# Train final model
best_params = rf_grid.best_params_
rf_model = RandomForestRegressor(**best_params, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
y_train_pred = rf_model.predict(X_train)
y_test_pred = rf_model.predict(X_test)

# Evaluate
train_mae = mean_absolute_error(y_train, y_train_pred)
train_mape = mean_absolute_percentage_error(y_train, y_train_pred)
train_mdae = np.median(np.abs(y_train - y_train_pred))
train_mdape = median_absolute_percentage_error(y_train, y_train_pred)

test_mae = mean_absolute_error(y_test, y_test_pred)
test_mape = mean_absolute_percentage_error(y_test, y_test_pred)
test_mdae = np.median(np.abs(y_test - y_test_pred))
test_mdape = median_absolute_percentage_error(y_test, y_test_pred)

print("\nTraining Metrics:")
print(f"Train MAE: {train_mae:.5f}, MdAE: {train_mdae:.5f}, MAPE: {train_mape:.5f}, MdAPE:
{train_mdape:.5f}")
print("\nTest Metrics:")
print(f"Test MAE: {test_mae:.5f}, MdAE: {test_mdae:.5f}, MAPE: {test_mape:.5f}, MdAPE:
{test_mdape:.5f}")

# Feature importance
feature_importance = pd.DataFrame(
    data=rf_model.feature_importances_,
    index=rf_model.feature_names_in_,
    columns=['importance']
).sort_values(by='importance', ascending=False)
print("\nFeature Importance (Top 10):\n", feature_importance.head(10))
```

```
# Save predictions
output_validation = pd.DataFrame(
    data={'price_predicted': y_test_pred, 'price_actual': y_test},
    index=X_test.index
)
print("\nSample Predictions:\n", output_validation.head())

# Error analysis
output_validation['error'] = abs(output_validation['price_predicted'] - output_validation['price_actual'])
print("\nТоп-5 прогнозів із найбільшими помилками:\n", output_validation.sort_values('error',
ascending=False).head())

error_analysis_cols = ['price', 'total_area', 'distance_to_metro', 'nearest_metro', 'district', 'condition', 'address',
'microdistrict']
top_error_rows = output_validation.sort_values('error', ascending=False).head(10).index
print("\nХарактеристики рядків із найбільшими помилками:\n", df_original.loc[top_error_rows,
error_analysis_cols])
print("\nЧастка 'Other' в address:", (df_original.loc[top_error_rows, 'address'] == 'Other').mean())
print("Частка 'Other' в microdistrict:", (df_original.loc[top_error_rows, 'microdistrict'] == 'Other').mean())

# Check proportion of 'Other'
for col in high_cardinality_cols:
    print(f"Частка 'Other' у {col}: {(df[col] == 'Other').mean():.3f}")
    print(f"Кількість унікальних значень у {col}: {df[col].nunique()}")

# Check price distribution
print("\nРозподіл цін у тренувальній вибірці:\n", y_train.describe())
print("\nРозподіл цін у тестовій вибірці:\n", y_test.describe())

# Visualize price distribution
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(y_train, bins=30, alpha=0.5, label='Train')
plt.title('Train Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend()

plt.subplot(1, 2, 2)
plt.hist(y_test, bins=30, alpha=0.5, label='Test')
plt.title('Test Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend()
plt.tight_layout()
plt.show()

# Visualize predictions
plt.figure(figsize=(10, 6))
plt.scatter(output_validation['price_actual'], output_validation['price_predicted'], alpha=0.5)
plt.plot([output_validation['price_actual'].min(), output_validation['price_actual'].max()],
        [output_validation['price_actual'].min(), output_validation['price_actual'].max()],
        color='black', linestyle='--')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices (Random Forest)')
plt.show()
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Custom metric: Median Absolute Percentage Error
def median_absolute_percentage_error(actual: np.ndarray, predicted: np.ndarray):
    EPSILON = 1e-10
    return np.median(np.abs((actual - predicted) / (actual + EPSILON)))

# Load dataset
df = pd.read_csv('Apartments_final_metro.csv')

# Save a copy of the original DataFrame for error analysis
df_original = df.copy()

# Data preprocessing
df = df[df['price'].notnull() & (df['price'] != 0)]
df = df[(df['price'] >= 50000) & (df['price'] <= 400000)]

# Handle missing values
numeric_cols = ['year_built', 'rooms', 'total_area', 'living_area', 'not_living_area',
                'flat_floor', 'max_floor', 'distance_to_metro']
for col in numeric_cols:
    df[col] = df[col].fillna(df[col].median())

categorical_cols = ['address', 'complex', 'microdistrict', 'district', 'city',
                    'construction_type', 'nearest_metro', 'condition']
df['complex'] = df['complex'].fillna('no_complex')
df['microdistrict'] = df['microdistrict'].fillna(df['district'])
df['construction_type'] = df['construction_type'].replace('0', 'unknown').fillna('unknown')
df['condition'] = df['condition'].fillna('unknown')
df['nearest_metro'] = df['nearest_metro'].fillna('unknown')
df['city'] = df['city'].fillna('unknown')

# Remove rare categories
high_cardinality_cols = ['address', 'microdistrict', 'complex']
for col in high_cardinality_cols:
    counts = df[col].value_counts()
    if col == 'address' or col == 'complex':
        rare_categories = counts[counts < 5].index
    else:
        rare_categories = counts[counts < 15].index
    df[col] = df[col].replace(rare_categories, 'Other')

# Remove outliers
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
```

```
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

for col in ['total_area', 'living_area', 'distance_to_metro', 'price']:
    df = remove_outliers(df, col)

# Discretize distance_to_metro
bins = [0, 0.5, 1, 2, float('inf')]
labels = ['<0.5km', '0.5-1km', '1-2km', '2+km']
df['distance_to_metro_binned'] = pd.cut(df['distance_to_metro'], bins=bins, labels=labels,
include_lowest=True)

# Add interaction features
df['area_x_district'] = df['total_area'] * df['district'].astype('category').cat.codes
df['condition_x_district'] = df['condition'].astype('category').cat.codes *
df['district'].astype('category').cat.codes
df['area_x_condition'] = df['total_area'] * df['condition'].astype('category').cat.codes
df['distance_x_district'] = df['distance_to_metro'] * df['district'].astype('category').cat.codes

# One-Hot Encoding for low-cardinality variables
one_hot_cols = ['district', 'construction_type', 'condition', 'nearest_metro', 'city', 'distance_to_metro_binned']
df = pd.get_dummies(df, columns=one_hot_cols, prefix=one_hot_cols, drop_first=True)

# Split data with stratification
X = df.drop('price', axis=1)
y = df['price']
y_bins = pd.qcut(y, q=10, duplicates='drop')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_bins, random_state=42)

# Target Encoding with leakage prevention
def target_encode_train_test(train_df, test_df, column, target, smoothing=10):
    global_mean = target.mean()
    temp_df = pd.DataFrame({column: train_df[column], 'target': target})
    group_means = temp_df.groupby(column)['target'].mean()
    group_counts = temp_df.groupby(column)['target'].count()
    smoothed_mean = (group_counts * group_means + smoothing * global_mean) / (group_counts +
smoothing)

    train_df[column + '_target_encoded'] = train_df[column].map(smoothed_mean)
    test_df[column + '_target_encoded'] = test_df[column].map(smoothed_mean).fillna(global_mean)
    return train_df, test_df

X_train = X_train.copy()
X_test = X_test.copy()
for col in high_cardinality_cols:
    X_train, X_test = target_encode_train_test(X_train, X_test, col, y_train)

# Define features
encoded_cols = [col + '_target_encoded' for col in high_cardinality_cols]
features = [col for col in X_train.columns if col in numeric_cols + encoded_cols + ['area_x_district',
'condition_x_district', 'area_x_condition', 'distance_x_district'] or col.startswith(tuple(one_hot_cols))]
X_train = X_train[features]
X_test = X_test[features]

# Standardize numeric features
scaler = StandardScaler()
X_train_scaled = X_train.copy()
```

```
X_test_scaled = X_test.copy()
X_train_scaled[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test_scaled[numeric_cols] = scaler.transform(X_test[numeric_cols])

# Check data size
print(f"Кількість рядків для навчання: {len(X_train)}")
print(f"Кількість рядків для тестування: {len(X_test)}")
print(f"Загальна кількість рядків: {len(df)}")

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = lr_model.predict(X_train_scaled)
y_test_pred = lr_model.predict(X_test_scaled)

# Evaluate
train_mae = mean_absolute_error(y_train, y_train_pred)
train_mape = mean_absolute_percentage_error(y_train, y_train_pred)
train_mdae = np.median(np.abs(y_train - y_train_pred))
train_mdape = median_absolute_percentage_error(y_train, y_train_pred)

test_mae = mean_absolute_error(y_test, y_test_pred)
test_mape = mean_absolute_percentage_error(y_test, y_test_pred)
test_mdae = np.median(np.abs(y_test - y_test_pred))
test_mdape = median_absolute_percentage_error(y_test, y_test_pred)

print("\nTraining Metrics:")
print(f"Train MAE: {train_mae:.5f}, MdAE: {train_mdae:.5f}, MAPE: {train_mape:.5f}, MdAPE:
{train_mdape:.5f}")
print("\nTest Metrics:")
print(f"Test MAE: {test_mae:.5f}, MdAE: {test_mdae:.5f}, MAPE: {test_mape:.5f}, MdAPE:
{test_mdape:.5f}")

# Feature importance (coefficients)
feature_importance = pd.DataFrame(
    data=np.abs(lr_model.coef_),
    index=X_train_scaled.columns,
    columns=['coefficient']
).sort_values(by='coefficient', ascending=False)
print("\nFeature Importance (Top 10, Absolute Coefficients):\n", feature_importance.head(10))

# Save predictions
output_validation = pd.DataFrame(
    data={'price_predicted': y_test_pred, 'price_actual': y_test},
    index=X_test.index
)
print("\nSample Predictions:\n", output_validation.head())

# Error analysis
output_validation['error'] = abs(output_validation['price_predicted'] - output_validation['price_actual'])
print("\nТоп-5 прогнозів із найбільшими помилками:\n", output_validation.sort_values('error',
ascending=False).head())

error_analysis_cols = ['price', 'total_area', 'distance_to_metro', 'nearest_metro', 'district', 'condition', 'address',
'microdistrict']
```

```
top_error_rows = output_validation.sort_values('error', ascending=False).head(10).index
print("\nХарактеристики рядків із найбільшими помилками:\n", df_original.loc[top_error_rows,
error_analysis_cols])
print("\nЧастка 'Other' в address:", (df_original.loc[top_error_rows, 'address'] == 'Other').mean())
print("Частка 'Other' в microdistrict:", (df_original.loc[top_error_rows, 'microdistrict'] == 'Other').mean())

# Check proportion of 'Other'
for col in high_cardinality_cols:
    print(f"Частка 'Other' у {col}: {(df[col] == 'Other').mean():.3f}")
    print(f"Кількість унікальних значень у {col}: {df[col].nunique()}")

# Check price distribution
print("\nРозподіл цін у тренувальній вибірці:\n", y_train.describe())
print("\nРозподіл цін у тестовій вибірці:\n", y_test.describe())

# Visualize price distribution
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(y_train, bins=30, alpha=0.5, label='Train')
plt.title('Train Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend()

plt.subplot(1, 2, 2)
plt.hist(y_test, bins=30, alpha=0.5, label='Test')
plt.title('Test Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend()
plt.tight_layout()
plt.show()

# Visualize predictions
plt.figure(figsize=(10, 6))
plt.scatter(output_validation['price_actual'], output_validation['price_predicted'], alpha=0.5)
plt.plot([output_validation['price_actual'].min(), output_validation['price_actual'].max()],
        [output_validation['price_actual'].min(), output_validation['price_actual'].max()],
        color='black', linestyle='--')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices (Linear Regression)')
plt.show()
```