

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту**

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«__» _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

**на тему: «Інтелектуальна система для оцінки та прогнозування ринкової
вартості нерухомості з урахуванням часових трендів»**

Виконав:

студент IV курсу, групи КІ-22

Рудюк Дмитро Анатолійович _____

Керівник:

професорка кафедри математичних

методів системного аналізу, д.т.н., професор

Недашківська Надія Іванівна. _____

Консультант з нормоконтролю:

фахівець першої категорії кафедри

штучного інтелекту, к.т.н., доцент,

Комариста Богдана Миколаївна _____

Рецензент:

професор кафедри математичних

методів системного аналізу, д.т.н., професор

Бідюк Петро Іванович _____

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«15» січня 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Рудюка Дмитра Анатолійовича

1. Тема роботи «Інтелектуальна система оцінки та прогнозування ринкової вартості нерухомості з урахуванням часових трендів», керівник роботи Недашківська Надія Іванівна, доцент кафедри ММСА, затверджені наказом по НН ІПСА від «27» травня 2026 р. № 1944-с.
2. Термін подання студентом роботи «05» червня 2026 року.
3. Вихідні дані до роботи: набір даних про продаж житлової нерухомості King County House Sales, що містить характеристики об'єктів нерухомості, інформацію про їх розташування, технічні параметри, дату продажу та ринкову вартість; сучасні методи машинного навчання та аналізу часових рядів; програмні засоби Python, бібліотеки Scikit-learn, XGBoost, CatBoost, LightGBM, TensorFlow та Keras.
4. Зміст роботи: аналіз предметної області оцінювання та прогнозування вартості нерухомості, дослідження факторів впливу на ринкову ціну об'єктів, підготовка та обробка набору даних, вибір і реалізація моделей машинного навчання для прогнозування вартості нерухомості, розробка моделі аналізу часових трендів, програмна реалізація інтелектуальної системи, оцінювання якості моделей, порівняння отриманих результатів та аналіз ефективності розробленої системи.

5. Перелік ілюстративного матеріалу: електронна презентація.

6. Дата видачі завдання 18.02.2026

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	20.04.2026	Виконано
2	Підготовка першого розділу	28.04.2026	Виконано
3	Підготовка другого розділу	05.05.2026	Виконано
4	Розробка програмного продукту	12.05.2026	Виконано
5	Підготовка третього розділу	19.05.2026	Виконано
6	Оформлення дипломної роботи	02.06.2026	Виконано
7	Підготовка презентації доповіді	05.06.2026	Виконано

Студент

Дмитро РУДЮК

Керівник

Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота:

НЕРУХОМІСТЬ, РИНКОВА ВАРТІСТЬ, МАШИННЕ НАВЧАННЯ,
ПРОГНОЗУВАННЯ ЦІН, ЧАСОВІ ТРЕНДИ, XGBOOST, CATBOOST,
LIGHTGBM, LSTM, PYTHON.

Об'єктом дослідження є процес оцінки та прогнозування ринкової вартості житлової нерухомості із застосуванням методів машинного навчання.

Предметом дослідження є моделі машинного навчання та часових рядів для прогнозування вартості нерухомості на основі технічних, просторових, якісних і часових характеристик об'єктів.

Метою роботи є розробка інтелектуальної системи для оцінки та прогнозування ринкової вартості нерухомості з урахуванням часових трендів.

Актуальність дослідження зумовлена динамічністю ринку нерухомості, значною кількістю факторів, що впливають на формування ціни, та потребою в автоматизованих інструментах оцінювання житла. У роботі проаналізовано предметну область оцінки нерухомості, основні фактори впливу на її ринкову вартість, особливості використаних даних і сучасні методи прогнозування. Розглянуто регресійні моделі, ансамблеві методи градієнтного бустингу, методи аналізу часових рядів і рекурентні нейронні мережі. Обґрунтовано використання Linear Regression, XGBoost, CatBoost, LightGBM, ARIMA та LSTM.

Розроблено програмний продукт мовою Python, який дає змогу прогнозувати вартість нерухомості за параметрами об'єкта, враховувати часові тенденції ринку, порівнювати результати різних моделей і візуалізувати отримані прогнози. Отримані результати підтверджують практичну цінність системи як допоміжного інструменту для покупців, продавців, аналітиків ринку нерухомості та фахівців з оцінювання майна.

ABSTRACT

Master's thesis:

REAL ESTATE, MARKET VALUE, MACHINE LEARNING, PRICE FORECASTING, TIME SERIES TRENDS, XGBOOST, CATBOOST, LIGHTGBM, LSTM, PYTHON.

The object of the study is the process of assessing and forecasting the market value of residential real estate using machine learning methods.

The subject of the study is machine learning and time series models for forecasting real estate value based on the technical, spatial, qualitative, and temporal characteristics of properties.

The aim of the work is to develop an intelligent system for assessing and forecasting the market value of real estate, taking into account time trends.

The relevance of the research is determined by the dynamism of the real estate market, the significant number of factors influencing price formation, and the need for automated housing valuation tools. The paper analyzes the subject area of real estate valuation, the main factors influencing its market value, the characteristics of the data used, and modern forecasting methods. Regression models, ensemble gradient boosting methods, time series analysis methods, and recurrent neural networks are considered. The use of Linear Regression, XGBoost, CatBoost, LightGBM, ARIMA, and LSTM is justified.

A software product was developed in Python that allows for forecasting real estate value based on property parameters, accounting for market trends over time, comparing the results of different models, and visualizing the obtained forecasts. The results confirm the practical value of the system as a tool for buyers, sellers, real estate market analysts, and property appraisers.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	8
ВСТУП.....	10
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ОБЛАСТІ ПРОГНОЗУВАННЯ ВАРТОСТІ НЕРУХОМОСТІ	12
1.1 Аналіз предметної області прогнозування вартості нерухомості...	12
1.2 Методи прогнозування вартості нерухомості	15
1.3 Аналіз ознак, що впливають на вартість нерухомості	17
1.4 Постановка задачі дослідження	21
Висновки до розділу 1	23
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА ПІДГОТОВКА ДАНИХ ДЛЯ МОДЕЛЕЙ ПРОГНОЗУВАННЯ	24
2.1 Опис вхідного набору даних	24
2.2 Попередня обробка даних.....	26
2.3 Формування додаткових ознак	29
2.4 Математичне забезпечення системи прогнозування.....	33
2.5 Критерії оцінювання якості моделей.....	40
2.6 Поділ даних та підготовка вибірок для навчання моделей.....	43
Висновки до розділу 2.....	45
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ	46
3.1 Вибір засобів програмної реалізації	46
3.2 Архітектура програмної системи.....	47
3.3 Реалізація моделей прогнозування	50
3.4 Реалізація LSTM-моделі	53
3.5 Використані програмні бібліотеки та засоби розробки	56
Висновки до розділу 3.....	58
РОЗДІЛ 4 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ	60
4.1 Порівняння якості моделей	60
4.2 Аналіз похибок прогнозування	62

4.3 Аналіз важливості ознак.....	64
4.4 Аналіз результатів LSTM-моделі	68
Висновки до розділу 4.....	72
ВИСНОВКИ.....	74
ПЕРЕЛІК ПОСИЛАНЬ	76
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....	78
ДОДАТОК Б ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ.....	106

ПЕРЕЛІК СКОРОЧЕНЬ

MAE — Mean Absolute Error (середня абсолютна похибка)

MSE — Mean Squared Error (середньоквадратична похибка)

RMSE — Root Mean Squared Error (корінь середньоквадратичної похибки)

MAPE — Mean Absolute Percentage Error (середня абсолютна відсоткова похибка)

sMAPE — Symmetric Mean Absolute Percentage Error (симетрична середня абсолютна відсоткова похибка)

R² — Coefficient of Determination (коефіцієнт детермінації)

ML — Machine Learning (машинне навчання)

AI — Artificial Intelligence (штучний інтелект)

LSTM — Long Short-Term Memory (довга короткострокова пам'ять)

XGBoost — Extreme Gradient Boosting (екстремальний градієнтний бустинг)

CatBoost — Categorical Boosting (градієнтний бустинг для категоріальних даних)

LightGBM — Light Gradient Boosting Machine (легкий градієнтний бустинг)

API — Application Programming Interface (програмний інтерфейс застосунків)

CSV — Comma-Separated Values (значення, розділені комами)

JSON — JavaScript Object Notation (формат обміну даними)

pandas — бібліотека для обробки та аналізу даних

NumPy — бібліотека для наукових обчислень

Scikit-learn — бібліотека машинного навчання

TensorFlow — бібліотека для глибокого навчання нейронних мереж

Keras — високорівневий API для нейронних мереж

ARIMA — AutoRegressive Integrated Moving Average (авторегресійна інтегрована модель ковзного середнього)

CTR — Click-Through Rate (коефіцієнт клікабельності)

ADAM — Adaptive Moment Estimation (метод адаптивної оцінки моментів)

ВСТУП

Ринок¹ нерухомості є однією з важливих складових сучасної економіки та характеризується значною кількістю факторів, які впливають на формування вартості об'єктів. Ціна житлової нерухомості залежить від площі будинку, кількості кімнат, місця розташування, року побудови, технічного стану, рівня розвитку інфраструктури та багатьох інших характеристик. У зв'язку зі зростанням обсягів доступних даних традиційні методи оцінювання нерухомості поступово доповнюються сучасними інформаційними технологіями та методами аналізу даних.

Одним із перспективних напрямів розвитку систем підтримки прийняття рішень є використання технологій штучного інтелекту та машинного навчання. Сучасні алгоритми дозволяють виявляти приховані закономірності у великих наборах даних та будувати моделі прогнозування з високою точністю. Особливу ефективність у задачах регресійного аналізу демонструють ансамблеві алгоритми градієнтного бустингу, зокрема XGBoost, CatBoost та LightGBM, які широко використовуються для роботи з табличними даними [2–4]. Для аналізу часових залежностей і прогнозування тенденцій зміни показників застосовуються рекурентні нейронні мережі типу LSTM [5].

Актуальність теми дипломної роботи обумовлена необхідністю підвищення точності оцінювання вартості нерухомості та автоматизації процесу прийняття рішень на основі аналізу даних. Використання методів машинного навчання дозволяє враховувати складні взаємозв'язки між

¹ Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПП ім. Ігоря Сікорського (протокол №11 Вченої ради КПП ім. Ігоря Сікорського від 11 грудня 2023 р.).

характеристиками об'єктів нерухомості та їхньою ринковою вартістю, що сприяє отриманню більш точних прогнозів порівняно з традиційними підходами.

Об'єктом дослідження є процес прогнозування вартості житлової нерухомості на основі історичних даних та характеристик об'єктів.

Предметом дослідження є методи машинного навчання та аналізу часових рядів, що використовуються для прогнозування вартості нерухомості.

Метою роботи є розробка інтелектуальної системи прогнозування вартості нерухомості із застосуванням сучасних методів машинного навчання.

Для досягнення поставленої мети необхідно вирішити такі завдання: провести аналіз предметної області та існуючих підходів до прогнозування вартості нерухомості; дослідити набір даних та виконати його попередню обробку; реалізувати моделі Linear Regression, XGBoost, CatBoost та LightGBM; реалізувати LSTM-модель для аналізу часової динаміки цін; виконати оцінювання якості моделей за допомогою системи метрик та провести порівняльний аналіз отриманих результатів.

Практичне значення роботи полягає у створенні програмної системи, яка дозволяє автоматизувати процес оцінювання нерухомості та формувати прогнози її вартості на основі характеристик об'єктів. Результати дослідження можуть бути використані агентствами нерухомості, інвесторами та потенційними покупцями для підтримки прийняття рішень.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ОБЛАСТІ ПРОГНОЗУВАННЯ ВАРТОСТІ НЕРУХОМОСТІ

1.1 Аналіз предметної області прогнозування вартості нерухомості

Ринок нерухомості є складною системою, у якій ціна об'єкта формується під впливом великої кількості факторів. На вартість житлового будинку можуть впливати його фізичні характеристики, місце розташування, стан, рік побудови, площа земельної ділянки, загальна площа житла, кількість кімнат, якість будівлі та інші параметри. Окрім цього, важливим є часовий фактор, оскільки ринкова ціна нерухомості може змінюватися залежно від загальної економічної ситуації, попиту, пропозиції та періоду продажу.

У межах цієї роботи використовується набір даних про продажі будинків у King County. Він містить інформацію про фактичну ціну продажу нерухомості, дату продажу, площу будинку, площу ділянки, кількість спалень, кількість ванних кімнат, кількість поверхів, стан будинку, якість будівництва, географічні координати, поштовий індекс, рік побудови та інші ознаки [1]. Така структура даних дозволяє розглядати задачу оцінювання вартості нерухомості як задачу регресії, у якій вхідними даними є характеристики об'єкта, а вихідним значенням є прогнозована ціна.

Традиційний підхід до оцінювання нерухомості часто ґрунтується на порівнянні об'єкта з подібними об'єктами, які були продані раніше. Такий підхід може бути корисним, однак він значною мірою залежить від досвіду оцінювача та може не враховувати складні нелінійні залежності між ознаками. Наприклад, однакова площа будинку може мати різний вплив на ціну залежно від району, стану будинку або року побудови. Методи

машинного навчання дозволяють автоматично знаходити такі залежності на основі історичних даних і використовувати їх для побудови прогнозу.

Задача прогнозування вартості нерухомості належить до задач навчання з учителем, оскільки для кожного об'єкта у навчальній вибірці відома правильна відповідь — фактична ціна продажу. Модель навчається на прикладах, знаходячи залежність між набором ознак та цільовою змінною. Після навчання така модель може бути використана для прогнозування ціни нового об'єкта, для якого відомі характеристики, але невідома очікувана ринкова вартість.

Для розв'язання цієї задачі доцільно використовувати кілька моделей і порівнювати їх між собою. Лінійна регресія може використовуватися як базова модель, оскільки вона є простою для інтерпретації та дозволяє оцінити загальний характер залежності між ознаками і ціною. Водночас реальна залежність між параметрами нерухомості та її вартістю часто є нелінійною, тому у роботі також застосовуються ансамблеві методи градієнтного бустингу, зокрема XGBoost, CatBoost та LightGBM [2], [3], [4]. Ці алгоритми будують послідовність дерев рішень, де кожне наступне дерево виправляє помилки попередніх, що дозволяє отримувати високу точність на табличних даних.

Окремим напрямом аналізу є дослідження зміни цін у часі. Оскільки набір даних містить дату продажу, можна сформувати часовий ряд середньої вартості нерухомості за певні періоди та дослідити його динаміку. Для цього у роботі використовується LSTM-модель, яка належить до рекурентних нейронних мереж і застосовується для роботи з послідовностями [5]. На відміну від звичайних регресійних моделей, LSTM враховує попередні значення ряду та може використовувати їх для прогнозування наступних значень.

Таким чином, предметна область роботи поєднує два взаємопов'язані завдання: прогнозування ціни конкретного об'єкта нерухомості на основі його характеристик та аналіз часової динаміки середньої вартості. Такий підхід дозволяє не лише отримати прогноз ціни, а й оцінити якість різних моделей, визначити найважливіші фактори впливу та візуально проаналізувати результати роботи системи. Для кращого розуміння взаємозв'язків між основними етапами обробки даних та прогнозування вартості нерухомості було сформовано загальну схему роботи системи, яку наведено на рисунку 1.1.



Рисунок 1.1 – Загальна схема аналізу та прогнозування вартості нерухомості

1.2 Методи прогнозування вартості нерухомості

Для прогнозування вартості нерухомості можуть використовуватися різні методи машинного навчання, однак у межах цієї роботи основну увагу приділено моделям регресії. Це пов'язано з тим, що цільовою змінною є числове значення — ціна продажу будинку. Отже, задача полягає не у віднесенні об'єкта до певного класу, а у визначенні наближеного числового значення його ринкової вартості.

Найпростішою моделлю, яку доцільно використати як базову, є лінійна регресія. Вона описує залежність між ціною нерухомості та її характеристиками у вигляді лінійної комбінації ознак [16]. Математичний опис моделі лінійної регресії наведено у формулі 1.1:

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \dots + w_mx_m, \quad (1.1)$$

де $\hat{y}$ — прогнозована ціна об'єкта, $x_1, x_2, \dots, x_m$ — значення ознак, w_0 — вільний член, а $w_1, w_2, \dots, w_m$ — вагові коефіцієнти моделі. Перевагою лінійної регресії є простота реалізації та інтерпретації результатів [17]. Водночас така модель має обмеження, оскільки реальна залежність між характеристиками нерухомості та її ціною часто є нелінійною.

Для врахування складніших залежностей у роботі використовуються ансамблеві методи градієнтного бустингу. Суть такого підходу полягає у побудові послідовності слабких моделей, найчастіше дерев рішень, де кожна наступна модель навчається виправляти помилки попередніх. Завдяки цьому градієнтний бустинг може ефективно працювати з табличними даними,

враховувати нелінійні залежності між ознаками та виявляти складні взаємозв'язки у вибірці.

У роботі застосовуються три реалізації градієнтного бустингу: XGBoost, CatBoost та LightGBM. XGBoost є ефективною реалізацією бустингу над деревами рішень, яка часто використовується для задач регресії та класифікації на структурованих даних [2]. CatBoost також належить до методів градієнтного бустингу, але має зручні механізми роботи з категоріальними ознаками, що є корисним для задач із табличними даними [3]. LightGBM орієнтований на швидке навчання моделей і може бути ефективним при роботі з великими наборами даних [4].

Окрім моделей для прогнозування ціни окремого об'єкта, у роботі використовується LSTM-модель для аналізу часової динаміки середньої вартості нерухомості. LSTM є різновидом рекурентної нейронної мережі, яка здатна зберігати інформацію про попередні стани послідовності та використовувати її під час прогнозування наступних значень [5]. Це є важливим для часових рядів, оскільки ціна нерухомості може залежати не лише від поточного стану ринку, а й від попередньої динаміки змін.

Основна ідея LSTM полягає у використанні спеціальних механізмів пам'яті, які визначають, яку інформацію потрібно зберегти, яку забути, а яку використати для формування прогнозу. Завдяки цьому така модель може враховувати довші залежності у часовому ряді порівняно зі звичайними рекурентними мережами. У межах цієї роботи LSTM використовується не для прогнозування ціни кожного окремого будинку за всіма характеристиками, а для аналізу загального часового тренду середньої вартості нерухомості.

Таким чином, у роботі поєднуються два підходи. Перший підхід полягає у використанні регресійних моделей машинного навчання для прогнозування ціни конкретного об'єкта на основі його характеристик.

Другий підхід полягає у використанні LSTM для аналізу часової зміни середньої ціни. Це дозволяє оцінити задачу як з погляду характеристик окремого будинку, так і з погляду загальної динаміки ринку.

1.3 Аналіз ознак, що впливають на вартість нерухомості

Для побудови ефективної системи прогнозування вартості нерухомості необхідно визначити сукупність факторів, які найбільшою мірою впливають на формування ринкової ціни об'єкта. На відміну від багатьох інших економічних показників, вартість нерухомості залежить одночасно від великої кількості взаємопов'язаних характеристик, які можуть мати як прямий, так і непрямий вплив на кінцеву ціну. Саме тому сучасні системи прогнозування використовують багатофакторний підхід, що враховує фізичні параметри об'єкта, його місце розташування, часові характеристики та додаткові якісні показники [1].

У використаному наборі даних кожен об'єкт нерухомості описується набором ознак, які можна умовно поділити на декілька груп: характеристики площі та планування будинку, ознаки місця розташування, показники технічного стану, часові характеристики та додаткові параметри, що відображають привабливість об'єкта для потенційного покупця. Така структура даних дозволяє досліджувати як окремий вплив кожного фактора, так і їхню спільну дію під час формування ринкової вартості.

Одним із найважливіших факторів є площа житлового приміщення (sqft_living). Дослідження ринку нерухомості показують, що між житловою площею та вартістю об'єкта зазвичай існує позитивна залежність: зі збільшенням площі ціна будинку зростає. Проте така залежність не завжди є

лінійною. Наприклад, збільшення площі будинку зі 100 до 150 квадратних метрів може впливати на вартість значно сильніше, ніж збільшення площі з 300 до 350 квадратних метрів. Крім того, вплив площі може залежати від району розташування будинку, його класу якості та інших характеристик. Саме тому сучасні алгоритми машинного навчання здатні ефективніше враховувати подібні нелінійні взаємозв'язки порівняно з класичними статистичними моделями [13].

Важливими характеристиками є також площа земельної ділянки (`sqft_lot`), площа надземної частини будинку (`sqft_above`) та площа підвального приміщення (`sqft_basement`). Велика земельна ділянка може збільшувати привабливість об'єкта, особливо в районах із низькою щільністю забудови. Водночас значна площа підвалу не завжди прямо впливає на ціну, оскільки її корисність залежить від призначення приміщення та рівня його облаштування. Тому різні складові площі можуть по-різному впливати на результат прогнозування.

Окрему групу становлять характеристики планування будинку. До них належать кількість спалень (`bedrooms`), кількість ванних кімнат (`bathrooms`) та кількість поверхів (`floors`). Ці параметри безпосередньо впливають на комфорт проживання та функціональність житла. Для більшості покупців збільшення кількості кімнат підвищує цінність об'єкта, однак надмірна кількість кімнат за відносно невеликої площі може навпаки знижувати зручність використання будинку. Аналогічно кількість ванних кімнат часто є важливим показником для великих сімей і може істотно впливати на ринкову ціну нерухомості.

Особливу роль у формуванні вартості нерухомості відіграє місце розташування об'єкта. У наборі даних воно описується за допомогою поштового індексу (`zipcode`), географічної широти (`lat`) та довготи (`long`). Практика оцінювання нерухомості свідчить, що два будинки з однаковими фізичними характеристиками можуть суттєво відрізнятися за вартістю лише

через різне розташування. Це пояснюється впливом таких факторів, як транспортна доступність, рівень розвитку інфраструктури, близькість до навчальних закладів, торговельних центрів, парків та інших об'єктів соціальної інфраструктури. Координати дозволяють моделям машинного навчання враховувати просторові закономірності, які не можуть бути повністю описані лише поштовим індексом [14].

Важливими показниками якості нерухомості є ознаки *condition* та *grade*. Ознака *condition* характеризує загальний технічний стан будинку, тоді як *grade* відображає якість конструкції, використаних матеріалів та рівень виконання будівельних робіт. На практиці будинки з високими значеннями *grade* зазвичай мають суттєво більшу вартість навіть за однакової площі. Це пояснюється використанням якісніших матеріалів, кращими архітектурними рішеннями та вищим рівнем комфорту для мешканців.

До додаткових факторів належать ознаки *waterfront* та *view*. Наявність виходу до водойми або панорамного виду є важливою конкурентною перевагою об'єкта нерухомості. Подібні характеристики трапляються відносно рідко, тому вони можуть значно збільшувати ринкову вартість будинку. У деяких випадках вплив таких ознак може перевищувати вплив площі чи року побудови, особливо для елітної нерухомості.

Важливе значення мають характеристики, що описують вік будівлі. До них належать рік побудови (*yr_built*) та рік реконструкції (*yr_renovated*). Нові будинки зазвичай відповідають сучасним стандартам енергоефективності та комфорту, тому можуть оцінюватися дорожче. Водночас реконструкція дозволяє суттєво підвищити ринкову вартість навіть старого об'єкта. Саме тому доцільно враховувати не лише абсолютний рік побудови, а й фактичний вік будинку на момент продажу, а також час, що минув після останнього капітального ремонту або реконструкції.

Окрему категорію становлять часові характеристики. Основною часовою ознакою є дата продажу (date), яка дозволяє врахувати динаміку ринку нерухомості. Ціни на житло можуть змінюватися під впливом економічних факторів, змін попиту та пропозиції, кредитної політики банків і сезонних коливань. Наприклад, у певні періоди року активність покупців може зростати, що призводить до збільшення середньої вартості об'єктів. На основі дати продажу можуть бути сформовані додаткові ознаки, такі як рік, квартал або місяць продажу. Крім того, ця інформація використовується для побудови часового ряду середньої вартості нерухомості та подальшого аналізу за допомогою LSTM-моделі [11].

У наборі даних також присутні ознаки `sqft_living15` та `sqft_lot15`, які характеризують середні параметри найближчих будинків. Такі показники дозволяють частково враховувати особливості навколишньої забудови та рівень розвитку району. Якщо об'єкт розташований серед будинків великої площі та високої вартості, його ринкова ціна також, як правило, буде вищою. Таким чином, зазначені ознаки дозволяють враховувати вплив сусіднього середовища на оцінку нерухомості.

Аналіз наведених характеристик показує, що між окремими ознаками та цільовою змінною існують складні взаємозв'язки. Більшість факторів впливають на вартість нерухомості не ізольовано, а у поєднанні з іншими характеристиками. Наприклад, велика площа може мати різний вплив залежно від району розташування, а високий клас будівлі може по-різному оцінюватися для нових і старих будинків. Саме тому для розв'язання задачі прогнозування доцільно використовувати моделі машинного навчання, здатні враховувати нелінійні залежності та складні комбінації ознак [13].

Таким чином, використаний набір даних містить достатню кількість характеристик для побудови моделей прогнозування вартості нерухомості. Поєднання фізичних параметрів будинку, просторових характеристик, показників якості та часових факторів створює інформаційну основу для

отримання точних прогнозів та подальшого аналізу закономірностей формування цін на ринку нерухомості.

1.4 Постановка задачі дослідження

У межах роботи розглядається задача прогнозування вартості житлової нерухомості на основі історичних даних про продажі будинків. Вхідними даними є характеристики об'єкта нерухомості, зокрема площа житла, площа земельної ділянки, кількість спалень, кількість ванних кімнат, кількість поверхів, стан будинку, клас якості будівлі, рік побудови, рік реконструкції, місце розташування та дата продажу. Вихідним значенням є прогнозована ціна продажу об'єкта.

Задача прогнозування ціни нерухомості належить до задач регресії, оскільки результатом роботи моделі є числове значення. Нехай задано навчальну вибірку у формулі 1.2.

$$D = \{(X_i, y_i)\}_{i=1}^n, \quad (1.2)$$

де X_i — вектор характеристик i -го об'єкта нерухомості, y_i — фактична ціна продажу цього об'єкта, а n — кількість записів у вибірці. Необхідно побудувати таку модель $f(X)$, яка за характеристиками об'єкта формує прогноз його вартості $\hat{y}$.

Математично задачу можна подати у вигляді наближення невідомої залежності між вхідними ознаками та ціною об'єкта. Для кожного об'єкта модель обчислює прогнозоване значення, після чого результат порівнюється з фактичною ціною. Під час навчання параметри моделі підбираються таким чином, щоб мінімізувати похибку між фактичними та прогнозованими значеннями. Для оцінювання якості прогнозування використовуються метрики MAE, MSE, RMSE, MAPE, sMAPE, R^2 та інші критерії, які дозволяють кількісно оцінити точність моделі [6].

Окрім прогнозування ціни окремого об'єкта, у роботі також розглядається задача аналізу часової динаміки середньої вартості нерухомості. Для цього на основі дати продажу формується часовий ряд, у якому кожне значення відповідає середній ціні нерухомості за певний період. Для аналізу такого ряду використовується LSTM-модель, оскільки вона здатна враховувати попередні значення послідовності під час прогнозування наступних значень [5].

Метою розв'язання поставленої задачі є створення програмної системи, яка дозволяє завантажувати набір даних, виконувати попередню обробку, навчати декілька моделей прогнозування, порівнювати їхню якість, будувати графіки результатів та отримувати прогноз вартості нерухомості для нового об'єкта. Результатом роботи системи має бути не лише прогноз ціни, а й аналітична інформація про точність моделей, характер похибок та вплив окремих ознак на результат прогнозування.

Висновки до розділу 1

У першому розділі було розглянуто предметну область прогнозування вартості нерухомості та визначено основні фактори, які можуть впливати на ціну об'єкта. Встановлено, що вартість будинку залежить не лише від його фізичних характеристик, таких як площа, кількість кімнат, стан і рік побудови, а й від місця розташування та часу продажу.

Було визначено, що задача прогнозування ціни нерухомості належить до задач регресії, оскільки результатом роботи моделі є числове значення. Для її розв'язання доцільно використовувати як базові регресійні моделі, так і сучасні ансамблеві методи машинного навчання, зокрема XGBoost, CatBoost та LightGBM [2], [3], [4]. Також обґрунтовано використання LSTM-моделі для аналізу часової динаміки середньої вартості нерухомості, оскільки така архітектура призначена для роботи з послідовностями та часовими рядами [5].

У результаті аналізу було сформульовано постановку задачі дослідження. Вхідними даними є характеристики об'єктів нерухомості та дата продажу, а вихідним значенням — прогнозована ціна. Подальші розділи роботи присвячені опису структури даних, попередній обробці, реалізації моделей прогнозування, оцінюванню їхньої точності та аналізу отриманих результатів.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА ПІДГОТОВКА ДАНИХ ДЛЯ МОДЕЛЕЙ ПРОГНОЗУВАННЯ

2.1 Опис вхідного набору даних

У роботі використовується набір даних про продажі житлової нерухомості в окрузі King County. Початковий набір даних містить 21613 записів і 21 ознаку, серед яких є ідентифікатор об'єкта, дата продажу, фактична ціна, кількість спалень, кількість ванних кімнат, площа житла, площа земельної ділянки, кількість поверхів, наявність виходу до водойми, оцінка виду, стан будинку, клас якості будівлі, площа надземної частини, площа підвалу, рік побудови, рік реконструкції, поштовий індекс, географічні координати та характеристики сусідніх об'єктів [1].

Цільовою змінною у наборі даних є `price`, тобто фактична ціна продажу будинку. Саме це значення використовується як правильна відповідь під час навчання моделей. Усі інші характеристики розглядаються як вхідні ознаки, на основі яких модель повинна сформувати прогноз вартості нерухомості.

Ознаки у наборі даних мають різні типи. Частина з них є числовими неперервними величинами, наприклад `price`, `bathrooms`, `floors`, `lat` та `long`. Інша частина є числовими дискретними ознаками, зокрема `bedrooms`, `sqft_living`, `sqft_lot`, `sqft_above`, `sqft_basement`, `yr_built`, `yr_renovated`, `sqft_living15` та `sqft_lot15`. Деякі ознаки мають категоріальний або умовно категоріальний характер, наприклад `waterfront`, `view`, `condition`, `grade` та `zipcode`. Окремо виділяється ознака `date`, яка містить дату продажу і використовується для формування часових характеристик.

Для подальшої роботи з моделями дані потребують попередньої обробки. Це пов'язано з тим, що алгоритми машинного навчання працюють не з початковими текстовими або змішаними форматами, а з числовими

значеннями, підготовленими до навчання. Тому дату продажу необхідно перетворити у набір окремих часових ознак, категоріальні або умовно категоріальні змінні потрібно привести до зручного для моделі вигляду, а числові ознаки за потреби масштабувати або використати для створення нових похідних характеристик.

У програмній реалізації також використовується розширений набір даних, який містить 64839 записів. Він має таку саму структуру, як і початковий набір, але більшу кількість спостережень. Це дає змогу збільшити обсяг навчальної вибірки та отримати стабільніші результати під час навчання моделей. Основні характеристики набору даних, що використовуються для навчання моделей прогнозування, наведено в таблиці 2.1.

Таблиця 2.1 – Опис ознак вхідного набору даних

<i>Назва ознаки</i>	<i>Опис ознаки</i>	<i>Тип даних</i>	<i>Роль у моделі</i>
sqft_basement	Площа підвального приміщення	Числовий дискретний	Вхідна ознака
yr_built	Рік побудови будинку	Числовий дискретний / часова ознака	Вхідна ознака
yr_renovated	Рік реконструкції будинку	Числовий дискретний / часова ознака	Вхідна ознака
zipcode	Поштовий індекс району	Категоріальний	Вхідна ознака
lat	Географічна широта	Числовий неперервний	Вхідна ознака
long	Географічна довгота	Числовий неперервний	Вхідна ознака
sqft_living15	Середня житлова площа найближчих будинків	Числовий дискретний	Вхідна ознака
sqft_lot15	Середня площа ділянок найближчих будинків	Числовий дискретний	Вхідна ознака
grade	Клас якості будівлі та конструкції	Категоріальний порядковий	Вхідна ознака
sqft_above	Площа надземної частини будинку	Числовий дискретний	Вхідна ознака

Кінець таблиці 2.1

<i>Назва ознаки</i>	<i>Опис ознаки</i>	<i>Тип даних</i>	<i>Роль у моделі</i>
id	Унікальний ідентифікатор запису про об'єкт нерухомості	Числовий дискретний	Службова ознака, не використовується для навчання
date	Дата продажу об'єкта нерухомості	Дата / часова ознака	Використовується для формування часових ознак
price	Фактична ціна продажу будинку	Числовий неперервний	Цільова змінна
bedrooms	Кількість спалень у будинку	Числовий дискретний	Вхідна ознака
bathrooms	Кількість ванних кімнат	Числовий неперервний	Вхідна ознака
sqft_living	Загальна житлова площа будинку	Числовий дискретний	Вхідна ознака
sqft_lot	Площа земельної ділянки	Числовий дискретний	Вхідна ознака
floors	Кількість поверхів	Числовий неперервний	Вхідна ознака
waterfront	Наявність виходу до водойми	Бінарний / категоріальний	Вхідна ознака
view	Оцінка виду з будинку	Категоріальний порядковий	Вхідна ознака
condition	Загальний стан будинку	Категоріальний порядковий	Вхідна ознака

Як видно з таблиці 2.1, набір даних містить ознаки різних типів: числові, категоріальні, бінарні та часові. Це необхідно враховувати під час попередньої обробки, оскільки різні типи ознак потребують різних підходів до підготовки перед навчанням моделей. Цільовою змінною є `price`, а ознака `id` має службовий характер і не використовується як фактор прогнозування.

2.2 Попередня обробка даних

Перед навчанням моделей машинного навчання набір даних необхідно підготувати до подальшого аналізу. Попередня обробка є одним із ключових

етапів побудови систем прогнозування, оскільки якість вхідних даних безпосередньо впливає на точність отриманих результатів. Наявність пропущених значень, помилок у форматуванні або аномальних спостережень може негативно впливати на процес навчання моделей і призводити до зниження якості прогнозів [6].

У програмній системі попередня обробка даних виконується перед етапом формування ознак та навчання моделей. Вона включає перевірку типів даних, обробку часових характеристик, заповнення пропущених значень, обмеження впливу викидів та перетворення категоріальних ознак у формат, придатний для використання алгоритмами машинного навчання.

Першим етапом є перевірка структури набору даних та коректності типів змінних. Під час завантаження окремі значення можуть бути представлені у текстовому форматі або містити додаткові символи, які перешкоджають їх використанню в обчисленнях. Для усунення таких проблем числові ознаки приводяться до відповідного числового типу. Це забезпечує можливість виконання статистичного аналізу та подальшого використання даних у моделях прогнозування.

Особлива увага приділяється обробці часових даних. У наборі даних присутня ознака **date**, яка містить інформацію про дату продажу об'єкта нерухомості. Для подальшого аналізу вона перетворюється у формат дати та часу, після чого формується ознака **sale_date**. На її основі створюються додаткові часові характеристики, зокрема рік, місяць, квартал і день продажу.

Виділення окремих часових компонентів дозволяє враховувати сезонні особливості ринку нерухомості. Активність покупців і продавців може змінюватися залежно від періоду року, що впливає на рівень попиту та середню вартість житла. Крім того, часові характеристики використовуються для побудови часового ряду, який надалі застосовується під час навчання LSTM-моделі.

Під час підготовки даних також враховуються можливі помилки у записі дат. Якщо значення не може бути коректно інтерпретоване як дата, система використовує доступну інформацію для відновлення часових параметрів або формує резервне значення. Це дозволяє мінімізувати втрати даних та забезпечити стабільність роботи програмної системи.

Наступним етапом є обробка пропущених значень. Пропуски є поширеною проблемою під час роботи з реальними наборами даних і можуть ускладнювати роботу алгоритмів машинного навчання. Замість видалення записів використовується підхід заповнення пропущених значень, що дозволяє зберегти максимальну кількість спостережень та уникнути втрати потенційно корисної інформації [6].

Для числових ознак пропущені значення замінюються медіаною відповідної колонки. Такий підхід є більш стійким до аномальних значень порівняно із використанням середнього арифметичного та дозволяє краще зберегти структуру даних [6].

Для категоріальних ознак використовується заповнення найбільш поширеним значенням. Якщо визначити таке значення неможливо, застосовується службова категорія **unknown**, яка позначає відсутність достовірної інформації щодо відповідної характеристики [6].

Після усунення пропусків виконується обробка викидів. Аномальні значення можуть виникати внаслідок помилок введення даних або бути пов'язаними з нетиповими об'єктами нерухомості. Їх наявність здатна впливати на результати навчання моделей та спотворювати статистичні характеристики вибірки.

Для зменшення впливу таких значень використовується підхід, заснований на аналізі розподілу даних та визначенні допустимих меж для кожної числової ознаки [6]. Значення, які виходять за встановлені межі, не видаляються з набору даних, а обмежуються граничними значеннями. Це

дозволяє зберегти всі записи та водночас зменшити вплив екстремальних спостережень на результати прогнозування.

Після завершення обробки числових даних виконується перетворення категоріальних ознак у числовий формат. Оскільки більшість алгоритмів машинного навчання працюють лише з числовими даними, текстові значення повинні бути відповідним чином закодовані. Для цього використовується метод **One-Hot Encoding**, який передбачає створення окремої бінарної ознаки для кожного можливого значення категоріальної змінної [6].

У результаті кожна категорія представляється окремою колонкою, значення якої вказує на наявність або відсутність відповідної характеристики в конкретному записі. Такий підхід дозволяє уникнути появи штучних числових залежностей між категоріями та забезпечує коректну роботу моделей прогнозування.

Після виконання всіх етапів попередньої обробки формується фінальний набір даних, придатний для подальшого аналізу та навчання моделей. Підготовка та трансформація даних реалізовані засобами бібліотеки **pandas** [7]. Отримані ознаки використовуються під час навчання моделей Linear Regression, XGBoost, CatBoost і LightGBM, а також для формування часового ряду, який застосовується в LSTM-моделі. Таким чином, попередня обробка забезпечує підвищення якості вхідних даних і створює основу для отримання надійних результатів прогнозування вартості нерухомості.

2.3 Формування додаткових ознак

Після завершення попередньої обробки даних виконується етап формування додаткових ознак. У задачах машинного навчання якість ознак

часто має не менше значення, ніж вибір самого алгоритму прогнозування. Навіть за наявності великого набору початкових характеристик не всі важливі закономірності можуть бути безпосередньо представлені у вихідних даних. Тому додаткові ознаки формуються на основі вже наявної інформації та дозволяють більш повно описати властивості об'єктів нерухомості.

Основною метою формування нових характеристик є підвищення інформативності набору даних та покращення здатності моделей виявляти приховані залежності між параметрами об'єкта та його ринковою вартістю. Такий підхід широко використовується під час побудови прогнозних моделей і є важливим елементом процесу підготовки даних [6].

Однією з найважливіших груп додаткових характеристик є часові ознаки. На основі дати продажу формуються окремі змінні, що відображають рік, місяць, квартал та день здійснення угоди. Виділення цих компонентів дозволяє моделям враховувати часові зміни на ринку нерухомості та можливі сезонні коливання цін. Наприклад, у різні періоди року активність покупців і продавців може відрізнятися, що впливає на рівень попиту та середню вартість житла.

Крім використання часових компонентів як окремих ознак, інформація про дату продажу застосовується для побудови часового ряду середньої вартості нерухомості. На основі цього ряду здійснюється навчання LSTM-моделі, яка використовується для аналізу динаміки цін та прогнозування їх подальшої зміни [5], [11].

Важливою групою додаткових характеристик є ознаки, що описують вік та стан будинку. На основі року побудови визначається вік об'єкта на момент продажу. Ця характеристика дозволяє оцінити ступінь фізичного зносу будівлі та її відповідність сучасним будівельним стандартам. Як правило, новіші будинки мають вищу ринкову вартість, оскільки потребують менших витрат на ремонт та обслуговування.

Якщо об'єкт нерухомості проходив реконструкцію або капітальне оновлення, додатково враховується рік реконструкції. На основі цієї інформації може бути визначений період, що минув після останнього оновлення будинку. Такі характеристики дозволяють точніше оцінити фактичний технічний стан об'єкта, ніж використання лише року його первинного будівництва.

Окрему групу становлять ознаки, пов'язані з площею нерухомості. Хоча житлова площа є однією з найважливіших характеристик будинку, сама по собі вона не завжди повністю відображає особливості об'єкта. Тому додатково формуються похідні показники, які описують співвідношення між різними типами площ.

Зокрема, може враховуватися співвідношення житлової площі до площі земельної ділянки. Дана характеристика дозволяє оцінити ефективність використання земельної ділянки та просторові особливості забудови. Також можуть використовуватися показники частки підвального приміщення у загальній площі будинку та співвідношення житлової площі до кількості кімнат. Подібні характеристики надають моделям додаткову інформацію про структуру житла та рівень комфорту об'єкта.

Важливими для прогнозування є ознаки, які характеризують якість будинку та його привабливість для потенційних покупців. До таких характеристик належать клас якості будівництва, загальний стан будинку, наявність панорамного виду, а також наявність виходу до водойми. Ці фактори можуть істотно впливати на кінцеву ринкову вартість навіть у випадках, коли фізичні характеристики двох об'єктів є подібними.

Наприклад, два будинки можуть мати однакову площу, кількість кімнат та рік побудови, проте об'єкт із кращим видом або вищим класом якості будівництва часто матиме значно більшу вартість. Саме тому врахування якісних характеристик є важливою складовою процесу формування ознак.

Додатково враховуються просторові характеристики об'єктів нерухомості. У наборі даних для цього використовуються географічні координати та поштові індекси. Такі ознаки дозволяють моделям враховувати вплив місця розташування на вартість житла. Розташування є одним із ключових факторів ціноутворення на ринку нерухомості, оскільки визначає доступність транспортної інфраструктури, близькість до ділових центрів, навчальних закладів, торговельних об'єктів та інших елементів міського середовища.

Після формування додаткових характеристик виконується перевірка їхньої коректності та узгодженості з основним набором даних. Створені ознаки додаються до фінальної вибірки та використовуються під час навчання моделей прогнозування. Використання розширеного набору характеристик дозволяє алгоритмам ефективніше виявляти закономірності у даних та підвищувати точність прогнозів.

Особливо важливим формування додаткових ознак є для ансамблевих моделей градієнтного бустингу, таких як XGBoost, CatBoost та LightGBM. Дані алгоритми здатні автоматично виявляти складні взаємозв'язки між ознаками та використовувати їх для побудови більш точних прогнозів [2], [3], [4]. Завдяки цьому додаткові характеристики підвищують інформативність набору даних і сприяють покращенню якості роботи моделей.

Таким чином, формування додаткових ознак є важливим етапом підготовки даних, який дозволяє розширити початковий опис об'єктів нерухомості та надати моделям більше корисної інформації для прогнозування. Отриманий набір характеристик використовується на наступних етапах побудови та оцінювання моделей машинного навчання.

2.4 Математичне забезпечення системи прогнозування

Для прогнозування ринкової вартості нерухомості в роботі використовуються методи регресійного аналізу, ансамблеві алгоритми градієнтного бустингу та нейронні мережі для аналізу часових рядів. Математичне забезпечення системи охоплює моделі Linear Regression, XGBoost, CatBoost, LightGBM, LSTM, а також методи статистичного аналізу даних, зокрема дисперсійний аналіз ANOVA.

Базовою моделлю прогнозування виступає лінійна регресія, яка описує залежність між вартістю нерухомості та її характеристиками у вигляді лінійної комбінації ознак. Прогнозоване значення визначається за формулою 2.1:

$$\hat{y} = w_0 + \sum_{i=1}^m w_i x_i, \quad (2.1)$$

де $\hat{y}$ — прогнозована ціна об'єкта нерухомості; $\sum_{i=1}^m w_i x_i$ — значення ознак та

вагових коефіцієнтів моделі; w_0 — вільний член. Навчання моделі виконується шляхом мінімізації функції втрат, яка для задач регресії найчастіше задається через середньоквадратичну похибку у формулі 2.2[6]:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2, \quad (2.2)$$

де N — кількість спостережень у вибірці; y_i — фактичне значення ціни; $\hat{y}_i$ — прогнозоване значення.

Для підвищення точності прогнозування в системі використовуються ансамблеві алгоритми градієнтного бустингу. Одним із найбільш ефективних методів є XGBoost (Extreme Gradient Boosting), який будує композицію дерев рішень таким чином, щоб кожне наступне дерево мінімізувало помилки попередніх моделей [2]. Загальний прогноз ансамблю визначається як сума результатів усіх дерев та показаний у формулі 2.3:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad (2.3)$$

де $f_k(x_i)$ — прогноз k -го дерева рішень для об'єкта x_i .

Оптимізація моделі виконується шляхом мінімізації цільової функції та подана у формулі 2.4:

$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k), \quad (2.4)$$

де $l(y_i, \hat{y}_i)$ — функція втрат; $\Omega(f_k)$ — регуляризаційний компонент, що контролює складність моделі та зменшує ризик перенавчання. Регуляризаційний член визначається формулою 2.5:

$$\Omega(f) = \gamma T + \frac{\lambda}{2} \sum_{j=1}^T w_j^2, \quad (2.5)$$

де T — кількість листків дерева; w_j — ваги листків; γ та λ — параметри регуляризації [2].

Ще одним алгоритмом градієнтного бустингу, використаним у роботі, є LightGBM. Даний підхід реалізує ефективне навчання дерев рішень за рахунок гістограмного подання ознак та стратегії Leaf-wise Growth, яка дозволяє швидше знаходити оптимальні розбиття вузлів дерева [4]. Для оцінювання якості потенційного розбиття використовується критерій приросту інформації у формулі 2.6:

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma, \quad (2.6)$$

де G^L та G^R — суми градієнтів для лівої та правої підвибірок; H^L та H^R — відповідні суми других похідних функції втрат; λ та γ — параметри регуляризації [4].

Для роботи з категоріальними ознаками використовується алгоритм CatBoost, який автоматично виконує їх статистичне кодування та дозволяє уникнути втрати інформації під час попередньої обробки даних [3]. Основою такого перетворення є обчислення статистики CTR за формулою 2.7:

$$CTR = \frac{CountInClass + \alpha P}{TotalCount + \alpha}, \quad (2.7)$$

де CountInClass — кількість об'єктів відповідної категорії; TotalCount — загальна кількість об'єктів; P — апіорна ймовірність; α — коефіцієнт згладжування [3].

Для аналізу значущості окремих ознак та оцінювання впливу різних факторів на цільову змінну використовується дисперсійний аналіз ANOVA (Analysis of Variance). Основою методу у формулі 2.8 є F-критерій Фішера:

$$F = \frac{MS_{between}}{MS_{within}}, \quad (2.8)$$

де MSbetween — середній квадрат міжгрупової дисперсії; MSwithin — середній квадрат внутрішньогрупової дисперсії.

Величини MSbetween та MSwithin показані на формулах 2.9 та 2.10 відповідно:

$$MS_{between} = \frac{SS_{between}}{k - 1}, \quad (2.9)$$

$$MS_{within} = \frac{SS_{within}}{N - k}, \quad (2.10)$$

де SSbetween — міжгрупова сума квадратів; SSwithin — внутрішньогрупова сума квадратів; k — кількість груп; N — кількість спостережень [15]. Чим більше значення F-статистики, тим сильнішим вважається вплив відповідної ознаки на результат прогнозування.

Для аналізу часової динаміки зміни цін на нерухомість використовується нейронна мережа типу LSTM (Long Short-Term Memory), яка належить до класу рекурентних нейронних мереж і призначена для роботи з послідовностями та часовими рядами [5]. Робота LSTM ґрунтується на використанні механізмів пам'яті та спеціальних керувальних воріт.

Вхідні ворота визначаються співвідношенням у формулі 2.11:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad (2.11)$$

ворота забування, формула 2.12:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \quad (2.12)$$

оновлення стану пам'яті, формула 2.13:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t, \quad (2.13)$$

вихідні ворота, формула 2.14:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad (2.14)$$

прихований стан, формула 2.15:

$$h_t = o_t \tanh(C_t), \quad (2.15)$$

де x_t — вхідне значення у момент часу t ; h_t — прихований стан мережі; C_t — комірка пам'яті; W та b — параметри моделі; σ — сигмоїдна функція активації [5].

Таким чином, використане математичне забезпечення поєднує класичні статистичні методи аналізу даних, алгоритми градієнтного бустингу та нейронні мережі для часових рядів. Такий підхід дозволяє враховувати як лінійні, так і складні нелінійні залежності між характеристиками об'єктів нерухомості та їх ринковою вартістю, що забезпечує високу точність прогнозування.

Під час побудови моделей градієнтного бустингу важливу роль відіграють механізми регуляризації, які дозволяють зменшити ризик перенавчання та підвищити здатність моделі до узагальнення на нових даних. Перенавчання виникає у випадках, коли модель надто точно відтворює закономірності навчальної вибірки, включаючи випадковий шум, що призводить до погіршення якості прогнозування на тестових даних [2–4].

У моделі XGBoost використовується один із найбільш розвинених механізмів регуляризації серед алгоритмів градієнтного бустингу. Функція втрат доповнюється регуляризаційним доданком, який штрафує надмірну складність дерева рішень. Для цього застосовуються L1- та L2-регуляризація ваг листків дерева, а також параметри `gamma`, `max_depth`, `min_child_weight` і `subsample`, що обмежують складність моделі та знижують ймовірність перенавчання [2].

Алгоритм CatBoost також використовує низку механізмів регуляризації. Однією з його особливостей є застосування впорядкованого бустингу (Ordered Boosting), який зменшує зміщення під час навчання та запобігає витоку інформації між спостереженнями. Додатково використовуються параметри `depth`, `l2_leaf_reg` та `random_strength`, які контролюють складність дерев і забезпечують стабільність прогнозування [3].

У моделі LightGBM регуляризація реалізується за рахунок обмеження структури дерев та використання параметрів `lambda_l1` і `lambda_l2`, які відповідають за L1- та L2-регуляризацію. Крім того, застосовуються параметри `num_leaves`, `max_depth` і `min_data_in_leaf`, що дозволяють контролювати складність моделі та зменшувати вплив шумових даних [4].

Для оцінювання статистичної значущості ознак у роботі також використовується дисперсійний аналіз ANOVA (Analysis of Variance). Даний метод дозволяє визначити, наскільки сильно окрема ознака впливає на цільову змінну шляхом порівняння міжгрупової та внутрішньогрупової дисперсії. Якщо середні значення цільової змінної для різних груп суттєво відрізняються, така ознака вважається статистично значущою.

Основним результатом ANOVA є F-статистика, яка визначається як відношення міжгрупової дисперсії до внутрішньогрупової дисперсії. Високі значення F-статистики свідчать про суттєвий вплив відповідної ознаки на результат прогнозування. На основі отриманих значень формується ранжування ознак, що дозволяє оцінити їхню відносну важливість та підтвердити результати аналізу важливості ознак, отримані ансамблевими моделями градієнтного бустингу [6].

2.5 Критерії оцінювання якості моделей

Для порівняння моделей прогнозування недостатньо оцінювати результат лише візуально. Необхідно використовувати кількісні критерії, які показують, наскільки прогнозовані значення відрізняються від фактичних. У роботі для цього використовується набір метрик регресії, що дозволяє оцінити абсолютну похибку, середньоквадратичну похибку, відносну похибку, пояснену дисперсію та правильність напрямку зміни прогнозу.

Основними метриками є MAE, MSE, RMSE та R^2 . MAE показує середню абсолютну різницю між фактичною і прогнозованою ціною. Ця метрика зручна для інтерпретації, оскільки вимірюється в тих самих одиницях, що й ціна нерухомості. MSE обчислює середнє значення квадратів похибок і сильніше штрафує великі відхилення. RMSE є коренем із MSE, тому також вимірюється в одиницях ціни та показує типовий розмір похибки прогнозу. Коефіцієнт R^2 показує, яку частку варіації фактичних цін пояснює модель [6].

Окрім базових критеріїв, у програмі використовуються відносні метрики MAPE, sMAPE та MSPE. Вони дозволяють оцінити похибку у відсотках, що зручно для порівняння якості прогнозів на об'єктах з різною вартістю. Для часової динаміки також використовується MDA, яка показує, наскільки правильно модель визначає напрям зміни значення, тобто зростання або зниження ціни. Додатково застосовуються RMS, RMSD, MAD та MASE. Використання кількох метрик дає змогу отримати повнішу оцінку якості моделей, оскільки одна метрика не завжди відображає всі особливості похибки прогнозування. Для оцінювання якості моделей прогнозування використовується набір метрик, характеристики яких наведено в таблиці 2.2.

Таблиця 2.2 – Критерії оцінювання якості моделей прогнозування

<i>Критерій</i>	<i>Повна назва</i>	<i>Що показує</i>	<i>Бажане значення</i>
MAE	Mean Absolute Error	Середню абсолютну різницю між фактичною та прогнозованою ціною	Менше — краще
MSE	Mean Squared Error	Середнє значення квадратів похибок; сильніше штрафує великі помилки	Менше — краще
RMSE	Root Mean Squared Error	Типовий розмір похибки прогнозу в одиницях ціни	Менше — краще
R ²	Coefficient of Determination	Частку варіації фактичних значень, яку пояснює модель	Ближче до 1 — краще
MAPE	Mean Absolute Percentage Error	Середню абсолютну відсоткову похибку прогнозу	Менше — краще
sMAPE	Symmetric Mean Absolute Percentage Error	Симетричну відсоткову похибку, яка враховує і фактичне, і прогнозоване значення	Менше — краще
MSPE	Mean Squared Percentage Error	Середню квадратичну відсоткову похибку	Менше — краще
RMS	Root Mean Square	Середньоквадратичний рівень прогнозованих значень	Використовується для додаткового аналізу
RMSD	Root Mean Square Deviation	Середньоквадратичне відхилення між фактичними та прогнозованими значеннями	Менше — краще
MAD	Mean Absolute Deviation	Середнє абсолютне відхилення похибок від їхнього середнього значення	Менше — краще
MDA	Mean Directional Accuracy	Частку випадків, у яких модель правильно визначила напрям зміни значення	Більше — краще
MASE	Mean Absolute Scaled Error	Масштабовану абсолютну похибку відносно наївного прогнозу	Менше — краще

Як видно з таблиці 2.2, кожна метрика оцінює якість прогнозування з іншого боку. Для вибору найкращої моделі у роботі основна увага приділяється RMSE, MAE, MAPE та R². RMSE і MAE дозволяють оцінити похибку в одиницях ціни, MAPE показує середню відносну похибку у відсотках, а R² характеризує загальну здатність моделі пояснювати зміну вартості нерухомості. Інші критерії використовуються для додаткового аналізу результатів і порівняння поведінки моделей.

Математичний опис основних метрик оцінювання якості моделей наведено нижче.

Середня абсолютна похибка (MAE) визначається за формулою 2.16:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (2.16)$$

де y_i — фактичне значення ціни, $\hat{y}_i$ — прогнозоване значення, n — кількість спостережень.

Корінь із середньоквадратичної похибки (RMSE) обчислюється за формулою 2.17:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}, \quad (2.17)$$

Дана метрика характеризує типовий розмір похибки прогнозу та є чутливою до великих відхилень між прогнозованими і фактичними значеннями.

Коефіцієнт детермінації R^2 визначається за допомогою формули 2.18:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad (2.18)$$

де $\bar{y}$ — середнє значення цільової змінної у вибірці. Значення R^2 наближається до 1 для моделей, які добре пояснюють зміну цільової змінної [6].

Для оцінювання відносної похибки прогнозування використовуються метрики MAPE, sMAPE та MSPE.

Середня абсолютна відсоткова похибка (MAPE) визначається за формулою 2.19:

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad (2.19)$$

Дана метрика показує середню похибку прогнозування у відсотках від фактичного значення.

Симетрична середня абсолютна відсоткова похибка (sMAPE) обчислюється за формулою 2.20:

$$sMAPE = \frac{100}{n} \sum_{i=1}^n \frac{2|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}, \quad (2.20)$$

Перевагою цієї метрики є симетричне врахування фактичних та прогнозованих значень, що зменшує вплив дуже малих значень цільової змінної.

Середня квадратична відсоткова похибка (MSPE) визначається як формула 2.21:

$$MSPE = \frac{100}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{y_i} \right)^2, \quad (2.21)$$

На відміну від MAPE, дана метрика сильніше штрафує великі відносні помилки прогнозування, оскільки використовує квадрат відсоткової похибки [6].

2.6 Поділ даних та підготовка вибірок для навчання моделей

Після попередньої обробки та формування додаткових ознак набір даних поділяється на вхідні ознаки та цільову змінну. Цільовою змінною є *price*, тобто фактична ціна продажу об'єкта нерухомості. Усі інші підготовлені ознаки використовуються як вхідні параметри моделей, за винятком службових колонок, які не несуть корисної інформації для прогнозування, наприклад ідентифікатора запису.

Перед навчанням моделей дані поділяються на навчальну та тестову вибірки. Навчальна вибірка використовується для підбору параметрів моделі, а тестова — для перевірки якості прогнозування на даних, які модель не бачила під час навчання. Такий підхід дозволяє оцінити не лише здатність моделі запам'ятовувати навчальні приклади, а й її здатність узагальнювати закономірності для нових об'єктів.

Формально поділ можна описати так: початковий набір даних D розділяється на дві частини — D_{train} та D_{test} . На вибірці D_{train} модель навчається встановлювати залежність між характеристиками об'єкта і його ціною, а на вибірці D_{test} обчислюються метрики якості прогнозування. Це дозволяє порівняти моделі між собою за однакових умов.

У програмній реалізації для кожної моделі використовується однакова логіка підготовки даних, щоб результати порівняння були коректними. Після поділу вибірки моделі навчаються на однаковому наборі ознак, після чого для тестової вибірки формуються прогнозовані значення. Далі ці значення порівнюються з фактичними цінами, а якість прогнозування оцінюється за критеріями, наведеними у таблиці 2.2.

Для моделей машинного навчання, які працюють із табличними даними, важливо зберегти однакову структуру ознак у навчальній і тестовій вибірках. Це особливо актуально після кодування категоріальних змінних, коли одна початкова ознака може перетворюватися на декілька бінарних колонок. Якщо структура колонок буде відрізнятися, модель не зможе коректно виконати прогноз для нових даних.

Для LSTM-моделі підготовка даних виконується окремо, оскільки вона працює не з незалежними записами про окремі будинки, а з послідовністю значень у часі. Для цього на основі дати продажу формується часовий ряд середньої ціни нерухомості. Далі цей ряд перетворюється у набір послідовностей фіксованої довжини, де попередні значення використовуються для прогнозування наступного значення. Такий формат

відповідає принципу роботи LSTM-моделей, які використовуються для аналізу часових рядів [5].

Отже, підготовка вибірок є важливим етапом роботи системи. Вона забезпечує коректне навчання моделей, об'єктивне порівняння їхньої якості та можливість перевірити, наскільки добре модель працює з новими даними.

Висновки до розділу 2

У другому розділі було описано вхідний набір даних, який використовується для прогнозування вартості нерухомості. Визначено основні ознаки, їхній зміст, тип даних та роль у моделі. Цільовою змінною є ціна продажу об'єкта нерухомості, а всі інші підготовлені характеристики використовуються як вхідні параметри моделей.

Було розглянуто основні етапи попередньої обробки даних: очищення числових значень, обробку дати продажу, заповнення пропущених значень, обмеження викидів та перетворення категоріальних ознак у числовий формат. Також описано формування додаткових ознак, які дозволяють краще врахувати вік будинку, часові характеристики продажу, співвідношення площ та інші важливі властивості об'єкта.

Окремо було визначено критерії оцінювання якості моделей. Для аналізу результатів використовуються метрики MAE, MSE, RMSE, R^2 , MAPE, sMAPE, MSPE, RMS, RMSD, MAD, MDA та MASE. Використання кількох критеріїв дозволяє оцінити похибку прогнозування з різних сторін і об'єктивніше порівняти моделі між собою.

Після підготовки дані поділяються на навчальну та тестову вибірки. Навчальна вибірка використовується для побудови моделей, а тестова — для перевірки їхньої точності на нових даних.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ

3.1 Вибір засобів програмної реалізації

Для реалізації програмної системи було використано мову програмування Python. Вона є зручною для задач аналізу даних і машинного навчання, оскільки має велику кількість бібліотек для обробки даних, побудови моделей та візуалізації результатів.

Для роботи з табличними даними використано бібліотеки pandas та NumPy. Вони застосовуються для завантаження набору даних, очищення значень, обробки пропусків, формування нових ознак і виконання числових обчислень [7], [8].

Для побудови моделей машинного навчання використано бібліотеки Scikit-learn, XGBoost, CatBoost та LightGBM. Scikit-learn застосовується для базових моделей, поділу даних, попередньої обробки та розрахунку метрик якості [6]. XGBoost, CatBoost і LightGBM використовуються для реалізації ансамблевих моделей градієнтного бустингу [2], [3], [4].

Для реалізації LSTM-моделі використано TensorFlow/Keras. Цей інструмент дозволяє будувати нейронні мережі для обробки послідовностей і часових рядів [5]. Для побудови графіків результатів застосовано бібліотеки Matplotlib і Plotly, які дають змогу візуалізувати фактичні значення, прогнози, похибки та важливість ознак [9], [10].

Обрані засоби дозволяють реалізувати повний цикл роботи системи: завантаження даних, попередню обробку, навчання моделей, оцінювання точності, побудову графіків і формування прогнозу вартості нерухомості.

3.2 Архітектура програмної системи

Програмна система має модульну структуру. Це дозволяє розділити етапи роботи з даними, навчання моделей, оцінювання результатів і побудови графіків. Такий підхід спрощує підтримку програми та дає змогу окремо змінювати або доповнювати потрібні частини системи.

Основними етапами роботи системи є завантаження набору даних, попередня обробка, формування додаткових ознак, поділ даних на навчальну і тестову вибірки, навчання моделей, обчислення метрик якості та візуалізація результатів. Після навчання моделей система порівнює їх між собою і дозволяє визначити модель з найкращими показниками.

Для наочного представлення взаємодії основних компонентів програмної системи було розроблено її архітектурну схему, наведену на рисунку 3.1.

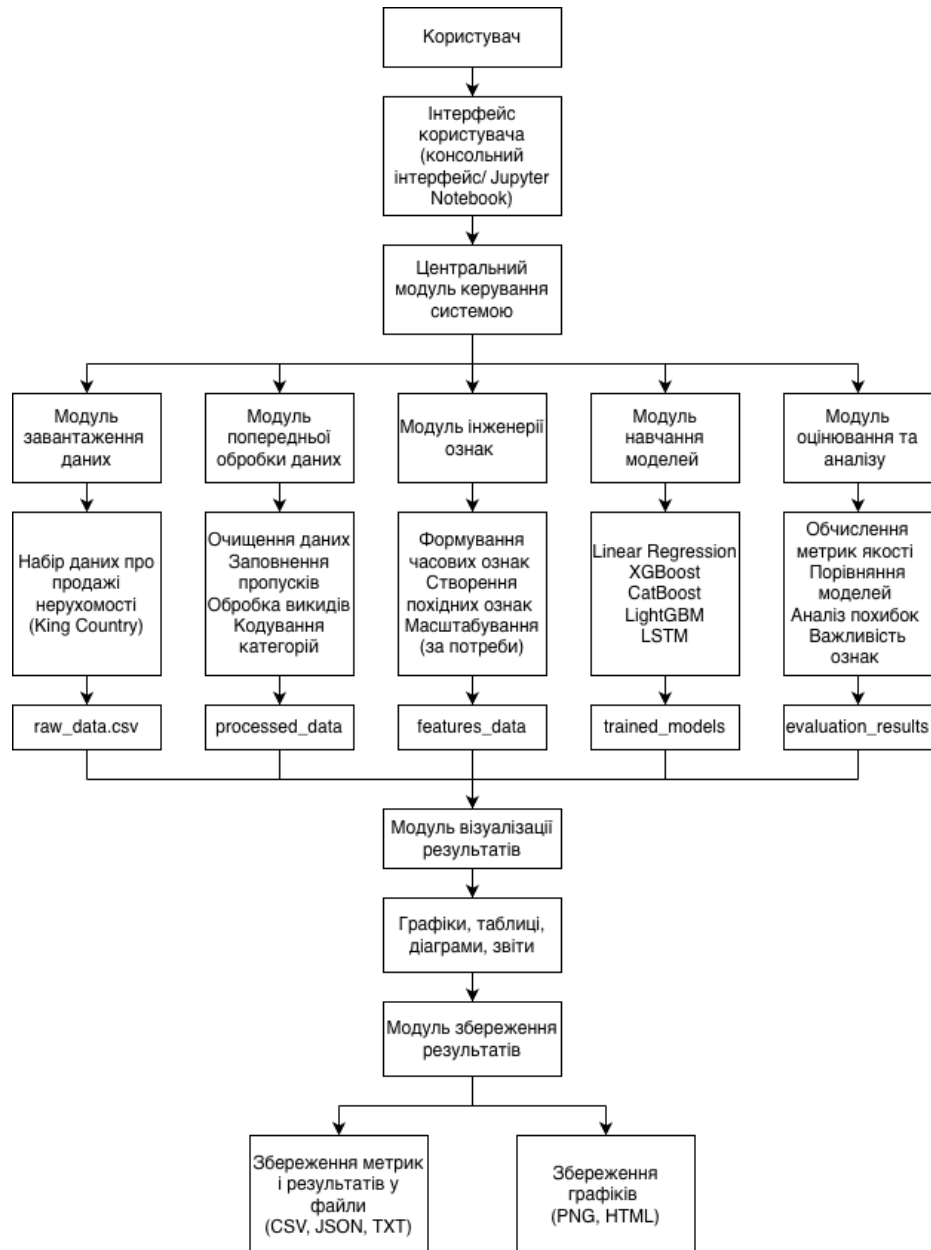


Рисунок 3.1 – Архітектура програмної системи прогнозування вартості нерухомості

Як показано на рисунку 3.1, система складається з модулів обробки даних, підготовки ознак, навчання моделей машинного навчання, аналізу результатів та збереження отриманих даних. Взаємодія між окремими компонентами забезпечує повний цикл прогнозування вартості нерухомості від завантаження даних до отримання підсумкових результатів.

У програмі реалізовано кілька моделей прогнозування: Linear Regression, XGBoost, CatBoost та LightGBM. Вони використовуються для прогнозування ціни конкретного об'єкта нерухомості на основі його характеристик. Окремо реалізовано LSTM-модель, яка застосовується для аналізу часової динаміки середньої вартості нерухомості.

Результати роботи системи подаються у вигляді таблиць метрик і графіків. Графіки дозволяють порівняти фактичні та прогнозовані значення, проаналізувати похибки, оцінити важливість ознак і дослідити часову зміну середньої ціни. Це робить результати роботи програми більш зрозумілими для користувача.

Послідовність виконання основних етапів роботи програмної системи наведено на рисунку 3.2, де видно, що робота системи починається із завантаження та підготовки даних. Після цього виконуються формування ознак, навчання моделей прогнозування та оцінювання їхньої якості. На завершальному етапі результати аналізуються, візуалізуються та надаються користувачеві у вигляді прогнозів і графічних звітів.

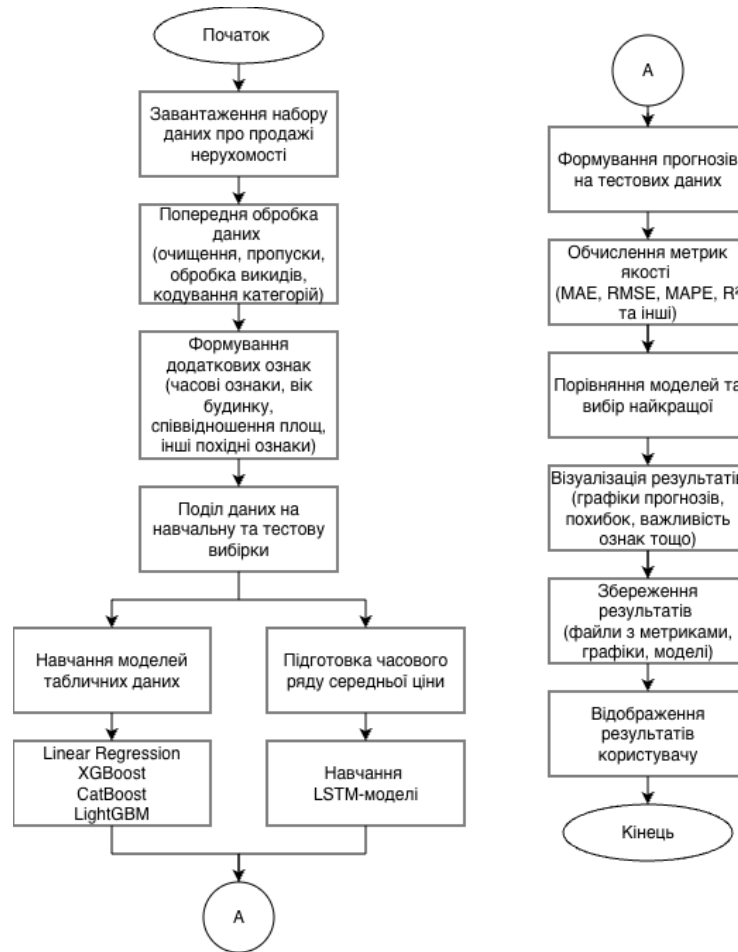


Рисунок 3.2 – Функціональна схема роботи програмної системи прогнозування вартості нерухомості

3.3 Реалізація моделей прогнозування

Для прогнозування вартості нерухомості у програмній системі реалізовано декілька моделей машинного навчання. Використання кількох алгоритмів дозволяє виконати порівняльний аналіз їхньої ефективності та визначити підхід, який забезпечує найвищу точність прогнозування на досліджуваному наборі даних. У роботі використано модель лінійної регресії

(Linear Regression), а також ансамблеві алгоритми градієнтного бустингу XGBoost, CatBoost та LightGBM.

Лінійна регресія використовується як базова модель для порівняння результатів. Вона встановлює лінійну залежність між характеристиками об'єкта нерухомості та його вартістю. Основною перевагою даного підходу є простота реалізації та висока швидкість навчання. Крім того, модель дозволяє оцінити вплив окремих ознак на результат прогнозування за допомогою вагових коефіцієнтів. Проте у задачах оцінювання нерухомості взаємозв'язки між характеристиками об'єкта та його ціною часто мають нелінійний характер, що обмежує можливості використання лінійної регресії як основного інструмента прогнозування.

Для підвищення точності прогнозів у системі реалізовано алгоритм XGBoost. Дана модель є однією з найбільш популярних реалізацій градієнтного бустингу над деревами рішень і широко застосовується для розв'язання задач регресії та класифікації [2]. Принцип її роботи полягає у послідовному створенні дерев рішень, кожне з яких навчається виправляти помилки попередніх моделей. У результаті формується ансамбль дерев, прогноз якого є комбінацією результатів усіх побудованих моделей. Завдяки використанню регуляризації XGBoost демонструє високу стійкість до перенавчання та здатний ефективно працювати з великою кількістю ознак.

Ще однією реалізованою моделлю є CatBoost, розроблена спеціально для роботи зі структурованими даними та категоріальними ознаками [3]. Особливістю алгоритму є використання спеціальних методів обробки категоріальних змінних, які дозволяють уникнути значної втрати інформації під час їх кодування. Крім того, CatBoost застосовує механізм впорядкованого бустингу (Ordered Boosting), що зменшує ризик перенавчання та покращує узагальнювальну здатність моделі. Завдяки цим особливостям алгоритм часто демонструє високу якість прогнозування навіть без складного налаштування параметрів.

Для порівняння також використовується алгоритм LightGBM, розроблений компанією Microsoft [4]. На відміну від традиційних реалізацій градієнтного бустингу, дана модель використовує листоорієнтований підхід до побудови дерев рішень. Це дозволяє значно скоротити час навчання та зменшити використання оперативної пам'яті. Завдяки високій швидкодії LightGBM є ефективним рішенням для роботи з великими наборами даних і часто використовується в практичних задачах аналізу даних.

Перед навчанням моделей виконується підготовка вхідних даних. Із набору даних виділяється цільова змінна **price**, яка містить фактичну вартість об'єктів нерухомості. Решта характеристик використовуються як вхідні ознаки для побудови прогностичних моделей. Після формування набору ознак дані поділяються на навчальну та тестову вибірки. Навчальна вибірка використовується для налаштування параметрів моделей, тоді як тестова застосовується для оцінювання їхньої здатності узагальнювати отримані знання та формувати прогнози для нових даних.

Процес навчання для всіх моделей виконується за однаковою схемою. Спочатку модель отримує навчальні дані та формує внутрішнє представлення залежностей між ознаками і цільовою змінною. Після завершення навчання модель використовується для прогнозування вартості об'єктів із тестової вибірки. Отримані результати порівнюються з фактичними значеннями цін, що дозволяє оцінити якість роботи алгоритму.

Для кількісної оцінки точності прогнозування використовуються метрики MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), MAPE (Mean Absolute Percentage Error) та коефіцієнт детермінації R^2 [6]. Використання декількох метрик дозволяє комплексно оцінити ефективність моделей, оскільки кожна з них характеризує різні аспекти якості прогнозу. MAE показує середню абсолютну похибку прогнозування, RMSE сильніше враховує великі помилки, MAPE відображає відносну похибку у відсотках, а

коефіцієнт R^2 характеризує ступінь пояснення варіації цільової змінної моделлю.

Таким чином, реалізована система поєднує як базову статистичну модель, так і сучасні ансамблеві алгоритми машинного навчання. Це дозволяє провести об'єктивне порівняння різних підходів до прогнозування вартості нерухомості та визначити модель, яка забезпечує найкращу якість прогнозів для досліджуваного набору даних.

3.4 Реалізація LSTM-моделі

Окрім моделей машинного навчання, призначених для прогнозування вартості окремих об'єктів нерухомості, у програмній системі реалізовано модель на основі архітектури Long Short-Term Memory (LSTM). Дана модель використовується для аналізу часової динаміки середньої вартості нерухомості та прогнозування подальшої зміни цін на основі історичних даних. На відміну від моделей XGBoost, CatBoost та LightGBM, які працюють із набором незалежних ознак об'єкта, LSTM орієнтована на аналіз послідовностей і враховує порядок спостережень у часі [11].

Для побудови часового ряду використовується інформація про дату продажу об'єктів нерухомості та їхню вартість. Початковий набір даних групується за датою продажу, після чого для кожного періоду обчислюється середня ціна нерухомості. У результаті формується часовий ряд, який відображає зміну середньої вартості житла протягом досліджуваного періоду. Такий підхід дозволяє зменшити вплив випадкових коливань окремих цін та зосередитися на загальній ринковій тенденції.

Перед навчанням нейронної мережі виконується попередня обробка даних. Одним із важливих етапів є масштабування значень часового ряду.

Оскільки нейронні мережі чутливі до діапазону вхідних даних, значення середньої вартості нормалізуються до інтервалу від 0 до 1. Це сприяє стабільнішому навчанню моделі та прискорює процес оптимізації вагових коефіцієнтів.

Після масштабування часовий ряд перетворюється на набір навчальних послідовностей. Для кожного прогнозованого значення використовується певна кількість попередніх спостережень, які формують вхідне вікно моделі. Наприклад, якщо довжина вікна становить 10 часових кроків, то для прогнозування наступного значення використовуються десять попередніх значень ряду. У результаті формується множина навчальних прикладів, придатних для подальшого навчання нейронної мережі.

Основою реалізованої моделі є шар LSTM, який містить спеціальні комірки пам'яті. На відміну від звичайних рекурентних нейронних мереж, LSTM здатна зберігати інформацію протягом тривалих часових інтервалів і враховувати довгострокові залежності між спостереженнями [11]. Це особливо важливо для аналізу часових рядів, у яких поточні значення можуть залежати від подій, що відбувалися значно раніше.

Архітектура LSTM базується на використанні трьох механізмів керування інформацією: воріт забування (forget gate), вхідних воріт (input gate) та вихідних воріт (output gate). Ворота забування визначають, яку інформацію необхідно видалити з пам'яті мережі. Вхідні ворота контролюють надходження нових даних до комірки пам'яті, а вихідні ворота формують результат роботи мережі на поточному часовому кроці. Завдяки такій структурі модель може ефективно відокремлювати корисну інформацію від випадкового шуму та зберігати важливі закономірності протягом тривалого часу.

У програмній реалізації модель створена за допомогою бібліотеки TensorFlow/Keras [5]. Архітектура складається з LSTM-шару та вихідного

повнозв'язного шару (Dense), який формує прогнозоване значення. Для навчання використовується алгоритм оптимізації Adam, що забезпечує ефективне налаштування параметрів нейронної мережі та швидку збіжність процесу навчання.

Під час навчання модель отримує послідовності попередніх значень часового ряду та поступово мінімізує похибку між прогнозованими й фактичними значеннями. Після завершення процесу навчання мережа використовується для прогнозування значень на тестовій вибірці. Отримані результати перетворюються назад до початкового масштабу, що дозволяє інтерпретувати їх у вигляді реальних значень вартості нерухомості.

Для оцінювання якості роботи моделі використовується порівняння прогнозованих і фактичних значень часового ряду. Результати візуалізуються за допомогою графіків, на яких відображаються реальна динаміка середньої вартості нерухомості та прогноз, сформований нейронною мережею. Такий підхід дозволяє наочно оцінити здатність моделі відтворювати основні тенденції ринку та виявляти можливі відхилення між прогнозом і реальними даними.

Додатково виконується аналіз похибок прогнозування. Для цього розраховуються значення помилок між фактичними та прогнозованими даними, а також досліджується їх розподіл. Аналіз похибок дає змогу оцінити стабільність роботи моделі та визначити періоди, у яких прогнозування виконується з найбільшою або найменшою точністю.

Таким чином, реалізована LSTM-модель є додатковим інструментом дослідження ринку нерухомості. Якщо ансамблеві алгоритми прогнозують вартість конкретного об'єкта на основі його характеристик, то LSTM дозволяє аналізувати часові закономірності зміни середньої вартості житла. Поєднання цих підходів забезпечує комплексний аналіз ринку та дозволяє отримувати як індивідуальні оцінки об'єктів, так і прогнози загальної динаміки цін.

3.5 Використані програмні бібліотеки та засоби розробки

Для реалізації програмної системи прогнозування вартості нерухомості було використано сучасні засоби розробки та бібліотеки машинного навчання мовою програмування Python. Вибір Python обумовлений широкими можливостями для аналізу даних, побудови моделей машинного навчання, візуалізації результатів та створення програмних систем для обробки великих обсягів інформації.

Основним середовищем розробки виступає Python, який є однією з найпоширеніших мов програмування у сфері аналізу даних та штучного інтелекту. Завдяки великій кількості спеціалізованих бібліотек Python дозволяє реалізувати повний цикл роботи з даними: від їх завантаження та попередньої обробки до навчання моделей і візуалізації результатів [13].

Для роботи з табличними даними використовується бібліотека **pandas**. Вона забезпечує зручні інструменти для завантаження наборів даних, фільтрації записів, обробки пропущених значень, формування нових ознак та виконання статистичного аналізу. У межах даної роботи бібліотека застосовується під час підготовки набору даних, створення часових характеристик та формування вибірок для навчання моделей [7].

Важливу роль у реалізації системи відіграє бібліотека **NumPy**, яка забезпечує ефективну роботу з багатовимірними масивами та математичними операціями. Використання NumPy дозволяє виконувати швидкі обчислення над великими наборами даних і є основою роботи багатьох інших бібліотек машинного навчання [8].

Для побудови та оцінювання моделей машинного навчання використовується бібліотека **scikit-learn**. Вона містить реалізації численних алгоритмів класифікації, регресії та аналізу даних, а також інструменти для

підготовки вибірок і розрахунку показників якості моделей. У розробленій системі `scikit-learn` застосовується для реалізації лінійної регресії, поділу даних на навчальну та тестову вибірки, а також для обчислення метрик оцінювання прогнозів [6].

Однією з основних моделей прогнозування є **XGBoost**. Дана бібліотека реалізує алгоритм градієнтного бустингу над деревами рішень та широко використовується у задачах прогнозування на табличних даних. Перевагами XGBoost є висока точність, можливість роботи з великими наборами даних та ефективне використання обчислювальних ресурсів [2].

Для побудови моделей також використовується бібліотека **CatBoost**, яка спеціально розроблена для роботи з категоріальними ознаками. Однією з її переваг є можливість автоматичної обробки категоріальних даних без необхідності складного попереднього кодування. Це дозволяє підвищити якість прогнозування та спростити процес підготовки даних [3].

Ще одним інструментом прогнозування є **LightGBM**. Бібліотека реалізує алгоритм градієнтного бустингу з використанням оптимізованих структур дерев рішень, що забезпечує високу швидкість навчання навіть на великих наборах даних. Завдяки ефективному використанню пам'яті та швидкому навчанню LightGBM часто застосовується у практичних задачах прогнозування [4].

Для аналізу часових залежностей використовується бібліотека **TensorFlow** та її високорівневий інтерфейс **Keras**. За допомогою цих інструментів реалізована LSTM-модель, яка використовується для прогнозування зміни середньої вартості нерухомості у часі. TensorFlow забезпечує побудову та навчання нейронних мереж, а Keras спрощує процес створення архітектури моделі та налаштування параметрів навчання [5], [11].

Візуалізація результатів роботи системи здійснюється за допомогою бібліотек **Matplotlib** та **Plotly**. Бібліотека Matplotlib використовується для

побудови статичних графіків, які відображають результати прогнозування, розподіл помилок та важливість ознак. Plotly дозволяє створювати інтерактивні візуалізації, що спрощують аналіз отриманих результатів та підвищують наочність представлення даних [9], [10].

Застосування зазначених бібліотек дозволило реалізувати повний цикл роботи системи прогнозування вартості нерухомості: від завантаження та підготовки даних до навчання моделей, оцінювання їхньої якості та візуалізації отриманих результатів. Використання сучасних програмних засобів забезпечує ефективність роботи системи, можливість подальшого розширення функціональності та відтворюваність отриманих результатів дослідження.

Висновки до розділу 3

У третьому розділі було описано програмну реалізацію системи прогнозування вартості нерухомості. Для розробки використано мову Python та бібліотеки для обробки даних, машинного навчання, нейронних мереж і візуалізації результатів.

Система має модульну структуру та виконує повний цикл роботи з даними: завантаження, попередню обробку, формування ознак, навчання моделей, оцінювання якості та побудову графіків. Для прогнозування ціни окремого об'єкта реалізовано Linear Regression, XGBoost, CatBoost та LightGBM, а для аналізу часової динаміки середньої вартості використано LSTM-модель.

Реалізований підхід дозволяє порівнювати результати різних моделей, визначати найточнішу з них та аналізувати якість прогнозування за допомогою числових метрик і графічних результатів.

РОЗДІЛ 4 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1 Порівняння якості моделей

Після навчання моделей було виконано порівняння їхньої якості на тестовій вибірці. Для цього фактичні ціни нерухомості порівнювались із прогнозованими значеннями, після чого для кожної моделі були обчислені метрики MAE, MSE, RMSE, MAPE, sMAPE та R^2 . Такий підхід дозволяє оцінити не лише загальну точність прогнозування, а й величину похибки в абсолютному та відносному вигляді.

Базовою моделлю для порівняння є Linear Regression. Вона має просту структуру та дозволяє оцінити лінійну залежність між характеристиками об'єкта і його ціною. Однак через складний характер ринку нерухомості така модель може не повністю враховувати нелінійні зв'язки між ознаками.

Кращі результати очікувано показують ансамблеві моделі XGBoost, CatBoost та LightGBM. Вони здатні враховувати складніші залежності між параметрами будинку, його розташуванням, станом і ціною. Порівняння моделей виконується за таблицею метрик, де менші значення MAE, RMSE і MAPE свідчать про меншу похибку, а більше значення R^2 — про кращу здатність моделі пояснювати зміну ціни.

За результатами порівняння обирається модель, яка має найкраще поєднання низької похибки та високого значення R^2 . Саме її доцільно використовувати як основну модель для прогнозування вартості нерухомості в програмній системі. Результати порівняння досліджуваних моделей за основними показниками якості прогнозування наведено в таблиці 4.1.

Таблиця 4.1 – Порівняння якості моделей прогнозування

<i>Модель</i>	<i>MAE</i>	<i>MSE</i>	<i>RMSE</i>	<i>MAPE, %</i>	<i>sMAPE, %</i>	<i>R²</i>
Linear Regression	268459.08	571304600000.00	755846.91	61.68	76.82	0.444
XGBoost	145781.86	277887800000.00	527150.65	18.26	36.39	0.730
CatBoost	147094.89	277961600000.00	527220.62	19.83	38.66	0.729
LightGBM	162631.17	293725200000.00	541964.16	19.71	40.86	0.714

За результатами таблиці 4.1 найкращі показники серед табличних моделей отримала модель XGBoost. Вона має найменші значення MAE, RMSE, MAPE та sMAPE, а також найбільше значення коефіцієнта детермінації R^2 . Це означає, що XGBoost точніше прогнозує вартість нерухомості порівняно з Linear Regression, CatBoost та LightGBM. Найгірший результат показала Linear Regression, що пояснюється її обмеженою здатністю враховувати нелінійні залежності між характеристиками об'єкта та його ціною. Для наочного порівняння ефективності різних моделей прогнозування побудовано графічну візуалізацію отриманих результатів, яку наведено на рисунку 4.1.

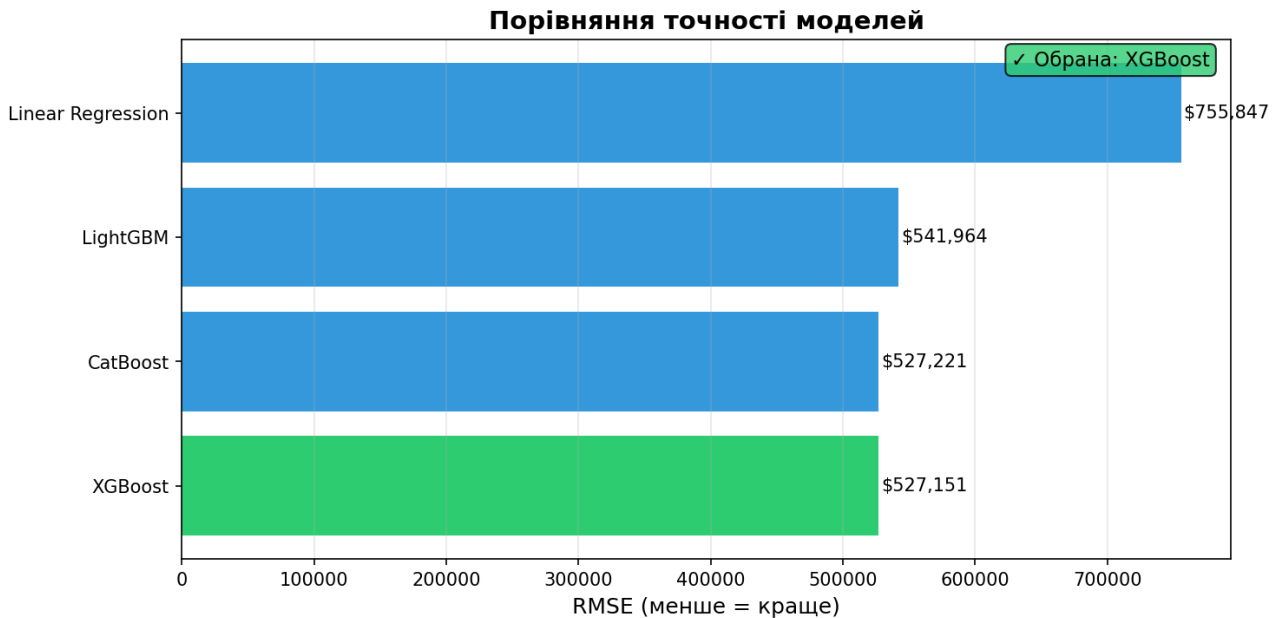


Рисунок 4.1 – Порівняння моделей прогнозування за основними метриками

На рисунку 4.1 наведено графічне порівняння якості моделей прогнозування. Візуалізація підтверджує результати таблиці 4.1: ансамблеві моделі мають помітно кращу якість прогнозування порівняно з лінійною регресією. Найкращі результати показала модель XGBoost, оскільки вона має найменшу похибку та найбільше значення R^2 серед розглянутих моделей.

4.2 Аналіз похибок прогнозування

Для детальнішої оцінки якості прогнозування було проаналізовано похибки моделі. Похибка визначається як різниця між фактичною та прогнозованою ціною об'єкта нерухомості. Такий аналіз дозволяє виявити, наскільки стабільно модель працює на тестових даних і чи є випадки значного відхилення прогнозу від реального значення.

Основну увагу приділено моделі XGBoost, оскільки вона показала найкращі результати за основними метриками. Аналіз фактичних і прогнозованих значень дозволяє оцінити, наскільки близько прогнози моделі

розташовані до реальних цін. Чим менше відхилення між цими значеннями, тим точнішою є модель.

Окремо розглядається розподіл залишків, тобто різниць між фактичними та прогнозованими значеннями. Якщо залишки не мають чіткої систематичної структури, це свідчить про те, що модель не допускає сталої помилки в один бік. Наявність великих залишків може бути пов'язана з дорогими або нетиповими об'єктами, ціна яких суттєво відрізняється від більшості записів у наборі даних.

Таким чином, аналіз похибок показує не лише середню точність моделі, а й характер її помилок. Це важливо для практичного використання системи, оскільки користувач повинен розуміти, що прогноз є наближеним значенням і може мати більшу похибку для нестандартних об'єктів нерухомості. Відповідність між фактичними та прогнозованими значеннями вартості нерухомості для обраної моделі представлено на рисунку 4.2.

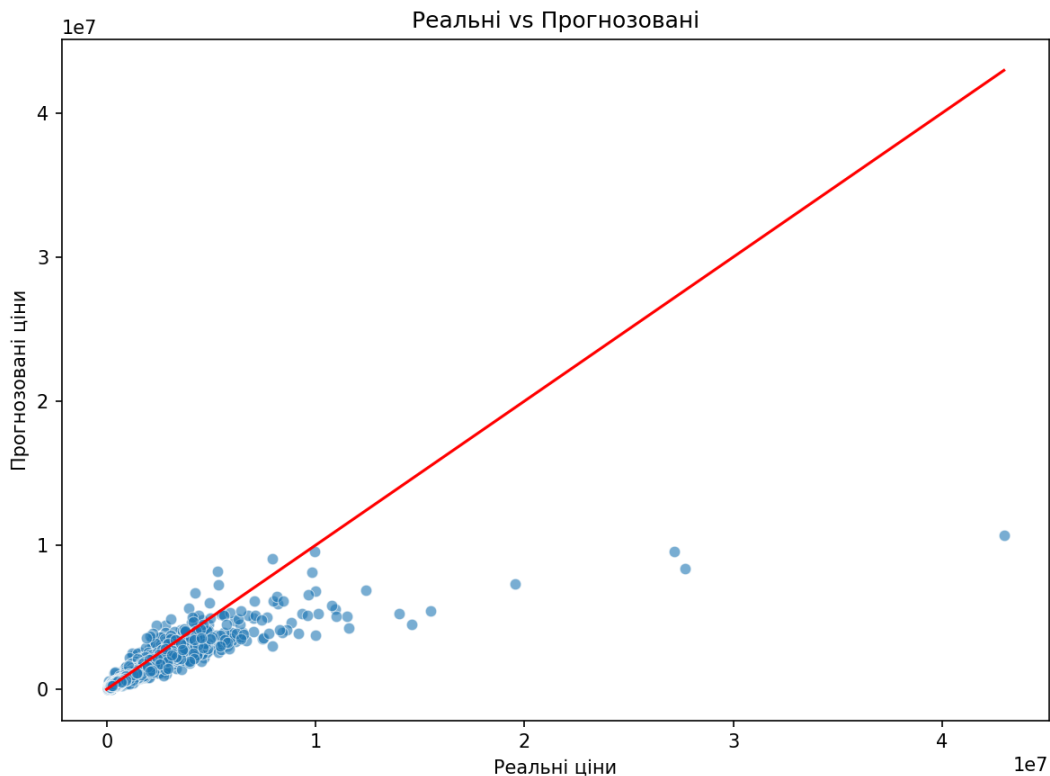


Рисунок 4.2 – Порівняння фактичних і прогнозованих значень

Для додаткового аналізу якості прогнозування було досліджено розподіл похибок моделі, який наведено на рисунку 4.3.

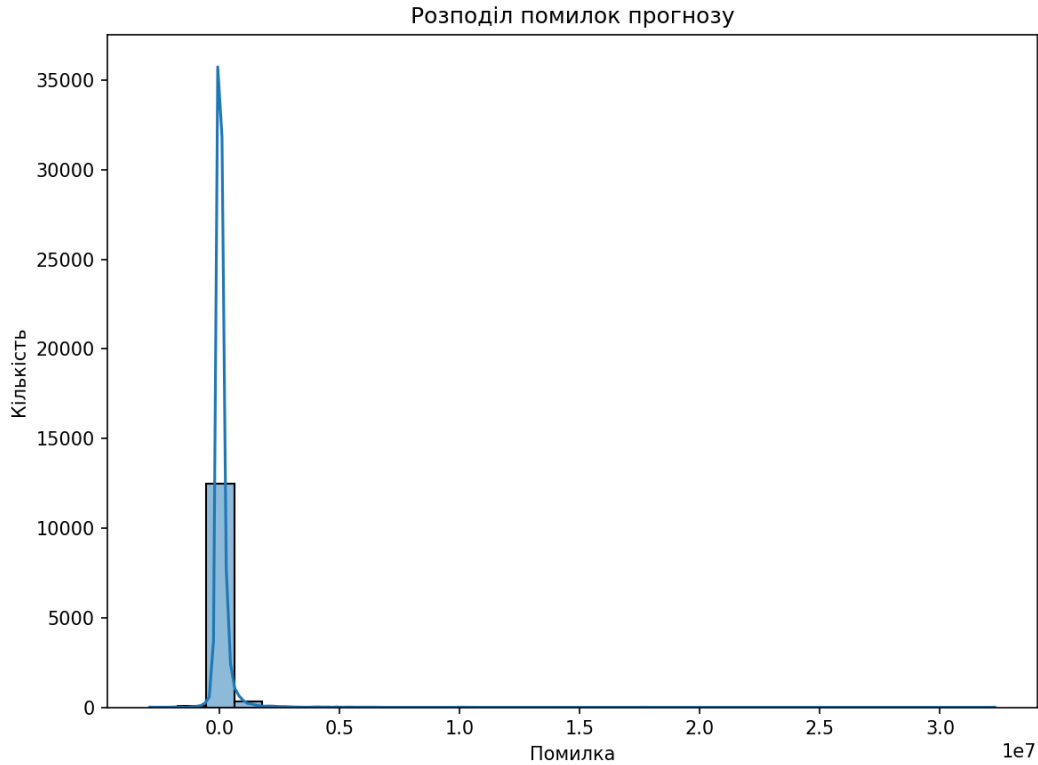


Рисунок 4.3 – Розподіл похибок прогнозування

4.3 Аналіз важливості ознак

Для кращого розуміння принципів роботи моделі було виконано аналіз важливості ознак. Однією з переваг ансамблевих моделей на основі дерев рішень є можливість оцінювання внеску кожної ознаки у формування кінцевого прогнозу. Такий аналіз дозволяє визначити, які характеристики нерухомості найбільше впливають на прогнозовану вартість об'єкта та які фактори є найважливішими під час прийняття рішення моделлю [2], [3], [4].

Оцінювання важливості ознак є важливим етапом інтерпретації результатів машинного навчання. Воно дає змогу не лише оцінити якість моделі, а й перевірити, чи відповідають виявлені закономірності реальним особливостям ринку нерухомості. Якщо серед найбільш значущих ознак переважають характеристики, які традиційно впливають на вартість житла, це свідчить про коректність побудованої моделі та адекватність використаних даних.

На рисунку 4.4 наведено результати аналізу важливості ознак, отримані під час навчання моделі. Для зручності часові характеристики виділено окремим кольором, що дозволяє оцінити їхній внесок у процес прогнозування порівняно з іншими параметрами.

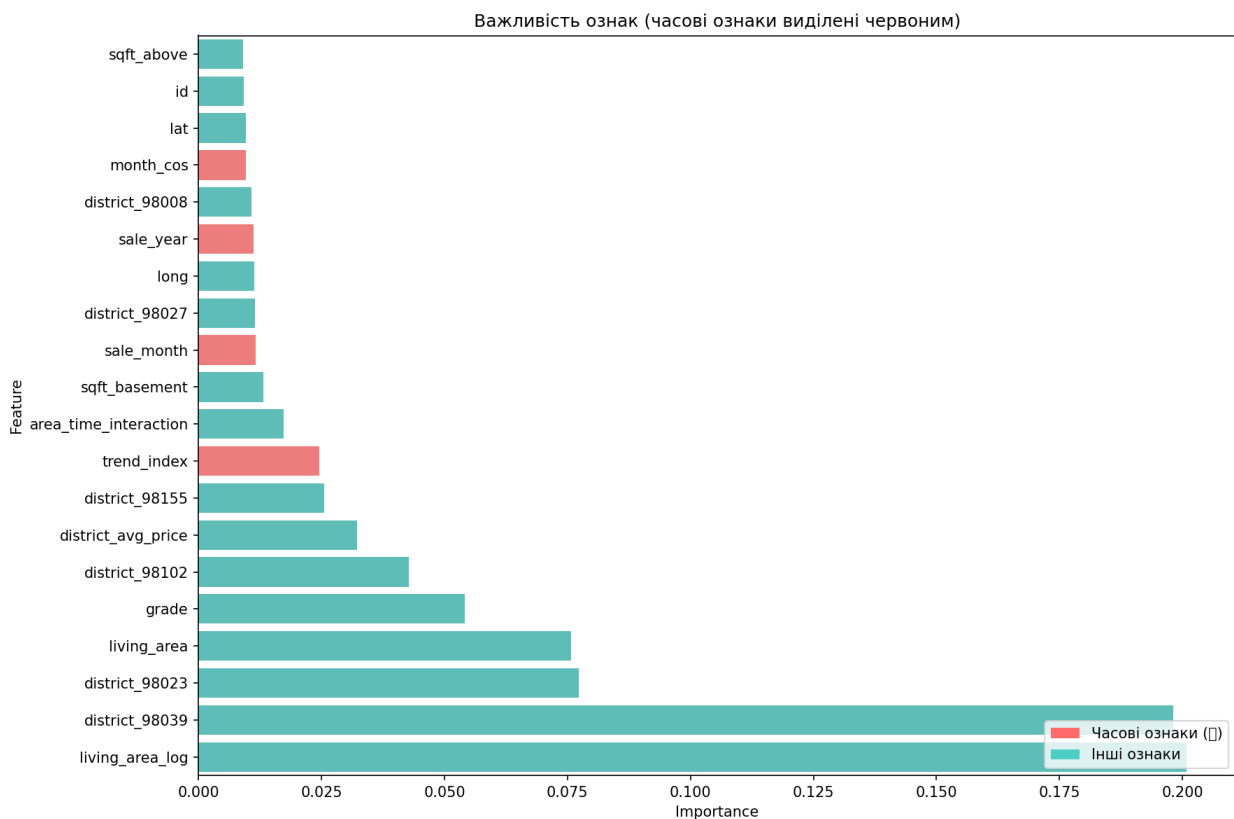


Рисунок 4.4 – Важливість ознак для прогнозування вартості нерухомості

Згідно з отриманими результатами, найбільший вплив на прогноз має ознака **living_area_log**. Вона є логарифмічним перетворенням житлової

площі будинку та демонструє найвищий рівень важливості серед усіх характеристик. Отриманий результат є цілком очікуваним, оскільки площа житла традиційно вважається одним із головних факторів, що визначають ринкову вартість нерухомості. Використання логарифмічного перетворення дозволяє краще враховувати нелінійний характер залежності між площею будинку та його ціною.

Другу групу важливих ознак формують характеристики, пов'язані з місцем розташування об'єкта. Зокрема, значний внесок мають ознаки **district_98039**, **district_98023** та **district_98102**, які відповідають окремим поштовим районам. Це свідчить про те, що географічне розташування є одним із ключових факторів формування вартості житла. Навіть об'єкти зі схожими фізичними характеристиками можуть суттєво відрізнятися за ціною залежно від району розташування, рівня розвитку інфраструктури та транспортної доступності.

Високу важливість також демонструє ознака **living_area**, яка безпосередньо характеризує житлову площу будинку. Одночасна наявність серед лідерів як вихідної площі, так і її логарифмічного перетворення свідчить про те, що модель активно використовує інформацію про розміри об'єкта під час побудови прогнозу. Це підтверджує значущість даного параметра для задачі оцінювання вартості нерухомості.

Серед найбільш впливових характеристик також присутня ознака **grade**, яка відображає загальний рівень якості будівництва та конструктивних рішень будинку. Отримані результати узгоджуються з особливостями ринку нерухомості, оскільки будинки з вищою якістю матеріалів, сучасними архітектурними рішеннями та кращим технічним станом зазвичай мають вищу ринкову вартість.

Помітний вплив на результати прогнозування має ознака **district_avg_price**, яка характеризує середній рівень цін у відповідному

районі. Її висока важливість свідчить про те, що модель успішно враховує локальні особливості ринку нерухомості та використовує інформацію про загальний рівень цін у конкретній географічній зоні.

До групи ознак середньої важливості належать **district_98155**, **trend_index**, **area_time_interaction** та **sqft_basement**. Ознака **trend_index** відображає часову динаміку ринку та належить до групи часових характеристик. Її внесок демонструє, що модель враховує не лише характеристики окремих об'єктів, а й загальні тенденції зміни вартості нерухомості у часі. Ознака **area_time_interaction** описує взаємодію між часовими факторами та площею житла, що дозволяє враховувати можливі зміни впливу площі на ціну залежно від ринкової ситуації.

Порівняно менший, але все ж помітний внесок мають географічні координати **lat** та **long**. Незважаючи на те, що їхня важливість нижча за спеціально сформовані районні ознаки, вони додатково допомагають моделі враховувати просторове розташування об'єкта та особливості навколишнього середовища.

Окремий інтерес становить аналіз часових характеристик, які на рисунку виділені червоним кольором. До них належать **sale_month**, **sale_year**, **month_cos** та **trend_index**. Отримані результати показують, що часові ознаки мають менший вплив порівняно з характеристиками площі та розташування, проте їхній внесок залишається помітним. Це свідчить про наявність часових закономірностей на ринку нерухомості та підтверджує доцільність використання часових характеристик під час побудови прогнозних моделей.

Загалом результати аналізу важливості ознак підтверджують, що модель формує прогноз на основі факторів, які дійсно впливають на ринкову вартість житла. Найбільше значення мають площа будинку, його якість та місце розташування, що узгоджується з теоретичними положеннями,

розглянутими у першому розділі роботи. Водночас додаткові часові та похідні характеристики також роблять помітний внесок у підвищення точності прогнозування.

Таким чином, аналіз важливості ознак дозволив підтвердити адекватність побудованої моделі та визначити ключові фактори, які впливають на формування вартості нерухомості. Отримані результати можуть бути використані не лише для інтерпретації прогнозів, а й для подальшого вдосконалення системи шляхом розробки нових інформативних характеристик та оптимізації набору ознак.

4.4 Аналіз результатів LSTM-моделі

Окремо було проаналізовано результати LSTM-моделі, яка використовується для дослідження часової динаміки середньої вартості нерухомості. На відміну від моделей Linear Regression, XGBoost, CatBoost та LightGBM, ця модель працює не з окремими об'єктами нерухомості, а з послідовністю середніх цін у часі.

Перед навчанням дані були згруповані за датою продажу, після чого було сформовано часовий ряд середньої ціни. На основі попередніх значень цього ряду LSTM-модель прогнозує наступні значення. Такий підхід дозволяє оцінити загальний напрям зміни вартості нерухомості та виявити тенденції у часових даних.

Результати LSTM-моделі оцінюються за графіками фактичних і прогнозованих значень, а також за розподілом похибок. Якщо прогнозована крива повторює загальну форму фактичної динаміки, це свідчить про здатність моделі враховувати часові закономірності. Водночас окремі

відхилення можуть виникати через нерівномірність продажів, різну кількість об'єктів у часових періодах та різкі зміни середньої ціни.

Таким чином, LSTM-модель доповнює основні моделі прогнозування. Вона не замінює моделі оцінювання ціни конкретного будинку, а використовується для аналізу загального тренду ринку нерухомості у часі. Результати роботи LSTM-моделі у вигляді порівняння фактичних і прогнозованих значень наведено на рисунку 4.5

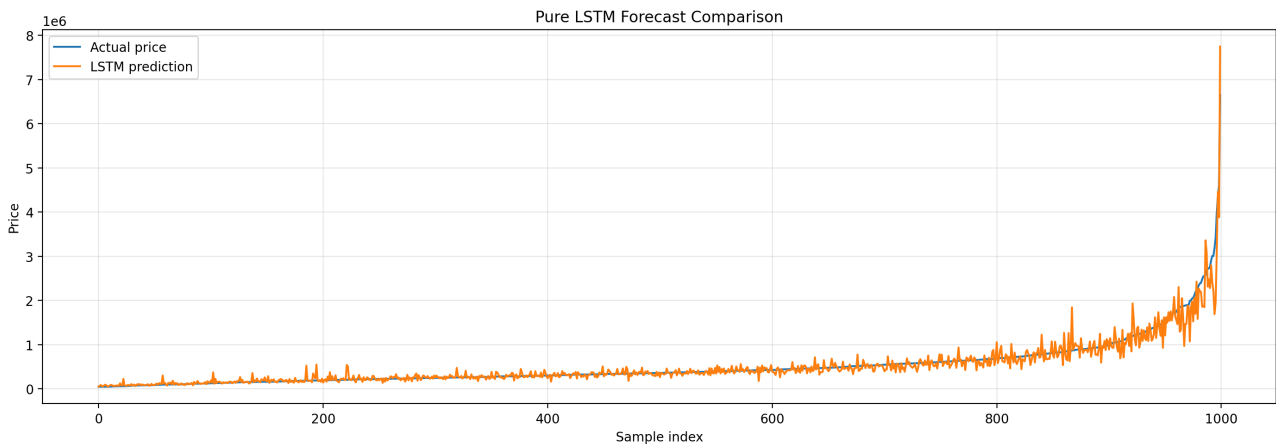


Рисунок 4.5 – Порівняння фактичних і прогнозованих значень LSTM-моделі

Оцінити стабільність прогнозів LSTM-моделі дозволяє аналіз абсолютної відсоткової похибки, розподіл якої наведено на рисунку 4.6.

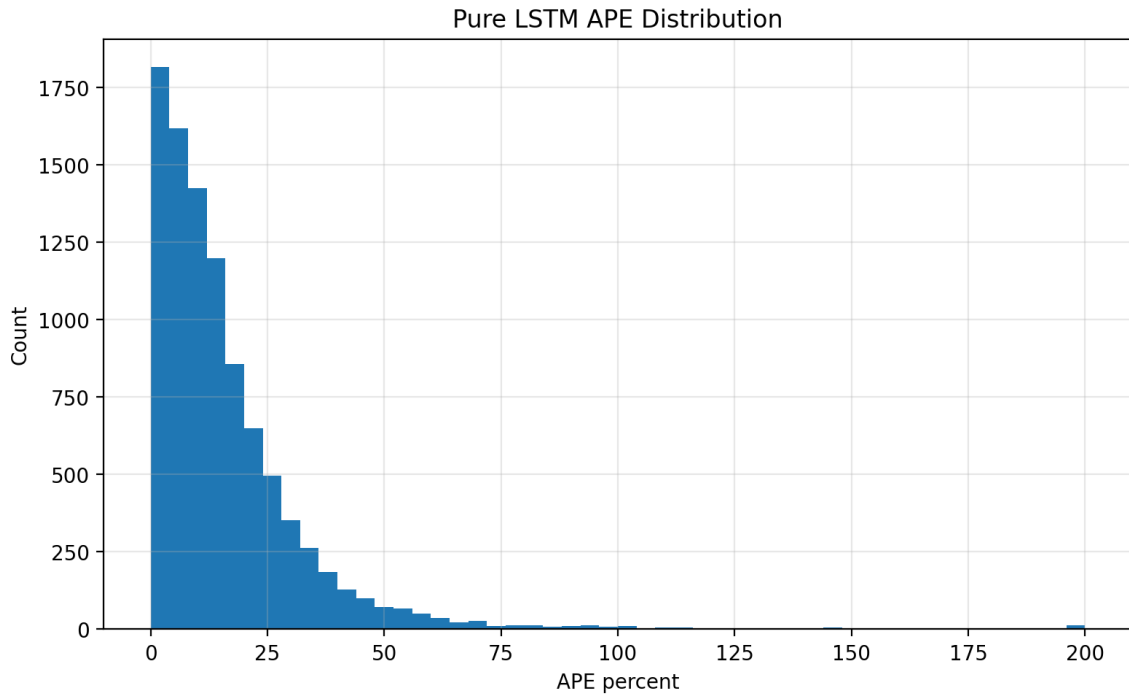


Рисунок 4.6 – Розподіл абсолютної відсоткової похибки LSTM-моделі

Прогноз часової динаміки середньої вартості нерухомості, отриманий за допомогою LSTM-моделі, наведено на рисунку 4.7.

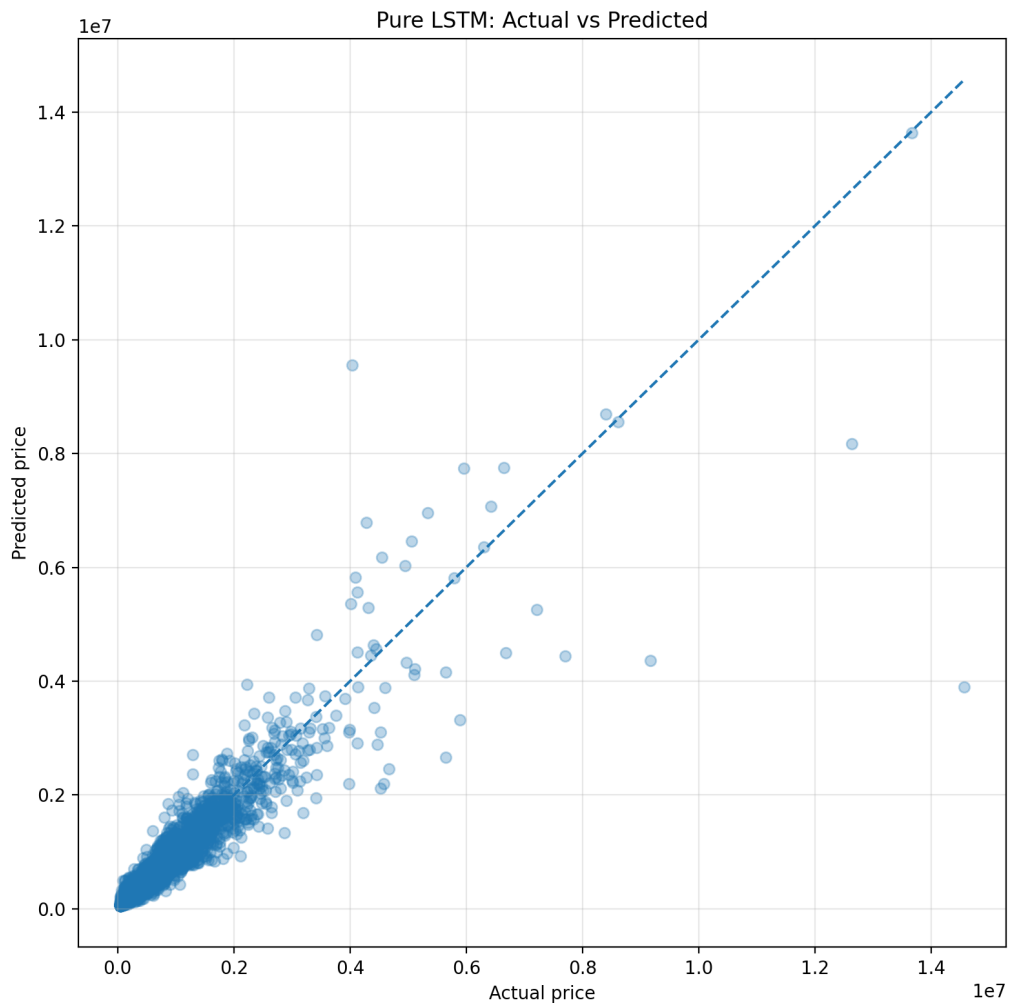


Рисунок 4.7 – Прогноз часової динаміки середньої вартості нерухомості

Для додаткової перевірки отриманих результатів було виконано аналіз дисперсії ANOVA (Analysis of Variance), який дозволяє оцінити статистичну значущість окремих факторів щодо формування ринкової вартості нерухомості. На відміну від аналізу важливості ознак у моделях градієнтного бустингу, ANOVA дає змогу оцінити, чи є відмінності між групами статистично значущими та яка частка варіації цільової змінної пояснюється відповідною ознакою.

Результати ANOVA підтвердили, що найбільший вплив на вартість нерухомості мають ознаки, пов'язані з якістю будинку (grade), його технічним станом (condition), місцем розташування (zipcode), а також характеристиками

огляду та близькості до водойми (view та waterfront). Для зазначених факторів отримано високі значення F-статистики та статистично значущі значення p-value ($p < 0,05$), що свідчить про наявність суттєвих відмінностей між відповідними групами об'єктів нерухомості.

Отримані результати узгоджуються з аналізом важливості ознак, виконаним за допомогою моделей XGBoost, CatBoost та LightGBM. Найбільш значущими факторами виявилися характеристики, які безпосередньо впливають на споживчу цінність житла та його ринкову привабливість. Це підтверджує коректність сформованого набору ознак та адекватність побудованих моделей прогнозування. Результати дисперсійного аналізу наведено в таблиці 4.3.

Таблиця 4.2 – Результати ANOVA для основних категоріальних ознак

<i>Ознака</i>	<i>Статистична значущість</i>	<i>Розмір ефекту η^2</i>	<i>Інтерпретація</i>
grade	Так	Високий	Дуже сильний вплив
zipcode	Так	Високий	Дуже сильний вплив
condition	Так	Середній	Помітний вплив
view	Так	Середній	Помітний вплив
waterfront	Так	Середній	Помітний вплив

Висновки до розділу 4

У четвертому розділі було проаналізовано результати роботи програмної системи прогнозування вартості нерухомості. Для порівняння моделей використано набір метрик, що дозволяє оцінити абсолютну, середньоквадратичну та відносну похибку прогнозування.

За результатами порівняння найкращі показники серед табличних моделей отримала модель XGBoost. Вона має найменші значення MAE, RMSE, MAPE та sMAPE, а також найбільше значення R^2 . Це свідчить про те, що ця модель краще враховує складні нелінійні залежності між характеристиками об'єкта нерухомості та його ціною.

Аналіз похибок показав, що прогноз є наближеним значенням і може мати більші відхилення для нетипових або дуже дорогих об'єктів. Аналіз важливості ознак підтвердив, що на результат прогнозування найбільше впливають характеристики, пов'язані з площею, якістю будинку та його розташуванням.

Окремо було розглянуто результати LSTM-моделі для аналізу часової динаміки середньої вартості нерухомості. Вона дозволяє оцінити загальну тенденцію зміни цін у часі та доповнює основні моделі, які прогнозують вартість конкретного об'єкта.

ВИСНОВКИ

У роботі було розроблено програмну систему для аналізу та прогнозування вартості житлової нерухомості на основі методів машинного навчання і нейронних мереж. Для цього використано набір даних про продажі будинків у King County, який містить фактичні ціни продажу та характеристики об'єктів нерухомості.

У процесі виконання роботи було проаналізовано предметну область прогнозування вартості нерухомості, визначено основні фактори, що впливають на ціну, а також сформульовано постановку задачі дослідження. Задачу прогнозування ціни розглянуто як задачу регресії, у якій вхідними даними є характеристики об'єкта, а вихідним значенням — прогнозована ціна продажу.

Було виконано попередню обробку даних, зокрема обробку дати продажу, заповнення пропущених значень, обмеження викидів, перетворення категоріальних ознак і формування додаткових характеристик. Це дозволило підготувати дані до навчання моделей та підвищити якість прогнозування.

Для прогнозування вартості нерухомості було реалізовано кілька моделей: Linear Regression, XGBoost, CatBoost та LightGBM. Також окремо реалізовано LSTM-модель для аналізу часової динаміки середньої вартості нерухомості. Якість моделей оцінювалася за набором метрик, зокрема MAE, MSE, RMSE, MAPE, sMAPE та R^2 .

За результатами експериментів найкращі показники серед табличних моделей отримала модель XGBoost. Вона показала найменші значення похибок і найбільше значення коефіцієнта детермінації R^2 , тому її доцільно використовувати як основну модель для прогнозування вартості нерухомості в розробленій системі.

Аналіз похибок показав, що модель загалом добре відтворює залежність між характеристиками об'єкта та його ціною, однак для

нетипових або дуже дорогих об'єктів похибка може зростати. Аналіз важливості ознак підтвердив, що значний вплив на прогноз мають площа, якість будинку, його стан, розташування та характеристики сусідніх об'єктів.

Реалізована LSTM-модель дозволила проаналізувати часову зміну середньої вартості нерухомості. Вона використовується як додатковий інструмент для дослідження загальної динаміки ринку та доповнює основні моделі, які прогнозують ціну конкретного об'єкта.

Отже, поставлену мету роботи досягнуто. Розроблена система виконує повний цикл аналізу: завантаження та підготовку даних, навчання моделей, оцінювання їхньої точності, побудову графіків і формування прогнозу вартості нерухомості. Практичне значення роботи полягає у можливості автоматизованого оцінювання орієнтовної ціни житлової нерухомості та аналізу факторів, які найбільше впливають на результат прогнозування.

ПЕРЕЛІК ПОСИЛАНЬ

1. House Sales in King County, USA. Kaggle Dataset. URL: <https://www.kaggle.com/datasets/harlfoxem/housesalesprediction>
2. XGBoost Documentation. Python API Reference. URL: https://xgboost.readthedocs.io/en/latest/python/python_api.html
3. CatBoost Documentation. URL: <https://catboost.ai/docs/en/>
4. LightGBM Documentation. Python API. URL: <https://lightgbm.readthedocs.io/en/latest/Python-API.html>
5. TensorFlow Documentation. tf.keras.layers.LSTM. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
6. Scikit-learn Documentation. Metrics and scoring: quantifying the quality of predictions. URL: https://scikit-learn.org/stable/modules/model_evaluation.html
7. pandas Documentation. User Guide. URL: https://pandas.pydata.org/docs/user_guide/
8. NumPy Documentation. URL: <https://numpy.org/doc/>
9. Matplotlib Documentation. URL: <https://matplotlib.org/stable/index.html>
10. Plotly Python Documentation. URL: <https://plotly.com/python/>

11. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997. Vol. 9, No. 8. P. 1735–1780.
12. Hyndman R. J., Koehler A. B. Another look at measures of forecast accuracy. *International Journal of Forecasting*. 2006. Vol. 22, No. 4. P. 679–688.
13. James G., Witten D., Hastie T., Tibshirani R. *An Introduction to Statistical Learning: with Applications in Python*. Springer, 2023.
14. Peterson S., Flanagan A. Neural Network Hedonic Pricing Models in Mass Real Estate Appraisal // *Journal of Real Estate Research*. 2009. Vol. 31, No. 2. P. 147–164.
15. Montgomery D. C. *Design and Analysis of Experiments*. 10th ed. Hoboken : John Wiley & Sons, 2019. 752 p.
16. Аналіз нестационарних і нелінійних процесів в економіці та фінансах
Бідюк П.І., Романенко В.Д., Тимощук О.Л. Аналіз часових рядів. –
Київ: НТУУ «КПІ», 2012. – 600 с. URL: <https://ela.kpi.ua/handle/123456789/862>.
17. Аналіз нестационарних і нелінійних процесів в економіці та фінансах
[Електронний ресурс] : навч. посіб. для студентів спеціальності F4
Системний аналіз та наука про дані / КПІ ім. Ігоря Сікорського ; уклад.: П. І.
Бідюк, О. Л. Тимощук, В. Г. Гуськова, Т. І. Просянкіна-Жарова, Л. Б.
Левенчук. – Електрон. текст. дані (1 файл: 14,57 Мбайт). – Київ : КПІ ім.
Ігоря Сікорського, 2025. – 378 с. URI
<https://ela.kpi.ua/handle/123456789/77009>

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

preprocessing.py

```
import numpy as np

import pandas as pd


def clean_numeric_columns(dataframe: pd.DataFrame) -> pd.DataFrame:
    cleaned = dataframe.copy()

    for column in cleaned.columns:
        if "date" in str(column).lower():
            continue

        if pd.api.types.is_numeric_dtype(cleaned[column]):
            continue

        series_as_text = cleaned[column].astype("string")
        looks_numeric = series_as_text.str.contains(r"\d", na=False).mean()

        if looks_numeric < 0.4:
            continue

        normalized = (
            series_as_text
            .str.replace(" ", "", regex=False)
            .str.replace(",", "", regex=False)
            .str.replace(r"^[^0-9.\-]", "", regex=True)
        )

        converted = pd.to_numeric(normalized, errors="coerce")

        if converted.notna().mean() > 0.6:
            cleaned[column] = converted

    return cleaned
```

```

def process_sale_date(dataframe: pd.DataFrame) -> pd.DataFrame:
    processed = dataframe.copy()
    if "date" in processed.columns and "sale_date" not in processed.columns:
        processed["sale_date"] = processed["date"]
    if "sale_date" in processed.columns:
        parsed_dates = pd.to_datetime(processed["sale_date"], errors="coerce")
        if parsed_dates.isna().mean() > 0.5:
            raw_values = processed["sale_date"].astype(str).str.replace(".", "", regex=False).str.strip()
            parsed_dates = pd.to_datetime(raw_values, format="%Y%m%dT%H%M%S", errors="coerce")
            processed["sale_date"] = parsed_dates
        processed = clean_numeric_columns(processed)
    if "sale_year" in processed.columns:
        processed["sale_year"] = pd.to_numeric(processed["sale_year"], errors="coerce")
    if "sale_year" not in processed.columns:
        processed["sale_year"] = pd.NA
    if "sale_date" in processed.columns:
        sale_year_from_date = processed["sale_date"].dt.year
        processed["sale_year"] = pd.to_numeric(processed["sale_year"], errors="coerce")
        sale_year_from_date = pd.to_numeric(sale_year_from_date, errors="coerce")
        processed["sale_year"] = (
            processed["sale_year"]
            .fillna(sale_year_from_date)
            .infer_objects(copy=False)
        )
    if "sale_date" not in processed.columns or processed["sale_date"].isna().all():
        year_values = pd.to_numeric(processed["sale_year"], errors="coerce")

```

```
processed["sale_date"] = pd.to_datetime(year_values.astype("Int64").astype("string") +
"-01-01", errors="coerce")
```

```
elif processed["sale_date"].isna().any():
```

```
year_values = pd.to_numeric(processed["sale_year"], errors="coerce")
```

```
fallback_dates = pd.to_datetime(year_values.astype("Int64").astype("string") + "-01-01",
errors="coerce")
```

```
processed["sale_date"] = processed["sale_date"].fillna(fallback_dates)
```

```
if "district" in processed.columns:
```

```
processed["district"] = processed["district"].astype("string")
```

```
return processed
```

```
def fill_missing_values(dataframe: pd.DataFrame) -> pd.DataFrame:
```

```
filled = clean_numeric_columns(dataframe.copy())
```

```
numeric_columns = filled.select_dtypes(include=[np.number]).columns
```

```
for column in numeric_columns:
```

```
filled[column] = filled[column].fillna(filled[column].median())
```

```
categorical_columns = filled.select_dtypes(include=["object", "category", "string"]).columns
```

```
for column in categorical_columns:
```

```
mode_values = filled[column].mode(dropna=True)
```

```
replacement = mode_values.iloc[0] if not mode_values.empty else "unknown"
```

```
filled[column] = filled[column].fillna(replacement)
```

```
return filled
```

```
def compute_preprocessing_statistics(
```

```
dataframe: pd.DataFrame,
```

```
exclude_columns: list[str] | None = None,
```

```
multiplier: float = 1.5,
```

```

) -> dict:

exclude_columns = exclude_columns or []

prepared = process_sale_date(dataframe)

stats: dict[str, object] = {
    "numeric_medians": {},
    "categorical_modes": {},
    "outlier_bounds": {},
    "exclude_columns": exclude_columns,
}

numeric_columns = [
    column
    for column in prepared.select_dtypes(include=[np.number]).columns
    if column not in exclude_columns and column != "district"
]

categorical_columns = [
    column
    for column in prepared.select_dtypes(include=["object", "category", "string"]).columns
    if column not in exclude_columns
]

for column in numeric_columns:
    series = prepared[column]
    if series.notna().any():
        stats["numeric_medians"][column] = float(series.median())
    else:
        stats["numeric_medians"][column] = 0.0
    q1 = float(series.quantile(0.25))
    q3 = float(series.quantile(0.75))
    iqr = q3 - q1

```

```

lower_bound = q1 - multiplier * iqr
upper_bound = q3 + multiplier * iqr
stats["outlier_bounds"][column] = (lower_bound, upper_bound)

for column in categorical_columns:
    mode_values = prepared[column].mode(dropna=True)
    replacement = mode_values.iloc[0] if not mode_values.empty else "unknown"
    stats["categorical_modes"][column] = replacement

return stats

```

```

def apply_preprocessing_with_statistics(dataframe: pd.DataFrame, preprocessing_stats: dict) ->
pd.DataFrame:

processed = process_sale_date(dataframe)

numeric_medians = preprocessing_stats.get("numeric_medians", {})
categorical_modes = preprocessing_stats.get("categorical_modes", {})
outlier_bounds = preprocessing_stats.get("outlier_bounds", {})

for column, median_value in numeric_medians.items():
    if column in processed.columns and pd.api.types.is_numeric_dtype(processed[column]):
        processed[column] = processed[column].fillna(median_value)

for column, mode_value in categorical_modes.items():
    if column in processed.columns:
        processed[column] = processed[column].fillna(mode_value)

for column, bounds in outlier_bounds.items():
    if column in processed.columns and pd.api.types.is_numeric_dtype(processed[column]):
        lower_bound, upper_bound = bounds
        processed[column] = processed[column].clip(lower=lower_bound, upper=upper_bound)

return processed

```

```
def fit_and_preprocess_train(dataframe: pd.DataFrame, exclude_columns: list[str] | None =
None) -> tuple[pd.DataFrame, dict]:

stats = compute_preprocessing_statistics(dataframe, exclude_columns=exclude_columns or [])

processed = apply_preprocessing_with_statistics(dataframe, stats)

return processed, stats
```

```
def cap_outliers(dataframe: pd.DataFrame, multiplier: float = 1.5) -> pd.DataFrame:

capped = dataframe.copy()

numeric_columns = capped.select_dtypes(include=[np.number]).columns

for column in numeric_columns:

q1 = capped[column].quantile(0.25)

q3 = capped[column].quantile(0.75)

iqr = q3 - q1

lower_bound = q1 - multiplier * iqr

upper_bound = q3 + multiplier * iqr

capped[column] = capped[column].clip(lower=lower_bound, upper=upper_bound)

return capped
```

```
def encode_categorical_features(dataframe: pd.DataFrame, exclude_columns: list[str] | None =
None) -> pd.DataFrame:

exclude_columns = exclude_columns or []

encoded = dataframe.copy()

datetime_columns = encoded.select_dtypes(include=["datetime64[ns]", "datetime64[ns,
UTC]").columns

encoded = encoded.drop(columns=list(datetime_columns), errors="ignore")

categorical_columns = [
```

```

column
for column in encoded.select_dtypes(include=["object", "category", "string"]).columns
if column not in exclude_columns
]
if categorical_columns:
encoded = pd.get_dummies(encoded, columns=categorical_columns, drop_first=False)
return encoded

```

```

def preprocess_dataset(dataframe: pd.DataFrame) -> pd.DataFrame:
processed = process_sale_date(dataframe)
if "district" in processed.columns:
processed["district"] = processed["district"].astype("string")
processed = fill_missing_values(processed)
processed = cap_outliers(processed)
return processed

```

feature_engineering.py

```

import numpy as np
import pandas as pd

```

```

def build_feature_statistics(
train_dataframe: pd.DataFrame,
target_column: str = "price",
) -> dict:
stats: dict[str, object] = {}

```

```

if "sale_date" in train_dataframe.columns and pd.api.types.is_datetime64_any_dtype(train_dataframe["sale_date"]):
    stats["reference_date"] = train_dataframe["sale_date"].min()
else:
    stats["reference_date"] = None

if "district" in train_dataframe.columns and target_column in train_dataframe.columns:
    district_means = train_dataframe.groupby("district")[target_column].mean().to_dict()
    stats["district_average_price"] = district_means
    stats["global_average_price"] = float(train_dataframe[target_column].mean())
else:
    stats["district_average_price"] = {}
    if target_column in train_dataframe.columns:
        stats["global_average_price"] = float(train_dataframe[target_column].mean())
    else:
        stats["global_average_price"] = np.nan

if "sale_date" in train_dataframe.columns and pd.api.types.is_datetime64_any_dtype(train_dataframe["sale_date"]):
    stats["train_date_min"] = train_dataframe["sale_date"].min()
    stats["train_date_max"] = train_dataframe["sale_date"].max()
else:
    stats["train_date_min"] = None
    stats["train_date_max"] = None

return stats

def create_time_and_market_features(
    dataframe: pd.DataFrame,
    feature_statistics: dict | None = None,
) -> pd.DataFrame:

```

```

features = dataframe.copy()

feature_statistics = feature_statistics or {}

reference_date = feature_statistics.get("reference_date")

if "sale_date" in features.columns and
pd.api.types.is_datetime64_any_dtype(features["sale_date"]):

    features["sale_year"] = pd.to_numeric(features["sale_date"].dt.year, errors="coerce")

    features["sale_month"] = features["sale_date"].dt.month

    features["sale_quarter"] = features["sale_date"].dt.quarter

    features["sale_day"] = features["sale_date"].dt.day

    if pd.notna(reference_date):

        features["trend_index"] = (features["sale_date"] - reference_date).dt.days.astype(float)

    else:

        features["trend_index"] = np.arange(len(features), dtype=float)

    else:

        features["sale_year"] = 0

        features["sale_month"] = 0

        features["sale_quarter"] = 0

        features["sale_day"] = 0

        features["trend_index"] = np.arange(len(features), dtype=float)

        features["month_sin"] = np.sin(2 * np.pi * features["sale_month"] / 12.0)

        features["month_cos"] = np.cos(2 * np.pi * features["sale_month"] / 12.0)

district_price_map = feature_statistics.get("district_average_price", {})

global_average_price = feature_statistics.get("global_average_price", np.nan)

if "district" in features.columns:

    features["district_avg_price"] =
    features["district"].map(district_price_map).fillna(global_average_price)

```

```

else:
    features["district_avg_price"] = global_average_price

    build_year_column = None

    if "year_built" in features.columns:
        build_year_column = "year_built"
    elif "yr_built" in features.columns:
        build_year_column = "yr_built"

    if build_year_column is not None:
        build_year_values = pd.to_numeric(features[build_year_column], errors="coerce")
        sale_year_values = pd.to_numeric(features["sale_year"], errors="coerce")
        features["yr_built"] = build_year_values
        features["year_built"] = build_year_values
        building_age = sale_year_values - build_year_values
        features["house_age"] = building_age.where(building_age >= 0, 0)
    else:
        features["house_age"] = np.nan
        features["year_built"] = np.nan
        features["yr_built"] = np.nan

    features["house_age_squared"] = features["house_age"] ** 2
    age_conditions = [
        features["house_age"] >= 40,
        (features["house_age"] >= 15) & (features["house_age"] < 40),
        features["house_age"] < 15,
    ]
    age_labels = ["old_house", "middle_house", "new_house"]

```

```

features["house_age_category"] = np.select(age_conditions, age_labels,
default="middle_house")

if "yr_renovated" in features.columns:
    renovated_year = pd.to_numeric(features["yr_renovated"], errors="coerce").fillna(0)
    features["is_renovated"] = (renovated_year > 0).astype(int)
    years_after_renovation = features["sale_year"] - renovated_year
    years_after_renovation = years_after_renovation.where(features["is_renovated"] == 1,
    features["house_age"])
    features["years_after_renovation"] = years_after_renovation.where(years_after_renovation >= 0,
    0)
else:
    features["is_renovated"] = 0
    features["years_after_renovation"] = features["house_age"]

if "living_area" in features.columns:
    features["living_area_log"] = np.log1p(pd.to_numeric(features["living_area"], errors="coerce"))
    features["area_time_interaction"] = features["living_area"] * features["trend_index"]
else:
    features["living_area_log"] = 0.0
    features["area_time_interaction"] = 0.0

features["age_time_interaction"] = features["house_age"] * features["trend_index"]

numeric_columns = features.select_dtypes(include=[np.number]).columns
for column in numeric_columns:
    median_value = features[column].median() if features[column].notna().any() else 0.0
    features[column] = features[column].fillna(median_value)
return features

```

models.py

```

from pathlib import Path

import joblib

import numpy as np

import pandas as pd

from sklearn.linear_model import LinearRegression

from xgboost import XGBRegressor

from catboost import CatBoostRegressor

from lightgbm import LGBMRegressor

from sklearn.preprocessing import MinMaxScaler


from data_loader import standardize_king_county_columns

from preprocessing import apply_preprocessing_with_statistics, encode_categorical_features

from feature_engineering import create_time_and_market_features


def _validate_features_for_training(x_train: pd.DataFrame, y_train: pd.Series) -> None:
    if x_train.isna().any().any():
        raise ValueError("Y train-ознаках є NaN")

    numeric_values = x_train.select_dtypes(include=[np.number]).values

    if np.isinf(numeric_values).any():
        raise ValueError("Y train-ознаках є inf")

    if y_train.isna().any() or np.isinf(y_train.values).any():
        raise ValueError("Y train-target є NaN або inf")


def train_linear_regression(x_train: pd.DataFrame, y_train: pd.Series) -> LinearRegression:

```

```

model = LinearRegression()
model.fit(x_train, y_train)
return model

```

```

def train_xgboost(x_train: pd.DataFrame, y_train: pd.Series) -> XGBRegressor:
    model = XGBRegressor(
        objective="reg:squarederror",
        n_estimators=400,
        learning_rate=0.05,
        max_depth=6,
        subsample=0.9,
        colsample_bytree=0.9,
        random_state=42,
        verbosity=0,
    )
    model.fit(x_train, y_train)
    return model

```

```

def train_catboost(x_train: pd.DataFrame, y_train: pd.Series) -> CatBoostRegressor:
    model = CatBoostRegressor(
        iterations=400,
        learning_rate=0.05,
        depth=6,
        verbose=0,
        random_state=42,
    )

```

```
model.fit(x_train, y_train, verbose=False)

return model
```

```
def train_lightgbm(x_train: pd.DataFrame, y_train: pd.Series) -> LGBMRegressor:
    model = LGBMRegressor(
        n_estimators=400,
        learning_rate=0.05,
        max_depth=6,
        num_leaves=31,
        random_state=42,
        verbose=-1,
    )
    model.fit(x_train, y_train)
    return model
```

```
def train_arima_model_wrapper(dataframe: pd.DataFrame) -> "Optional[object]":
    """
    Тренує ARIMA модель для ринкового коригування.
    Це додатковий інструмент для аналізу часових трендів ціни.
    """
    try:
        from arima_market_forecast import train_arima_model, prepare_price_timeseries
        if "sale_date" not in dataframe.columns or "price" not in dataframe.columns:
            return None
        ts_df = prepare_price_timeseries(dataframe, aggregation_period="MS", min_periods=6)
        if len(ts_df) < 4:
```

```

return None

result = train_arima_model(
    ts_df["price"].values,
    auto_parameters=True,
    models_dir="models",
    model_name="arima_price_model"
)

if "error" in result or "model" not in result:
    return None

return result.get("model")

except ImportError:
    return None

except Exception as e:
    import logging
    logging.getLogger(__name__).debug(f"ARIMA тренування пропущено: {str(e)[:50]}")
    return None

```

```

def train_all_models(x_train: pd.DataFrame, y_train: pd.Series) -> dict:
    _validate_features_for_training(x_train, y_train)

    models = {
        "Linear Regression": train_linear_regression(x_train, y_train),
        "XGBoost": train_xgboost(x_train, y_train),
        "CatBoost": train_catboost(x_train, y_train),
        "LightGBM": train_lightgbm(x_train, y_train),
    }

    return models

```

```
def predict_with_models(trained_models: dict, x_test: pd.DataFrame) -> dict:
```

```
    predictions = {}
```

```
    for model_name, model in trained_models.items():
```

```
        predictions[model_name] = model.predict(x_test)
```

```
    return predictions
```

```
def save_model_payload(payload: dict, model_file: str) -> None:
```

```
    model_file = Path(model_file)
```

```
    model_file.parent.mkdir(parents=True, exist_ok=True)
```

```
    joblib.dump(payload, model_file)
```

```
def load_model_payload(model_path: str) -> dict:
```

```
    return joblib.load(model_path)
```

```
def define_price_status(predicted_price: float, average_market_price: float, tolerance: float = 0.15) -> str:
```

```
    upper_bound = average_market_price * (1 + tolerance)
```

```
    lower_bound = average_market_price * (1 - tolerance)
```

```
    if predicted_price > upper_bound:
```

```
        return "завищена"
```

```
    if predicted_price < lower_bound:
```

```
        return "занижена"
```

```
    return "ринкова"
```

```

def prepare_input_features(property_data: dict, model_payload: dict) -> pd.DataFrame:
    input_frame = pd.DataFrame([property_data])
    input_frame = standardize_king_county_columns(input_frame)
    preprocessing_stats = model_payload.get("preprocessing_stats", {})
    input_frame = apply_preprocessing_with_statistics(input_frame, preprocessing_stats)
    input_frame = create_time_and_market_features(
        input_frame,
        feature_statistics=model_payload.get("feature_statistics", {}),
    )
    input_frame = encode_categorical_features(input_frame)
    aligned = input_frame.reindex(columns=model_payload["feature_columns"], fill_value=0.0)
    feature_medians = model_payload.get("feature_medians", {})
    for column, median_value in feature_medians.items():
        if column in aligned.columns:
            aligned[column] = aligned[column].fillna(median_value)
    return aligned

def predict_property_price(model_payload: dict, property_data: dict, dataframe: "pd.DataFrame | None" = None) -> dict:
    model = model_payload["model"]
    input_features = prepare_input_features(property_data, model_payload)
    base_price = float(model.predict(input_features)[0])

    market_adjustment = {
        "market_adjustment_factor": 1.0,
        "market_adjustment_source": "disabled",
    }

```

```

"market_prediction_price": None,
"market_baseline_price": None,
}

try:

from lstm_market_adjustment import calculate_market_adjustment

market_adjustment = calculate_market_adjustment(property_data.get("sale_date"))

except Exception as error:

# Спробуємо ARIMA як резервний варіант

if dataframe is not None:

try:

from arima_market_forecast import calculate_arima_market_adjustment

arima_adjustment = calculate_arima_market_adjustment(dataframe, steps_ahead=1)

if "adjustment_factor" in arima_adjustment and arima_adjustment.get("source") !=
"insufficient_data":

market_adjustment = {

"market_adjustment_factor": arima_adjustment.get("adjustment_factor", 1.0),
"market_adjustment_source": arima_adjustment.get("source", "arima_error"),
"market_prediction_price": arima_adjustment.get("forecasted_price"),
"market_baseline_price": arima_adjustment.get("current_price"),
}

else:

market_adjustment = {

"market_adjustment_factor": 1.0,
"market_adjustment_source": f"lstm_error_fallback: {error}",
"market_prediction_price": None,
"market_baseline_price": None,
}

```

```

except Exception as arima_error:
    market_adjustment = {
        "market_adjustment_factor": 1.0,
        "market_adjustment_source": f"lstm_error: {error}",
        "market_prediction_price": None,
        "market_baseline_price": None,
    }
else:
    market_adjustment = {
        "market_adjustment_factor": 1.0,
        "market_adjustment_source": f"lstm_error: {error}",
        "market_prediction_price": None,
        "market_baseline_price": None,
    }

predicted_price = float(base_price * market_adjustment["market_adjustment_factor"])
average_price = float(model_payload.get("global_average_price", predicted_price))
status = define_price_status(predicted_price, average_price)
warning_message = None
raw_sale_date = property_data.get("sale_date")
input_sale_date = pd.to_datetime(raw_sale_date, errors="coerce")
train_date_min = pd.to_datetime(model_payload.get("train_date_min"), errors="coerce")
train_date_max = pd.to_datetime(model_payload.get("train_date_max"), errors="coerce")
if pd.notna(input_sale_date) and pd.notna(train_date_min) and pd.notna(train_date_max):
    distance_days = 0
    if input_sale_date < train_date_min:
        distance_days = (train_date_min - input_sale_date).days
    elif input_sale_date > train_date_max:

```

```

distance_days = (input_sale_date - train_date_max).days

if distance_days > 365:

    warning_message = f"Дата продажу віддалена від періоду навчання на
    {distance_days} днів. Прогноз може бути менш надійним."

    return {

        "base_price": base_price,

        "predicted_price": predicted_price,

        "price_status": status,

        "reliability_warning": warning_message,

        "market_adjustment_factor": market_adjustment["market_adjustment_factor"],

        "market_adjustment_source": market_adjustment["market_adjustment_source"],

        "market_prediction_price": market_adjustment.get("market_prediction_price"),

        "market_baseline_price": market_adjustment.get("market_baseline_price"),

    }

```

```

def prepare_lstm_time_series(dataframe: pd.DataFrame, aggregation_period: str = "ME",
max_periods: int = 240) -> pd.DataFrame:

    df = dataframe.copy()

    if "sale_date" not in df.columns:

        raise ValueError("sale_date не знайдена")

    if "price" not in df.columns:

        raise ValueError("price не знайдена")

    df["sale_date"] = pd.to_datetime(df["sale_date"], errors="coerce")

    df = df.dropna(subset=["sale_date", "price"])

    df_sorted = df.sort_values("sale_date")

    agg_dict = {

        "price": ["mean", "count"],

    }

```

```

if "living_area" in df_sorted.columns:
    agg_dict["living_area"] = "mean"
elif "sqft_living" in df_sorted.columns:
    agg_dict["sqft_living"] = "mean"

time_series =
df_sorted.set_index("sale_date").resample(aggregation_period).agg(agg_dict).reset_index()

time_series.columns = ["date", "average_price", "sales_count", "avg_living_area"]

time_series = time_series.dropna(subset=["average_price"])

if len(time_series) > max_periods:
    time_series = time_series.tail(max_periods).reset_index(drop=True)

return time_series

```

```

def create_lstm_sequences(data: np.ndarray, lookback: int = 6) -> tuple:
    X, y = [], []
    for i in range(len(data) - lookback):
        X.append(data[i:i + lookback])
        y.append(data[i + lookback])
    return np.array(X), np.array(y)

```

===== LSTM INFERENCE ONLY (без тренування) =====

Для тренування LSTM див. colab_train_lstm.py або окремий training скрипт

Модель завантажується з папки models/ через model_inference.py

```

def predict_lstm_model(model, X_test: np.ndarray) -> np.ndarray:

```

```

    """Прогноз натренованою моделлю (інференс тільки)."""

```

```

if X_test.ndim == 2:
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
return model.predict(X_test, verbose=0)

def evaluate_lstm_model(y_true: np.ndarray, y_pred: np.ndarray) -> dict:
    """Оцінка якості прогнозу."""
    from evaluation import compute_all_metrics
    return compute_all_metrics(y_true, y_pred, y_train=y_true)

```

elavation.py

```

import numpy as np
import pandas as pd

def safe_divide(numerator, denominator, default=np.nan):
    with np.errstate(divide='ignore', invalid='ignore'):
        result = np.divide(numerator, denominator)
        result[~np.isfinite(result)] = default
    return result

def compute_mse(y_true, y_pred):
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
    if not mask.any():

```

```

return np.nan

return float(np.mean(np.square(y_true[mask] - y_pred[mask])))

```

```

def compute_mae(y_true, y_pred):
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
    if not mask.any():
        return np.nan
    return float(np.mean(np.abs(y_true[mask] - y_pred[mask])))

```

```

def compute_rmse(y_true, y_pred):
    mse = compute_mse(y_true, y_pred)
    if np.isnan(mse):
        return np.nan
    return float(np.sqrt(mse))

```

```

def compute_smape(y_true, y_pred):
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
    if not mask.any():
        return np.nan
    numerator = np.abs(y_true[mask] - y_pred[mask])
    denominator = (np.abs(y_true[mask]) + np.abs(y_pred[mask])) / 2.0

```

```
with np.errstate(divide='ignore', invalid='ignore'):
```

```
ratios = numerator / denominator
```

```
ratios[~np.isfinite(ratios)] = 0.0
```

```
return float(200.0 * np.mean(ratios))
```

```
def compute_mape(y_true, y_pred):
```

```
y_true = np.asarray(y_true, dtype=float)
```

```
y_pred = np.asarray(y_pred, dtype=float)
```

```
mask = np.isfinite(y_true) & np.isfinite(y_pred) & (y_true != 0)
```

```
if not mask.any():
```

```
return np.nan
```

```
return float(100.0 * np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])))
```

```
def compute_mase(y_true, y_pred, y_train=None):
```

```
y_true = np.asarray(y_true, dtype=float)
```

```
y_pred = np.asarray(y_pred, dtype=float)
```

```
mask = np.isfinite(y_true) & np.isfinite(y_pred)
```

```
if not mask.any():
```

```
return np.nan
```

```
mae_pred = np.mean(np.abs(y_true[mask] - y_pred[mask]))
```

```
if y_train is not None:
```

```
y_train = np.asarray(y_train, dtype=float)
```

```
naive_errors = np.abs(np.diff(y_train))
```

```
if len(naive_errors) > 0:
```

```
mae_naive = np.mean(naive_errors)
```

```
if mae_naive > 0:
```

```
return float(mae_pred / mae_naive)
```

```
return np.nan
```

```
def compute_mspe(y_true, y_pred):
```

```
    y_true = np.asarray(y_true, dtype=float)
```

```
    y_pred = np.asarray(y_pred, dtype=float)
```

```
    mask = np.isfinite(y_true) & np.isfinite(y_pred) & (y_true != 0)
```

```
    if not mask.any():
```

```
        return np.nan
```

```
    return float(100.0 * np.mean(np.square((y_true[mask] - y_pred[mask]) / y_true[mask])))
```

```
def compute_rms(y_true, y_pred):
```

```
    y_true = np.asarray(y_true, dtype=float)
```

```
    y_pred = np.asarray(y_pred, dtype=float)
```

```
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
```

```
    if not mask.any():
```

```
        return np.nan
```

```
    return float(np.sqrt(np.mean(np.square(y_pred[mask]))))
```

```
def compute_r_squared(y_true, y_pred):
```

```
    y_true = np.asarray(y_true, dtype=float)
```

```
    y_pred = np.asarray(y_pred, dtype=float)
```

```
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
```

```
    if not mask.any():
```

```
        return np.nan
```

```

ss_res = np.sum(np.square(y_true[mask] - y_pred[mask]))
ss_tot = np.sum(np.square(y_true[mask] - np.mean(y_true[mask])))
if ss_tot == 0:
    return np.nan
return float(1.0 - (ss_res / ss_tot))

```

```

def compute_mda(y_true, y_pred):
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
    if not mask.any() or len(y_true[mask]) < 2:
        return np.nan
    y_true_clean = y_true[mask]
    y_pred_clean = y_pred[mask]
    true_dir = np.sign(np.diff(y_true_clean))
    pred_dir = np.sign(np.diff(y_pred_clean))
    if len(true_dir) == 0:
        return np.nan
    correct = np.sum(true_dir == pred_dir)
    return float(100.0 * correct / len(true_dir))

```

```

def compute_mad(y_true, y_pred):
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    mask = np.isfinite(y_true) & np.isfinite(y_pred)
    if not mask.any():

```

```

return np.nan

errors = y_true[mask] - y_pred[mask]

mean_error = np.mean(errors)

return float(np.mean(np.abs(errors - mean_error)))

```

```

def compute_all_metrics(y_true, y_pred, y_train=None):
    rmse = compute_rmse(y_true, y_pred)
    r_squared = compute_r_squared(y_true, y_pred)
    metrics = {
        "MSE": compute_mse(y_true, y_pred),
        "MAE": compute_mae(y_true, y_pred),
        "sMAPE": compute_smape(y_true, y_pred),
        "MAPE": compute_mape(y_true, y_pred),
        "MASE": compute_mase(y_true, y_pred, y_train=y_train),
        "MSPE": compute_mspe(y_true, y_pred),
        "RMS": compute_rms(y_true, y_pred),
        "RMSE": rmse,
        "RMSD": rmse,
        "R2": r_squared,
        "R^2": r_squared,
        "MDA": compute_mda(y_true, y_pred),
        "MAD": compute_mad(y_true, y_pred),
    }
    return metrics

```

```

def evaluate_models(y_true, predictions_dict):

```

```

results_list = []

for model_name, y_pred in predictions_dict.items():

    metrics = compute_all_metrics(y_true, y_pred)

    row = {"Model": model_name, **metrics}

    results_list.append(row)

results_df = pd.DataFrame(results_list)

return results_df

```

```

def select_best_model(metrics_table):

    metrics_table_clean = metrics_table.copy()

    valid_rmse = metrics_table_clean[metrics_table_clean["RMSE"].notna() &
np.isfinite(metrics_table_clean["RMSE"])]

    if valid_rmse.empty:

        return metrics_table_clean.iloc[0]["Model"] if len(metrics_table_clean) > 0 else "Unknown"

    best_idx = valid_rmse["RMSE"].idxmin()

    return metrics_table_clean.loc[best_idx, "Model"]

```

```

def save_metrics(metrics_table, output_path):

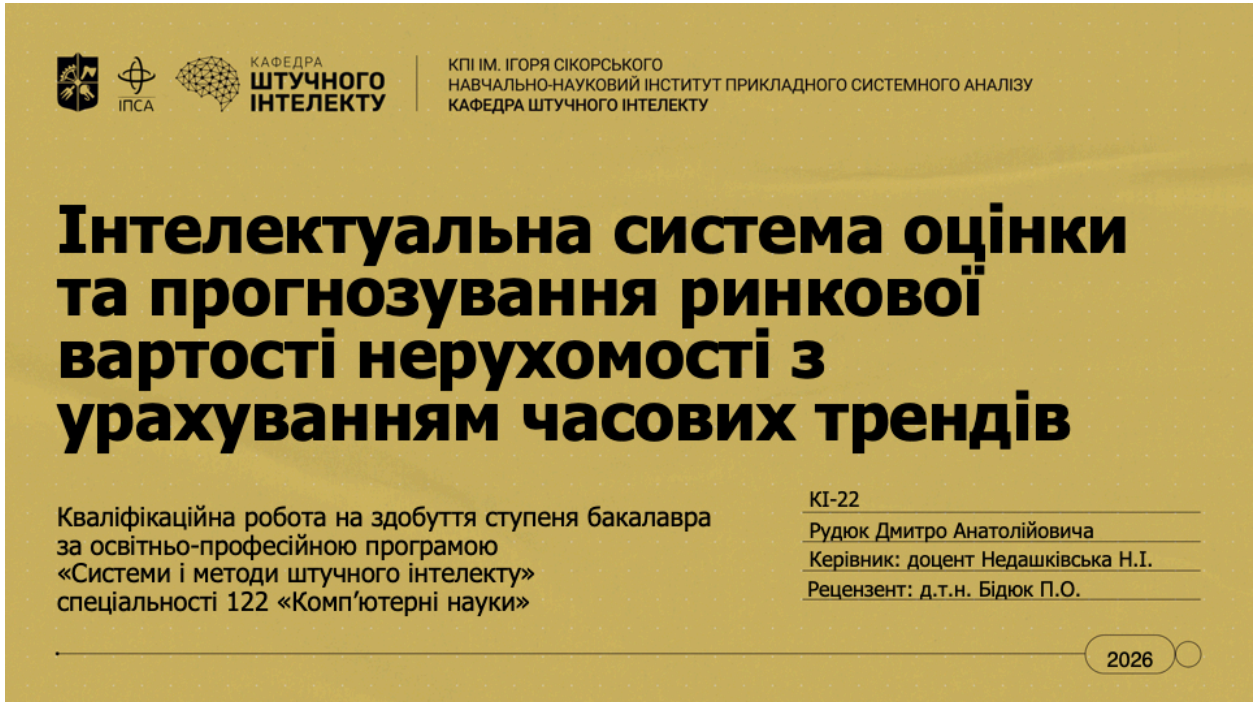
    from pathlib import Path

    Path(output_path).parent.mkdir(parents=True, exist_ok=True)

    metrics_table.to_csv(output_path, index=False)

```

ДОДАТОК Б ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ






 КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ

КПІ ІМ. ІГОРЯ СІКОРСЬКОГО
 НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
 КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

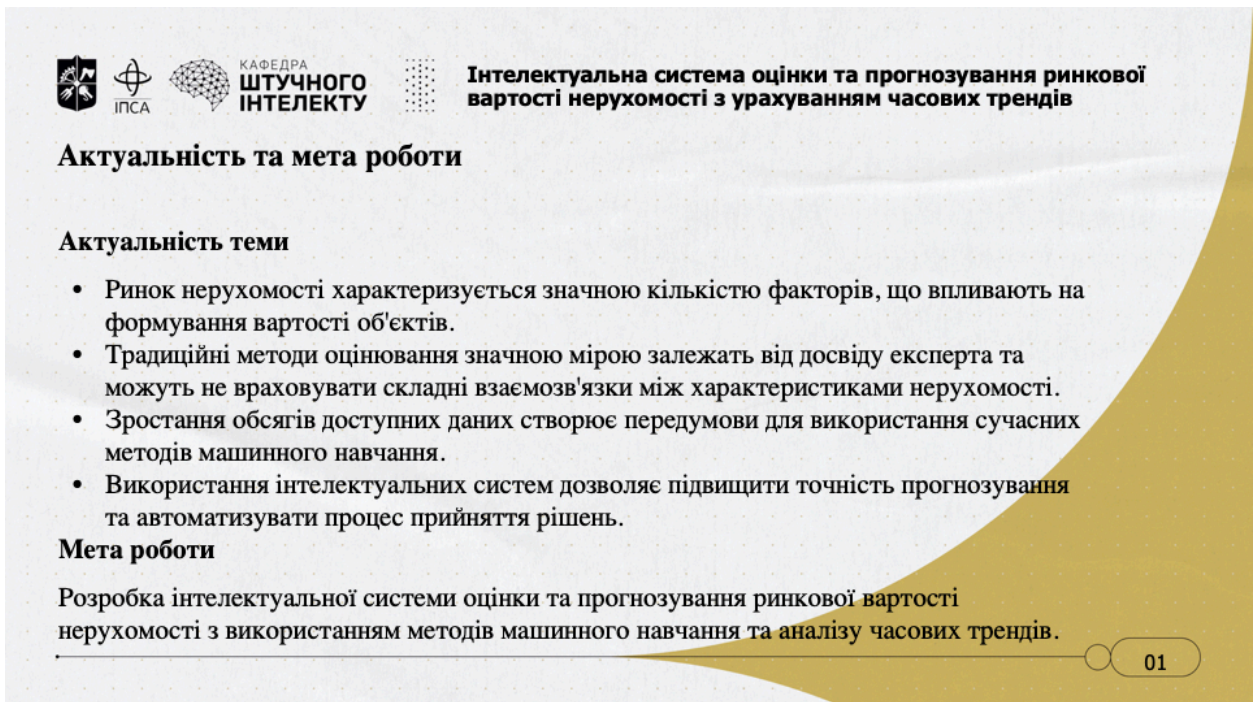
Інтелектуальна система оцінки та прогнозування ринкової вартості нерухомості з урахуванням часових трендів

Кваліфікаційна робота на здобуття ступеня бакалавра
 за освітньо-професійною програмою
 «Системи і методи штучного інтелекту»
 спеціальності 122 «Комп'ютерні науки»

КІ-22
 Рудюк Дмитро Анатолійович
 Керівник: доцент Недашківська Н.І.
 Рецензент: д.т.н. Бідюк П.О.

2026

Рисунок Б.1 – Титульний слайд






 КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ

Інтелектуальна система оцінки та прогнозування ринкової вартості нерухомості з урахуванням часових трендів

Актуальність та мета роботи

Актуальність теми

- Ринок нерухомості характеризується значною кількістю факторів, що впливають на формування вартості об'єктів.
- Традиційні методи оцінювання значною мірою залежать від досвіду експерта та можуть не враховувати складні взаємозв'язки між характеристиками нерухомості.
- Зростання обсягів доступних даних створює передумови для використання сучасних методів машинного навчання.
- Використання інтелектуальних систем дозволяє підвищити точність прогнозування та автоматизувати процес прийняття рішень.

Мета роботи

Розробка інтелектуальної системи оцінки та прогнозування ринкової вартості нерухомості з використанням методів машинного навчання та аналізу часових трендів.

01

Рисунок Б.2 – Слайд 1

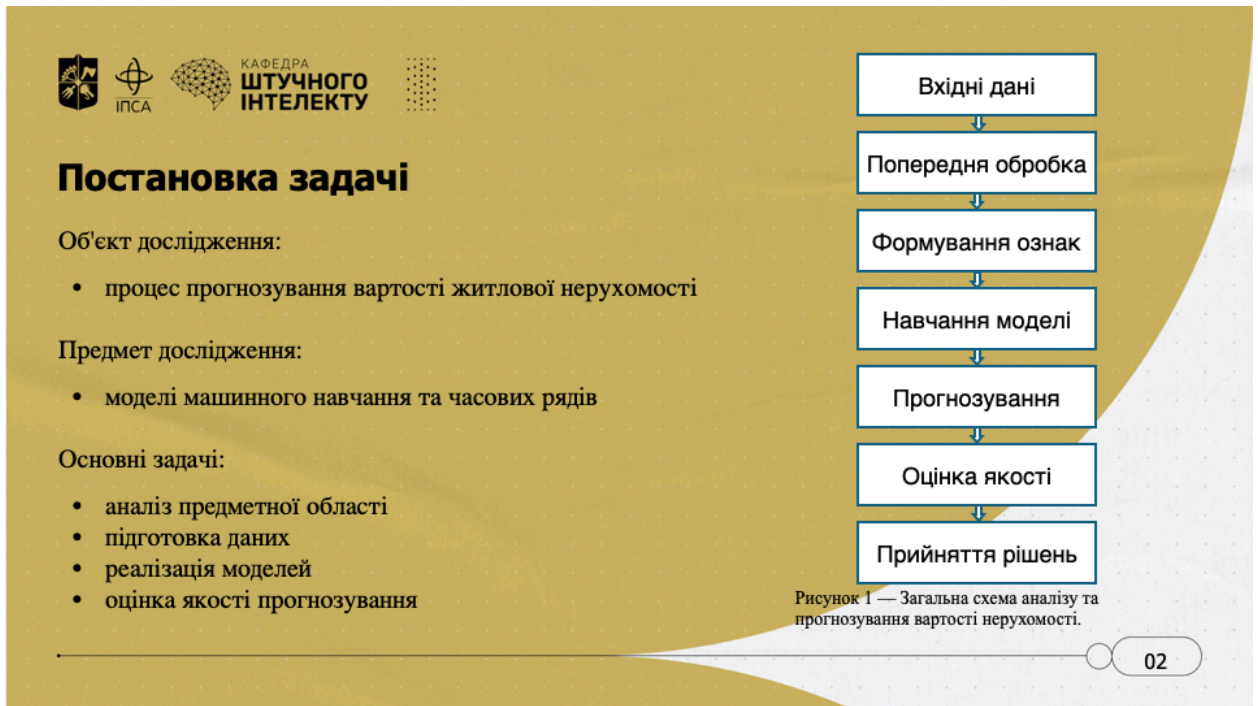


Рисунок Б.3 – Слайд 2

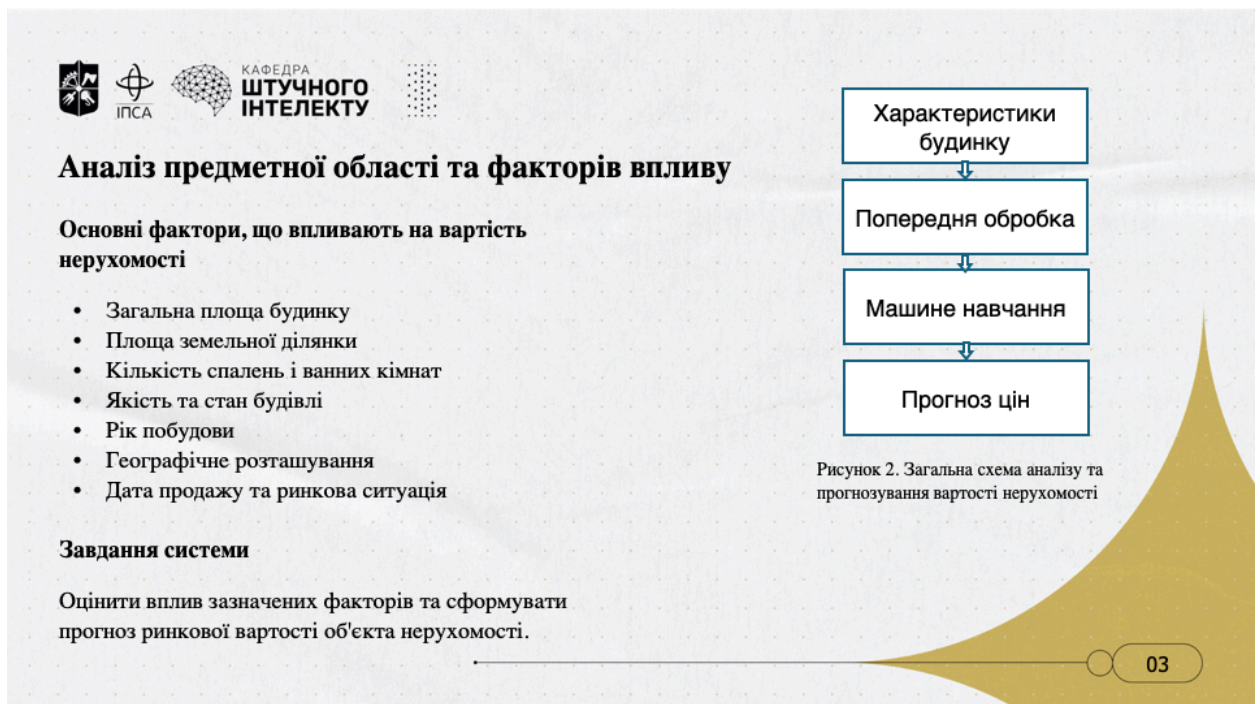


Рисунок Б.4 – Слайд 3

Вхідні дані та їх підготовка

Набір даних King County:

- Ціна продажу
- Площа будинку
- Площа ділянки
- Кількість кімнат
- Ванні кімнати
- Поверховість
- Якість будівництва
- Географічні координати
- Рік побудови
- Дата продажу

Виконані етапи:

- очищення даних
- обробка пропусків
- кодування ознак
- нормалізація
- створення часових ознак
- поділ на train/test

04

Рисунок Б.5 – Слайд 4

Використані моделі

Модель	Призначення
Linear Regression	Базова модель
XGBoost	Градiєнтний бустинг
CatBoost	Категорiальнi данi
LightGBM	Швидке навчання
LSTM	Часовi тренди

05

Рисунок Б.6 – Слайд 5



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ



Архітектура системи



Рисунок 3 — Архітектура системи

06

Рисунок Б.7 – Слайд 6



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ



Результати порівняння моделей

Model	MSE	MAE	sMAPE	MAPE	MASE	MSPE	RMS	RMSE	RMSD	R2	+ R ²	MDA	MAD
Linear Regression	35952962943.4457	108690.0794	39.7823	20.4726	None	8.0307	635228.7183	189612.6656	189612.6656	0.7343	0.7343	86.187	108710.485
CatBoost	20817537407.5125	77813.3004	27.1363	13.55	None	3.726	616006.082	144282.8382	144282.8382	0.8461	0.8461	89.6344	77632.3598
LightGBM	19751246965.2571	75091.6847	26.1412	13.2037	None	3.7259	627306.0474	140539.1297	140539.1297	0.854	0.854	90.3054	75258.6667
XGBoost	18628361527.4192	73083.8212	25.4356	12.9243	None	3.6261	629663.2225	136485.7558	136485.7558	0.8623	0.8623	89.9352	72959.4962

Рисунок 4 — Таблиця з результатами

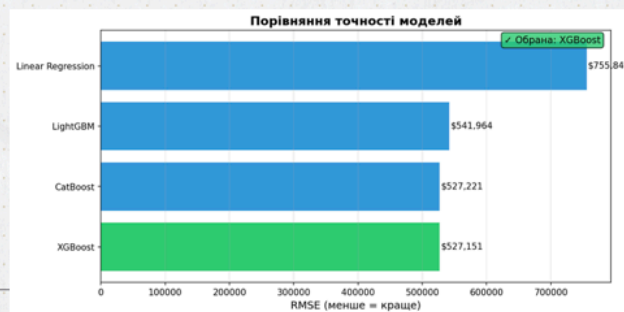


Рисунок 5 — Порівняння точності моделей

07

Рисунок Б.8 – Слайд 7



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ



Аналіз важливості ознак

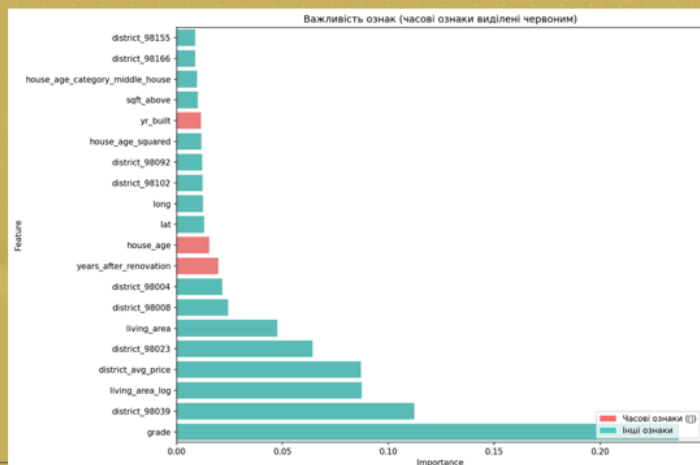


Рисунок 6 — Аналіз важливості ознак

08

Рисунок Б.9 – Слайд 8



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ



Прогноз часових трендів (LSTM)

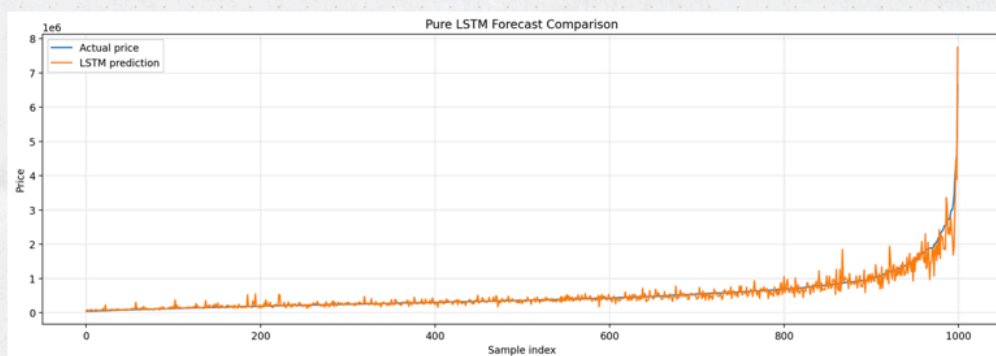


Рисунок 7 — Фактичні значення vs Прогноз LSTM

09

Рисунок Б.10 – Слайд 9

Висновки

Отже, у результаті роботи розроблено інтелектуальну систему оцінки та прогнозування вартості нерухомості з використанням методів машинного навчання. Проведене дослідження підтвердило ефективність моделей градієнтного бустингу та можливість врахування часових трендів для підвищення точності прогнозів. Розроблена система має практичну цінність і може бути використана для підтримки прийняття рішень на ринку нерухомості.

Рисунок Б.11 – Слайд 10