Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ВИЗНАЧЕННЯ ВАРТОСТІ КОМЕРЦІЙНИХ ТА ЖИТЛОВИХ ПРИМІЩЕНЬ НА РИНКУ НЕРУХОМОСТІ

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Анастасія ПРИМАК
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: _____
д.т.н., проф.В'ячеслав КОВТУН
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Андрій ЯРОВИЙ
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Зав. каф. КН, д.т.н., проф.
_____ Андрій ЯРОВИЙ
«06» травня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Примак Анастасії Вікторівні

1. Тема роботи: Програмний модуль визначення вартості комерційних та житлових приміщень на ринку нерухомості.
керівник роботи: Іванчук Ярослав Володимирович, д.т.н, професор кафедри КН
затверджені наказом вищого навчального закладу "20" березня 2025 року № 97

2. Строк подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи: набір даних ціноутворення нерухомості в Україні, що включає числові показники ціни [15000…300 000] USD, площі [30…99] м², року побудови [1950…2024], кількості кімнат [1…6], віддаленості від центру [0.5…19.9] км; тип будівлі, наявність інфраструктури. кількість сторінок інтерфейсу користувача – не менше 3 од.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): визначення об'єкту, предмета, мети та задач подальшого дослідження аналіз предметноїобласті, існуючихметодів, алгоритмів та інструментів для створення програмного модуля для визначення вартості комерційних та житлових приміщень на ринку нерухомості, проєктування програмного модуля визначення вартості комерційних та житлових приміщень на ринку нерухомості, програмна реалізація визначення вартості комерційних та житлових приміщень на ринку нерухомості.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму роботи взаємодії розроблених модулів, загальний вигляд інтерфейсу користувача для програмного модуля для прогнозування вартості нерухомості.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Іванчук Я.В., проф. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітки
		початок	закінчення	
1	Визначення об'єкта, предмета, мети та задачі подальшого дослідження	05.05.25	10.05.25	
2	Аналіз предметної області в сфері оцінки вартості нерухомості	11.05.25	18.05.25	
3	Проєктування програмних програмного модуля для оцінки вартості нерухомості	19.05.25	26.05.25	
4	Програмна реалізація програмного модуля для оцінки вартості нерухомості	27.05.25	31.05.25	
5	Оформлення пояснювальної записки	01.06.25	04.06.25	

Студент _____ Анастасія ПРИМАК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ярослав ІВАНЧУК
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 63 сторінки формату А4, на яких є 13 рисунків, список використаних джерел містить 28 найменувань.

Дана бакалаврська робота присвячена розробці та впровадженню програмного модуля для визначення вартості комерційних та житлових приміщень на ринку нерухомості. У роботі розглянуто проблему визначення ринкової вартості житлової та комерційної нерухомості в умовах зростаючого попиту та високої динаміки цін. Актуальність теми зумовлена необхідністю забезпечення більш точного, об'єктивного та швидкого способу оцінювання об'єктів нерухомості, особливо в умовах нестабільності ринку. Запропоновано програмний модуль, що використовує алгоритми машинного навчання, зокрема Random Forest, для прогнозування ціни нерухомості на основі ключових характеристик об'єкта та ринкових факторів. Архітектура системи реалізована у вигляді модульної структури, яка включає блоки обробки даних, прогнозування, виявлення аномалій, візуалізації результатів та взаємодії з користувачем.

У модулі також використано алгоритм Isolation Forest для фільтрації нетипових об'єктів. Реалізація здійснена за допомогою мови Python з використанням бібліотек Pandas, Numpy, Sklearn, Tkinter та Matplotlib. Проведене тестування на синтетичних та реальних наборах даних показало високу точність та надійність прогнозів. Отримані результати свідчать про ефективність запропонованого рішення в задачах автоматизованого оцінювання вартості нерухомості. Розроблене програмне забезпечення може бути використане інвесторами, забудовниками, ріелторами та іншими учасниками ринку для прийняття обґрунтованих рішень щодо ціноутворення й доцільності інвестицій.

Ключові слова: оцінка нерухомості, машинне навчання, Random Forest, програмний модуль, прогнозування цін, автоматизація.

ABSTRACT

The Bachelor's thesis consists of 63 pages of A4 format, on which there are 13 figures, the list of used sources contains 28 sources.

This Bachelor's thesis is devoted to the development and implementation of a software module for determining the value of commercial and residential premises in the real estate market. The work considers the problem of determining the market value of residential and commercial real estate in conditions of growing demand and high price dynamics. The relevance of the topic is due to the need to provide a more accurate, objective and fast method of evaluating real estate, especially in conditions of market instability. A software module is proposed that uses machine learning algorithms, in particular Random Forest, to predict the price of real estate based on key characteristics of the object and market factors. The system architecture is implemented in the form of a modular structure, which includes blocks for data processing, forecasting, anomaly detection, visualization of results and user interaction.

The module also uses the Isolation Forest algorithm for filtering atypical objects. The implementation was carried out using the Python language using the Pandas, Numpy, Sklearn, Tkinter and Matplotlib libraries. Testing on synthetic and real datasets showed high accuracy and reliability of forecasts. The results obtained indicate the effectiveness of the proposed solution in the tasks of automated real estate valuation. The developed software can be used by investors, developers, realtors and other market participants to make informed decisions on pricing and investment feasibility.

Keywords: real estate valuation, machine learning, Random Forest, software module, price forecasting, automation.

ЗМІСТ

ВСТУП

Актуальність дослідження. Ринок нерухомості є однією з найпопулярніших і найбільш потрібних сфер людської діяльності, особливо враховуючи стрімке зростання попиту на багатофункціональну інфраструктуру в кожному районі міста. Орендарі, продавці та покупці приміщень все більше звертають увагу на розміщення будівлі відносно транспортних сполучень, центральної інфраструктури міста, наявність парковок тощо. Зі зростанням кількості населення, та відповідно кількості об’єктів нерухомості, постає питання наявності більш автоматизованих методів для оцінки їх вартості. Звичні та часто використовувані методи оцінювання засновані переважно на експертних оцінках і статистичних підходах, що частково робить таку оцінку суб’єктивною та відповідно неточною. Для вирішення цього питання впроваджуються автоматизовані програмні системи, які враховують велику кількість параметрів, швидко аналізують ринкові тенденції та забезпечують більш об’єктивні результати [1].

Основною проблемою, яку покликаний вирішити розроблений програмний модуль, є суб’єктивність, складність і тривалість процесу оцінки вартості нерухомості, що часто базується на неточних або застарілих даних. Модуль дозволить автоматизувати розрахунок ціни з урахуванням актуальних ринкових факторів, підвищуючи точність, швидкість і прозорість оцінки.

Особливу актуальність такі модулі мають в умовах нестабільного ринку, де зміни в економічній ситуації, інфраструктурному розвитку міст, законодавстві чи валютному курсі можуть суттєво впливати на ринкову вартість об’єктів. Система є корисною для покупців, які зможуть зручно та швидко отримати ринкову ціну об’єкиа, для інвесторів, щоб оцінити доцільність вкладень, для забудовників, щоб аналізувати ринкові тренди та динамість цін. Завдяки автоматизації та прозорості розрахунків система сприятиме підвищенню зручності процесу оцінки та пришвидшенню ухвалення рішень.

Метою дослідження є підвищення точності оцінки ринкової вартості житлових і комерційних об'єктів нерухомості шляхом впровадження програмного модуля на основі алгоритмів машинного навчання.

Предмет дослідження – програмний модуль для визначення вартості нерухомості, його функціональні можливості, архітектура та інтеграція з користувацьким інтерфейсом.

Об'єктом дослідження є процес прогнозування ринкової вартості житлових і комерційних об’єктів нерухомості за допомогою засобів програмного забезпечення.

Задачі дослідження:

- провести аналіз предметної області з визначення вартості комерційних та житлових приміщень на ринку нерухомості;
- реалізувати архітектуру програмного модуля оцінки вартості нерухомості із застосуванням сучасних технологій обробки даних і машинного навчання;
- розробити інтерфейс користувача для введення характеристик об'єктів нерухомості та отримання результатів оцінки;
- провести тестування модуля на реальних або синтетичних наборах даних для оцінки точності прогнозів;
- проаналізувати результати роботи моделі за допомогою відповідних метрик та обґрунтувати її ефективність у різних сценаріях використання.

Апробація роботи. Робота апробувалася на конференції «Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025)» [2].

Публікації. Електронні тези на тему «Інформаційна система прогнозування вартості комерційних та житлових приміщень на ринку нерухомості» [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз особливостей об'єктів житлової та комерційної нерухомості

Ринок нерухомості є одним із найважливіших секторів економіки, що характеризується попитом, який постійно зростає у зв’язку із збільшенням населення та урбанізацією. Основними розгалуженнями на ринку нерухомості є житлові приміщення, та приміщення призначені для комерційної діяльності.

Житлова нерухомість охоплює об’єкти, призначені для постійного або тимчасового проживання. Це можуть бути квартири, приватні будинки, багатоквартирні житлові комплекси, гуртожитки тощо. Ключові характеристики, що впливають на вартість житлової нерухомості, включають загальну та житлову площу, поверховість, кількість кімнат, рік побудови, наявність та стан ремонту, тип будинку (панельний, цегляний, монолітний), інфраструктура поруч (школи, дитячі садки, магазини, транспорт), екологічна ситуація в районі, а також розташування — центральна частина міста, спальний район чи передмістя. Крім того на формування вартості таких приміщень суттєво впливає наявність паркінгу, платного чи безкоштовного, наявність охорони території, наявність місць для відпочинку на зразок парків чи дитячих майданчиків. На вторинному ринку додатково враховуються стан будівлі, присутність комунікацій, а також історія користування [3].

Житлова нерухомість характеризується відповідним правовим статусом. Згідно із законодавством, вона використовується виключно з цільовим призначенням, тобто для проживання. Зміна функціонального призначення можлива лише через проходження спеціальної процедури. У більшості країн житлові приміщення підлягають суворішому регулюванню щодо шуму, безпеки, реєстрації місця проживання тощо.

У свою чергу, комерційна нерухомість включає об’єкти, призначені для ведення різного роду комерційної діяльності. Це можуть бути офісні будівлі, магазини, складові приміщення, готелі та ресторани, торговельні центри,

виробничі заводи тощо. На оцінку вартості таких приміщень впливають інші категорії, відмінні від житлових будівель. Ціна комерційних будівель формується на основі призначення самого приміщення, загальної площі, розташування наявність інженерних комунікацій. Інженерна комунікація охоплює кілька критеріїв, таких як якість електропостачання, системи вентиляції та кондиціонування, опалення, телекомунікаційні мережі. Одним із важливих факторів є також комерційна привабливість району, де розташований об’єкт, та наявність подібних конкурентних будівель поблизу [3].

Комерційні об’єкти мають гнучкіші умови в плані перепланування та змін інтер'єру, оскільки часто адаптуються під вимоги конкретного орендаря або бізнесу. Такі приміщення розміщені в районах з меншим регулюванням шуму, не підлягають скаргам за суб’єктивними причинами (до прикладу скарг в житлових будинках між сусідами). Водночас вони підпадають під інакші норми будівництва та експлуатації, зокрема пожежної безпеки, доступності для маломобільних груп населення, енергетичної ефективності та санітарного контролю. Також комерційні приміщення часто передбачають наявність вхідної групи з фасаду, вітрини, паркування, доступу вантажного транспорту або логістичних можливостей [4].

Ключовою відмінністю між цими двома категоріями є їх функціональне призначення. Житлова нерухомість має задовольняти базові потреби людини для зручності проживання, в той час як комерційна створена для забезпечення умов ведення господарської діяльності, що вимагає ефективного використання площі, зручного розміщення, адаптивності до цільової аудиторії.

У сучасному містобудуванні набирають популярності змішані формати (mixed-use), коли житлова і комерційна функції поєднуються в межах одного будинку або житлового комплексу. Це дозволяє ефективніше використовувати площу, а також забезпечувати мешканців об’єктами інфраструктури в межах пішохідної доступності, або ж навіть в межах будинку.

Аналіз особливостей об'єктів житлової та комерційної нерухомості дозволяє визначити ключові вимоги до програмного модуля оцінки вартості. Зокрема, система має враховувати різне функціональне призначення об'єктів, специфіку

їхнього планування, типові характеристики та регуляторні обмеження. Це забезпечить точніше налаштування моделі, гнучку адаптацію до типу нерухомості та релевантність результатів оцінки.

На формування вартості об’єктів також впливає призначення. Якщо житлова нерухомість оцінюється здебільшого з позиції комфорту проживання, безпеки, розвиненої інфраструктури та екологічної ситуації, то для комерційної нерухомості пріоритетним є її здатність генерувати стабільний дохід. Тому, наприклад, для квартири чи приватного будинку вирішальним буде розташування поруч із школами, парками або громадським транспортом, тоді як для офісу чи магазину — інтенсивність пішохідного чи автомобільного трафіку, наявність паркувальних місць, можливість рекламного розміщення, а також популярність району, наявність потенційної клієнтури, або ж навпаки відсутність подразників для підприємницької діяльності.

На українському ринку житлова нерухомість традиційно є найбільш активним сегментом, зокрема у великих містах, таких як Київ, Львів, Одеса, Харків та Дніпро. У воєнний та післявоєнний період попит на житло значно зростає у більш мирних регіонах, у зв’язку з внутрішньою міграцією, а також потребами оновлення старого житлового фонду. Більшість осіб з внутрішньої міграції оселяються у великих містах, у їх центральних частинах. Це пов’язано із пошуком потенційної роботи в подальшому. Комерційна нерухомість навпаки демонструє нерівномірний розвиток. Так, наприклад, центрі Києва та Львова спостерігається стабільний попит на офіси та street-retail, однак у регіонах відбувається спад через скорочення малого бізнесу через не окупність [4].

Якщо розглянути світовий ринок нерухомості, то можна спостерігати тенденцію до зростання гнучких форматів комерційної нерухомості, таких як коворкінги, а також урбаністичних проєктів із комбінованим призначенням. Такими проектами можуть бути об’єднання житлової та комерційної площі в одному комплексі, наприклад ЖК. У розвинених країнах, таких як США, Німеччина, Нідерланди, нерухомість усе частіше оцінюється не лише за статичними параметрами, а й із використанням інструментів прогнозування

дохідності та ризику [3-4]. Тобто, все більше залучаються автоматизовані системи оцінки вартості нерухомості та прогнозування.

Всі фактори, що впливають на формування вартості нерухомості, можна умовно поділити на макроекономічні, регіональні, локальні та об'єктні. Макроекономічні фактори діють на рівні національної економіки та впливають на ринок нерухомості загалом. До них належать економічна ситуація в країні, рівень інфляції, законодавче регулювання галузі, демографічні тенденції та міграційні процеси. Економічна ситуація в країні безпосередньо впливає на платоспроможність населення та бізнесу, а отже, і на попит на нерухомість. Періоди економічного зростання зазвичай супроводжуються збільшенням ділової активності, зростанням доходів населення та, як наслідок, підвищенням цін на нерухомість. Натомість економічні кризи та рецесії призводять до скорочення доходів, зниження інвестиційної активності та падіння цін. Рівень інфляції також є вагомим фактором, оскільки нерухомість традиційно розглядається як інструмент захисту від інфляції, що може стимулювати попит на неї в періоди високої інфляції. Такі фактори не враховуються в автоматизованих системах напряму, тобто не висуваються окремим критерієм при прогнозуванні вартості на нерухомість, проте вони задають тенденцію впливу в глобальному плані [1, 5].

Регіональні фактори відображають специфіку конкретного регіону чи міста та впливають на загальний рівень цін у даній локації. Якщо регіон є популярним серед населення, має зручність розташування вартість приміщень у ньому збільшиться. До таких факторів належать економічний розвиток регіону, рівень доходів населення, стан місцевого ринку праці, темпи будівництва нової нерухомості, розвиненість транспортної інфраструктури, розташування. В Україні спостерігається значна регіональна диференціація цін на нерухомість, що відображає нерівномірність економічного розвитку різних регіонів. Найвищі ціни традиційно фіксуються у Києві та найбільших обласних центрах, де відповідно сконцентровані ділова активність, робочі місця та інвестиції.

Розвиненість транспортної інфраструктури є важливим фактором ціноутворення, особливо для великих міст. Наявність зручного транспортного

сполучення з центром міста та іншими важливими районами, доступність громадського транспорту, якість доріг – усе це суттєво впливає на привабливість району та, відповідно, на вартість нерухомості в ньому. Транспортне сполучення може включати в себе метро, автобусні, тролейбусні чи трамвайні зупинки поруч з будівлею, або ж в пішій доступності.

Локальні фактори діють на рівні конкретного району, для окремої будівлі та можуть суттєво диференціювати ціни на нерухомість навіть в межах одного міста. До них належать престижність району, розвиненість соціальної інфраструктури (школи, дитячі садки, медичні заклади, магазини, розважальні центри), наявність парків та рекреаційних зон. Це один з найбільш впливових факторів при прогнозуванні вартості, разом із об’єктними характеристиками. Об'єктні фактори стосуються безпосередньо характеристик самого об'єкта нерухомості та є найбільш різноманітними та специфічними для різних типів нерухомості. Для житлової нерухомості ключовими об'єктними факторами є технічні характеристики будівлі (матеріал стін, рік побудови, технічний стан), параметри приміщення (загальна та житлова площа, кількість кімнат, поверх, висота стелі, наявність балконів та лоджій), якість внутрішнього оздоблення, наявність та стан інженерних комунікацій, енергоефективність, правовий статус об'єкта.

Для торговельної нерухомості особливо важливими є такі фактори, як прохідність, видимість з вулиці, наявність вітрин, зручність доступу для покупців, можливість організації розвантаження товарів. Для офісної нерухомості значущими є престижність адреси, якість спільних приміщень та рівень управління будівлею, наявність конференц-залів та переговорних кімнат, рівень безпеки, якість телекомунікаційних мереж. Для складської та виробничої нерухомості ключовими є транспортна доступність, можливість під'їзду вантажного транспорту, наявність залізничних гілок, висота стель, несучі здатності підлог, наявність кранового обладнання, потужність енергопостачання [4–6].

Для створення ефективного програмного модуля визначення вартості нерухомості необхідно враховувати всі зазначені групи факторів. При цьому важливо виокремити ключові критерії, які впливають на ціну. Перш за все це

розподілення та комерційну та житлову нерухомість. Спільними критеріями для обох категорій будуть розташування (місто / район), відстань до центру міста, розвиненість інфраструктури, транспортна доступність. На житлову нерухомість найбільш впливають такі критерії, як площа, кількість кімнат, поверх, рік побудови, тип будівлі, наявність ремонту, наявність паркування. В свою чергу на комерційні приміщення окремо впливає також площа приміщення, популярність району, тип об'єкта, поверх, видимість , можливість перепланування, паркування або ж зручна логістика.

1.2 Аналітичний огляд існуючих методів оцінювання вартості нерухомості

Оцінювання нерухомості є визначенням ринкової ціни об'єкта нерухомого майна станом на певну дату з урахуванням його фізичних, економічних, юридичних та інших характеристик. Метою такого оцінювання є встановлення обґрунтованої вартості, яка відображає справедливу ціну приміщення чи будівлі на відкритому ринку між сторонами угоди. Цей процес зазвичай здійснюється сертифікованими фахівцями з оцінки нерухомості з використанням загальноприйнятих методів, які також називають традиційними. До них належать порівняльний, або ринковий метод, витратний та дохідний методи, метод капіталізації доходу, метод дисконтування грошових потоків (DCF), а також метод залишку. Оцінка вартості нерухомості необхідна у різних ситуаціях, наприклад при укладанні угод купівлі-продажу, в разі потреби іпотечного кредитування, страхування, оподаткування майна, інвестування, розподілу майна тощо. Вона забезпечує об'єктивне підґрунтя для ухвалення фінансових рішень і є важливим інструментом не лише на ринку нерухомості, але у фінансово-економічній сфері діяльності фізичних чи юридичних осіб [7].

Порівняльний, або ринковий спосіб ґрунтується на припущенні того, що вартість нерухомості відображає ціну, яку потенційний покупець готовий сплатити в умовах, максимально наближених до ринкових. Цей метод потребує ретельного

вивчення недавніх операцій купівлі-продажу, які стосуються аналогічних об'єктів, розташованих у тому ж, або ж схожому за характеристиками, районі. Оцінювач виконує корекцію вартості об'єктів-аналогів, беручи до уваги наявні між ними відмінності: розмір, поточний стан, місце розташування, технічні особливості та інші чинники, що впливають на вартість. Метод є популярним у житловому сегменті, де на ринку зазвичай присутня велика кількість подібних об'єктів, що дає змогу оперативно визначити ринкову вартість нерухомості на основі даних із відкритих джерел.

Витратний метод, також відомий як собівартісний, базується на принципі, що обґрунтована ринкова вартість не може бути вищою за витрати, понесені для відтворення ідентичного об'єкта з урахуванням зносу. Іншими словами, у рамках цього методу вартість визначається шляхом підрахунку витрат на відтворення (точна копія) або заміщення (аналогічний функціонально) будинку, додавання вартості земельної ділянки та віднімання накопиченого зносу. Знос враховується у трьох основних аспектах: фізичний, функціональний та зовнішній. Метод найефективніший при оцінці нових будівель, спеціалізованих об’єктів нерухомості та тих, що мають обмежену кількість аналогів на ринку та відсутність порівняння.

Метод доходу ґрунтується на оцінці майбутньої користі, яку здатна принести нерухомість, і застосовується переважно для комерційних об'єктів. У рамках доходного методу виділяють два основні підходи. Перший, метод капіталізації доходу, використовується у випадках, коли об'єкт генерує стабільний прибуток. Його суть полягає у розподілі чистого операційного доходу (NOI) на коефіцієнт капіталізації, який відображає очікувану рентабельність. Другий підхід – метод дисконтування грошових потоків (DCF) – використовується, якщо доходи нестабільні або їх передбачають на тривалий період. Оцінювач прогнозує усі майбутні грошові потоки (доходи за вирахуванням витрат) і дисконтує їх до поточної вартості, використовуючи відповідну ставку дисконтування. Цей спосіб дозволяє враховувати інвестиційні ризики, ринкову динаміку, інфляційні процеси та специфічні особливості конкретного об'єкта [8–9].

Метод залишку використовується для визначення вартості земельної ділянки, яку планується забудувати, або об'єкта нерухомості, що має перспективу розвитку. Згідно з цим методом, спочатку робиться прогноз майбутніх надходжень від експлуатації об'єкта. Наприклад, прибуток від продажу квартир у зведеному будинку. Після прогнозу цих надходжень з них віднімаються всі витрати на будівництво, проєктні роботи, просування на ринку та інші супутні витрати. Результат, який залишається, є приблизною оцінкою вартості землі або потенційного інвестиційного привабливості об'єкта. Даний метод активно застосовується в галузі містобудування, зокрема при розробці планів будівництва нових житлових чи комерційних комплексів [9].

В останні десятиліття традиційні методи оцінки доповнюються та поступово витісняються алгоритмічними підходами, які дозволяють автоматизувати процес оцінки та підвищити його об'єктивність. Основними такими методами є регресійний аналіз, методи машинного навчання та гібридні методи.

Методи оцінювання нерухомості, що базуються на машинному навчанні (ML), є більш сучасним підходом, який використовує алгоритми штучного інтелекту для моделювання та передбачення ринкової вартості об'єктів. На відміну від традиційних методів, ML-підходи мають здатність автоматично виявляти складні нелінійні взаємозв'язки між різноманітними змінними, такими як характеристики нерухомості, її місцезнаходження, умови ринку, демографічні показники, індекси цін та інші. Найбільш поширені алгоритми у цьому контексті охоплюють лінійну та поліноміальну регресію, дерева рішень, випадкові ліси (random forest), градієнтний бустинг (XGBoost, LightGBM) та нейронні мережі, включно з глибоким навчанням (deep learning). Ці моделі навчаються на великих масивах даних історичних операцій купівлі-продажу, географічної інформації та макроекономічних індикаторів. Після етапу навчання модель може швидко оцінювати нові об'єкти, демонструючи високу точність за наявності якісного та об'ємного набору даних [10-11]. Порівняльний аналіз даних методів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз методів машинного навчання

Метод	Складність реалізації	Точність	Швидкість обчислень	Випадки використання
Лінійна регресія	Низька	Середня	Висока	Базове прогнозування, експрес-оцінка, початковий аналіз ринку
Дерева рішень / Random Forest	Середня	Висока	Середня	Комплексна оцінка, аналіз факторів впливу, робота з категоріальними даними
Gradient Boosting (XGBoost, LightGBM)	Висока	Дуже висока	Середня	Професійна оцінка, змагання з точності, комерційні системи оцінки
Нейронні мережі	Дуже висока	Висока	Низька	Великі датасети, комбінування різних типів даних, дослідницькі проєкти
Кластеризація (k-means)	Низька	Низька	Висока	Сегментація ринку, попередній аналіз, визначення типів нерухомості

Розглядаючи детальніше, лінійна регресія створює модель зв'язку між вартістю житла та його ознаками за допомогою лінійного рівняння. Алгоритм відшукує найкращу пряму, котра ілюструє взаємозв'язок між параметрами, такими як площа, число кімнат, місце розташування та ціна конкретного об'єкту. Метод застосовує метод найменших квадратів, щоб мінімізувати розбіжність між фактичними та передбаченими цінами. Його перевагами є легкість впровадження

та розуміння здобутків, швидкість обчислень, можливість оперувати з невеликими обсягами даних. Проте, припущення про лінійну залежність часто не відповідає дійсності на ринку нерухомості, а також метод має сприйнятливість до аномальних значень.

Методи градієнтного бустингу функціонують за механізмом поступового покращення прогнозу, включаючи слабкі моделі (часто неглибокі дерева), кожна з яких корегує похибки попередніх. Алгоритм зосереджується на найважчих для передбачення випадках, планомірно знижуючи загальну похибку моделі. XGBoost та LightGBM - це оптимізовані реалізації цього підходу з додатковими методами регуляризації та прискорення обчислень. Перевагою є висока точність прогнозування, здатність автоматично виявляти складні патерни та вбудовані механізми регуляризації. Недоліками ж методу є ризик перенавчання при невірному налаштуванні і складність інтерпретації результатів.

Нейронні мережі функціонують подібно до людського мозку, застосовуючи шари сполучених штучних нейронів для розпізнавання складних нелінійних взаємодій між властивостями нерухомості та їх вартістю. Багатошарові перцептрони містять вхідний шар (особливості об'єкта), один або декілька прихованих шарів (що виявляють приховані закономірності) та вихідний шар (прогнозована ціна). Процес навчання відбувається за допомогою алгоритму зворотного розповсюдження помилки.

Кластеризація k-means групує об'єкти нерухомості в кластери на підставі подібності їх властивостей, розділяючи ринок на сегменти зі схожими рисами. Алгоритм ітеративно приписує кожний об'єкт до найближчого центроїда кластера та оновлює позиції центроїдів до здобуття оптимального розподілу. Хоча цей спосіб не застосовується безпосередньо для оцінки ціни, він допомагає в попередньому аналізі ринку та сегментуванні даних.

Дерева рішень будують деревоподібну структуру правил для ухвалення рішень, поступово розділяючи дані на основі найбільш значущих властивостей об'єкта. Random Forest поєднує багато дерев рішень, кожне з яких вивчається на випадковій вибірці даних та ознак, а кінцевий прогноз отримується шляхом

усереднення результатів усіх дерев. Це дозволяє враховувати складні нелінійні взаємозв'язки між характеристиками та виявляти приховані шаблони в даних. Недоліками дерев рішень є схильність до перенавчання (особливо поодинокі дерева), складність інтерпретації для Random Forest та потреба у великих обсягах даних для стабільних результатів [12–13].

Тим не менш, дерева рішень виділяються своєю унікальною інтерпретованістю. Їхню логіку просто пояснити у вигляді зрозумілих правил на зразок "якщо площа перевищує 100 кв. м і район елітний, ціна буде понад 200 000 грн". Це суттєво відрізняє їх від нейронні мережі або градієнтний бустингу. На відміну від лінійної регресії, дерева самі розпізнають нелінійні залежності та взаємодії між змінними, без потреби попередньо визначати їх. Вони також легко обробляють і числові, і категоріальні дані, без додаткової підготовки, що є перевагою при аналізі даних різних категорій. Більше того, дерева рішень показують стійкість до викидів та не потребують нормалізації даних, що робить їх надзвичайно привабливими для швидкого прототипування і в ситуаціях, де ключовим є прозорий процес ухвалення рішень для стейкхолдерів у сфері нерухомості. Схему роботи алгоритму Random Forest наведено на рисунку 1.1 [14 – 16].

ML-методи вже широко застосовуються платформами нерухомості (як-от Zillow чи Redfin) та банками для автоматизованої масової оцінки нерухомості (AVM — Automated Valuation Models).

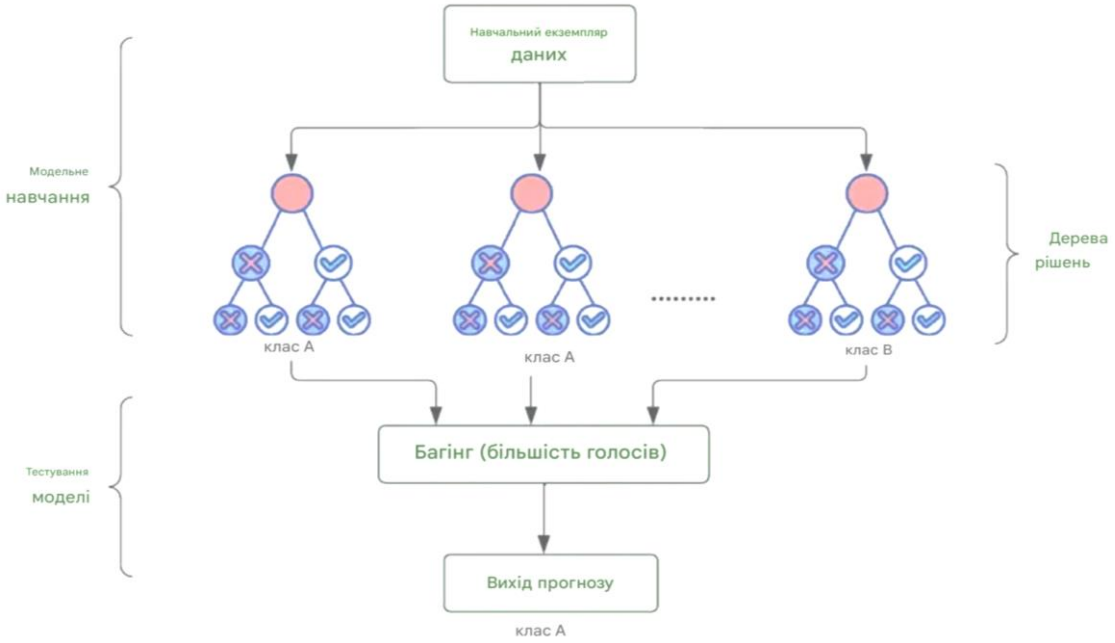

Рисунок 1.1 – Схема роботи алгоритму Random Forest [14]

1.3 Аналіз відомих програмних засобів оцінки вартості нерухомості

На сучасному ринку існує значна кількість програмних систем, що реалізують описані вище методи оцінки. Zillow Zestimate (США) — одна з найпопулярніших систем автоматизованої оцінки нерухомості, що охоплює понад 110 мільйонів об'єктів у США. Система використовує гібридний підхід, поєднуючи різні методи машинного навчання з просторовим аналізом. Zestimate враховує сотні мільйонів точок даних, включаючи характеристики об'єктів, дані про продажі, податкову інформацію та інформацію про ринок. Точність оцінки постійно вдосконалюється — середня похибка для житлової нерухомості складає близько 4-5% [17].

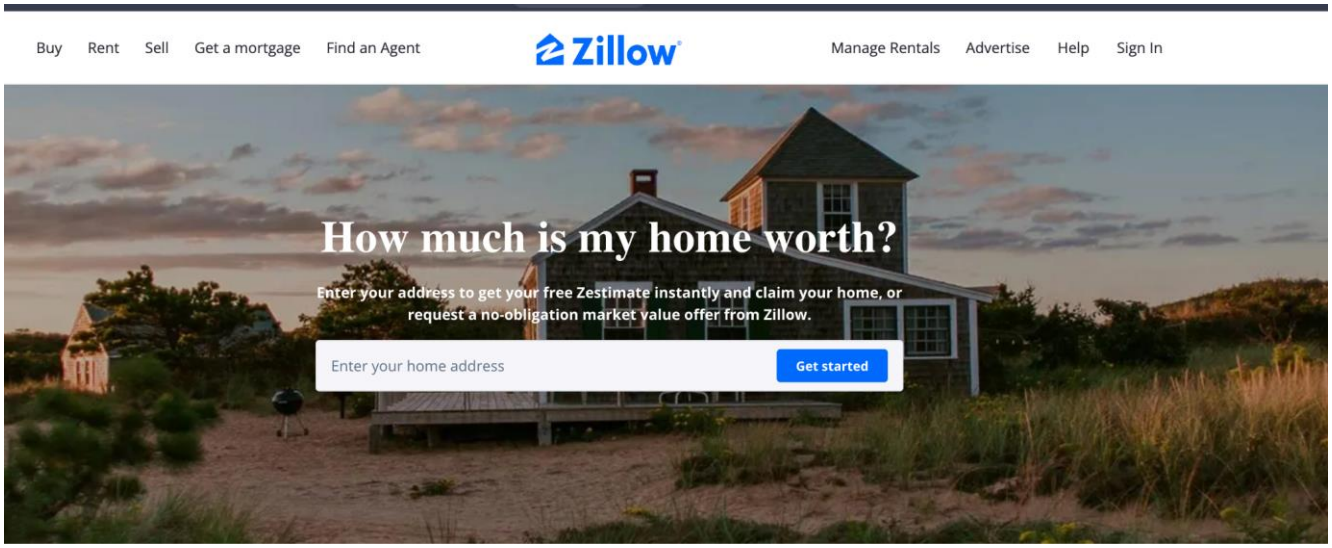

Рисунок 1.2 – Головна сторінка сайту Zillow Zestimate [17]

Real Price (Європа) — європейська платформа оцінки нерухомості, що використовує алгоритми глибокого навчання та аналізує понад 700 параметрів для кожного об'єкта. Система інтегрує дані з різних джерел, включаючи державні реєстри, оголошення про продаж та оренду, географічні дані та дані про інфраструктуру. Особливістю RealPrice є можливість щомісячного оновлення оцінок, що дозволяє відстежувати динаміку ринку [18].

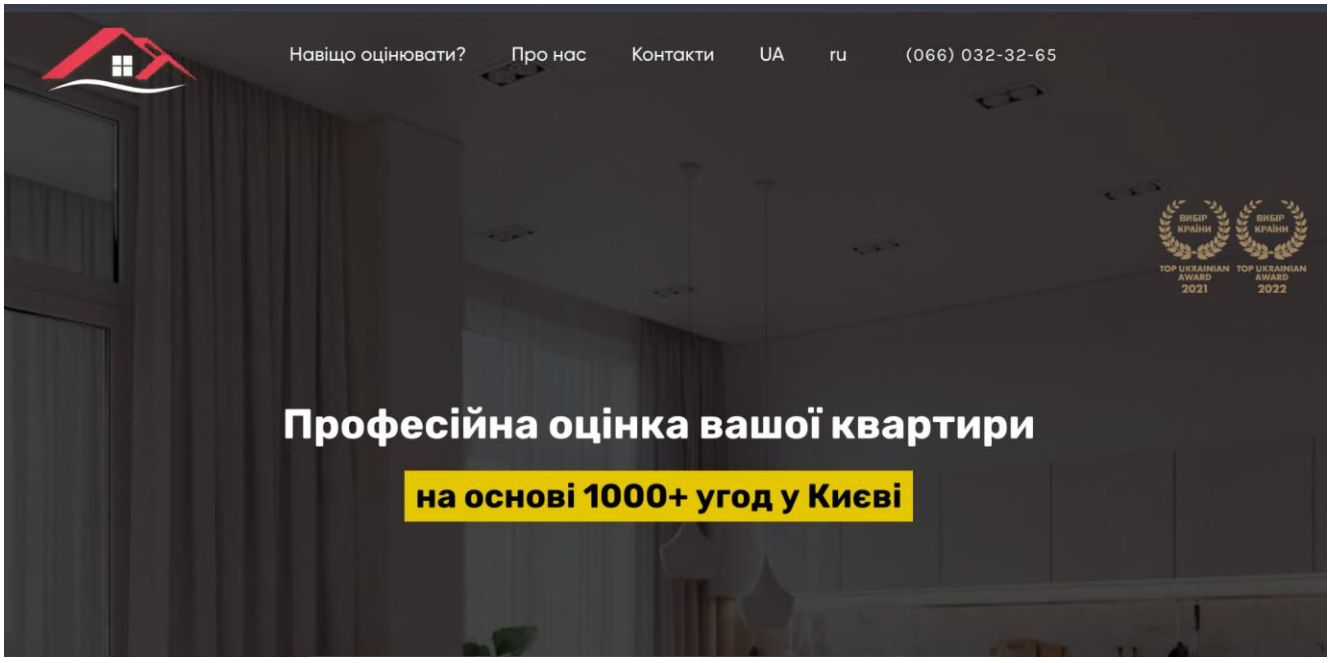

Рисунок 1.3 – Головна сторінка сайту Real Price [18]

Estimate.RealEstate (Україна) — один з найбільших українських порталів нерухомості, що пропонує функцію автоматизованої оцінки житла на основі аналізу ринкових пропозицій. Система використовує методи машинного навчання для аналізу даних про продаж та оренду квартир, а також враховує такі фактори, як місцезнаходження, площа, кількість кімнат, поверх, тип будинку тощо. Lun.ua також надає аналітику ринку нерухомості по різних регіонах України [19].

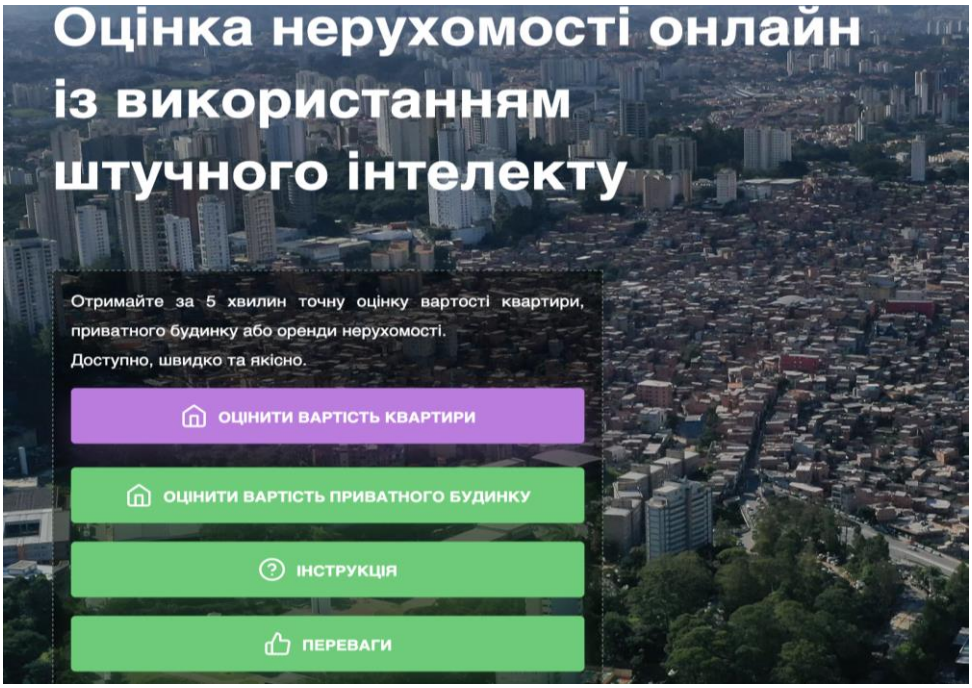

Рисунок 1.4 – Головна сторінка сайту Lun.ua (Україна) [19]

Незважаючи на значний прогрес у розвитку систем автоматизованої оцінки нерухомості, вони все ще стикаються з певними викликами. Недостатність даних — особливо актуальна проблема для ринків, що розвиваються, де інформація про угоди з нерухомістю може бути обмеженою або недостовірною. В Україні, наприклад, значна частина угод може не відображати реальну ціну продажу [19]. Складність врахування якісних факторів — такі характеристики, як якість ремонту, вид з вікна або престижність конкретної адреси, складно формалізувати та включити в алгоритмічні моделі. Затримка в реагуванні на ринкові зміни — більшість систем оновлюють свої моделі з певною періодичністю, що може призводити до неточностей в оцінці в періоди швидких змін на ринку. Регіональна

специфіка — системи, розроблені для одних ринків, можуть бути неефективними на інших через різницю в факторах, що впливають на формування вартості нерухомості. Непрозорість алгоритмів — багато комерційних систем не розкривають деталей своїх алгоритмів, що ускладнює їхню об'єктивну оцінку та порівняння.

1.4. Висновок до розділу 1

Проведений аналіз особливостей об'єктів житлової та комерційної нерухомості виявив принципові відмінності між цими категоріями, що зумовлені їхнім функціональним призначенням. Встановлено, що житлова нерухомість орієнтована на забезпечення комфортного проживання і характеризується такими основними параметрами, як площа, кількість кімнат, поверховість, розташування відносно інфраструктури та транспортних сполучень. Натомість комерційна нерухомість призначена для генерування прибутку і оцінюється за іншими критеріями: комерційною привабливістю району, прохідністю, видимістю, наявністю паркування та можливостями для логістики. Під час аналізу факторів впливу, було виділено чотири рівні факторів: макроекономічні, регіональні, локальні та об'єктні, що дозволило визначити ключові критерії для розробки алгоритму оцінювання вартості нерухомості. Аналітичний огляд існуючих методів та систем оцінювання вартості нерухомості виявив еволюцію підходів від традиційних (порівняльний, дохідний, витратний) до сучасних алгоритмічних, заснованих на методах машинного навчання. Визначено, що найбільш ефективними є гібридні підходи, які поєднують різні методи та враховують просторові залежності цін на нерухомість.

Проведений аналіз предметної області створює міцну теоретичну базу для подальшого проєктування та розробки програмного модуля визначення вартості комерційних та житлових приміщень на ринку нерухомості, що буде здійснено в наступних розділах.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ОЦІНКИ ВАРТОСТІ ОБ'ЄКТІВ НЕРУХОМОСТІ

2.1 Розробка архітектури програмного модулю

Для створення ефективної системи визначення вартості нерухомості необхідно забезпечити модульну архітектуру, яка дозволить гнучко налаштовувати окремі компоненти та забезпечить масштабованість системи. Розроблена архітектура програмного модуля заснована на принципах багаторівневої взаємодії компонентів із чітким розмежуванням відповідальності кожного з них.

Розроблене програмне забезпечення представляє собою модульну систему, що складається з п'яти основних компонентів:

- Модуль обробки та підготовки даних — відповідає за завантаження, очищення, нормалізацію та підготовку вхідних даних для подальшого аналізу. Цей модуль виконує фільтрацію некоректних записів, кодування категоріальних змінних та розподіл даних на навчальну і тестову вибірки.
- Модуль прогнозування вартості — реалізує алгоритм Random Forest для прогнозування цін на нерухомість на основі підготовлених даних. Цей модуль забезпечує високу точність прогнозування завдяки використанню ансамблю дерев рішень.
- Модуль виявлення аномалій — використовує алгоритм Isolation Forest для виявлення нетипових об'єктів нерухомості, що допомагає підвищити надійність прогнозування та ідентифікувати потенційно проблемні дані.
- Модуль формування результатів — відповідає за обробку результатів аналізу, формування прогнозованих цін та розрахунок метрик якості моделі.
- Модуль візуалізації — забезпечує графічне представлення результатів аналізу через різні типи графіків та діаграм, а також реалізує інтерфейс взаємодії з користувачем.

Загальна структурна схема розробленого програмного забезпечення представлена на рисунку 2.1.

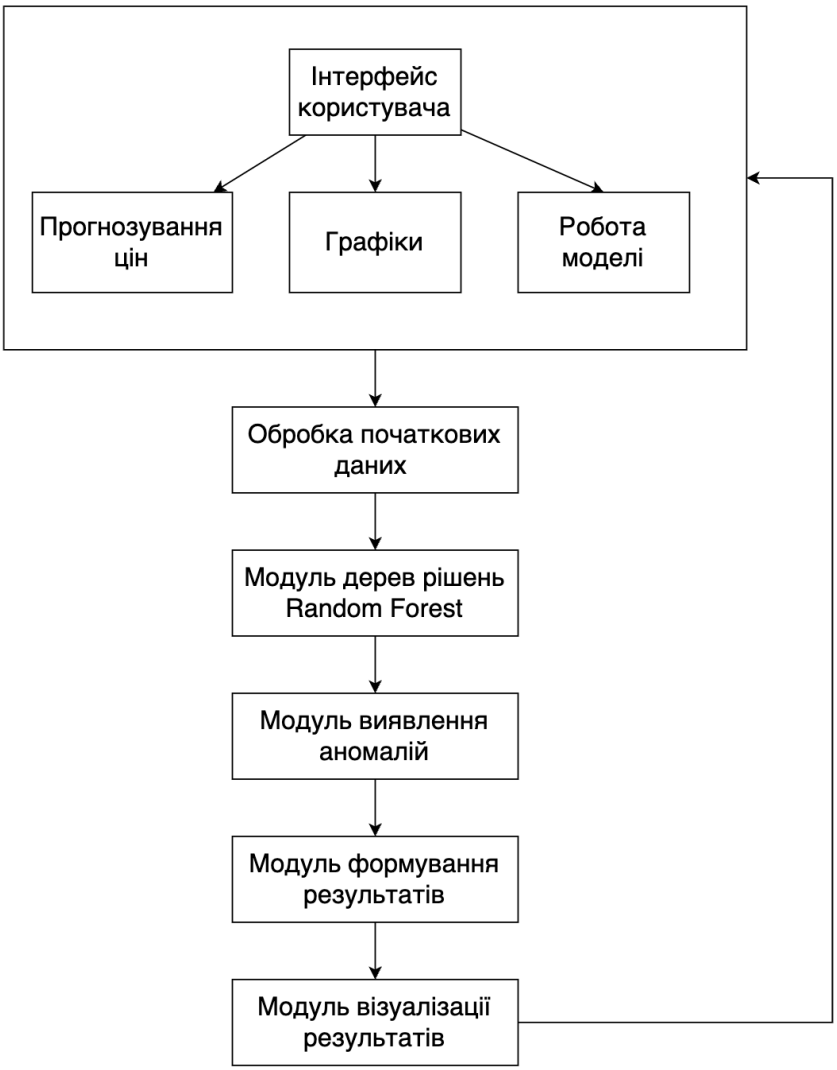

Рисунок 2.1 - Структурна схема архітектури програмного забезпечення

Модульна архітектура забезпечує такі переваги:

- масштабованість — можливість легко додавати нові функціональні можливості або модифікувати існуючі без перебудови всієї системи;
- гнучкість — кожен модуль може бути налаштований або замінений відповідно до конкретних вимог проекту;
- надійність — ізоляція компонентів системи зменшує ризик поширення помилок між модулями;
- повторне використання — можливість використання окремих модулів в інших проектах або системах [20].

В основу розробки покладено модель машинного навчання на базі алгоритму Random Forest, який був обраний через його високу ефективність при роботі з

великими наборами даних, стійкість до перенавчання та здатність визначати важливість різних характеристик у формуванні кінцевої вартості. Для підвищення надійності прогнозування система також включає модуль виявлення аномалій на основі алгоритму Isolation Forest, який дозволяє ідентифікувати нетипові об'єкти нерухомості та враховувати їх при прогнозуванні [21–23].

Взаємодія між модулями здійснюється через чітко визначені інтерфейси, що забезпечує гнучкість і масштабованість системи. Кожен модуль може функціонувати відносно незалежно, отримуючи необхідні дані від інших модулів і передаючи свої результати далі по ланцюгу обробки.

Графічний інтерфейс користувача розроблений з використанням бібліотеки tkinter, що забезпечує кросплатформеність та зручність взаємодії з програмою. Інтерфейс включає форми для введення параметрів нерухомості, кнопки керування, візуалізацію результатів прогнозування та налаштування параметрів моделі.

Для забезпечення ефективної роботи з даними використовуються бібліотеки pandas та numpy, які надають зручні інструменти для маніпуляції та аналізу даних. Для візуалізації результатів застосовуються бібліотеки matplotlib та seaborn, що дозволяють створювати інформативні та естетично привабливі графіки. Така архітектурна організація програмного модуля забезпечує високу гнучкість, надійність та масштабованість системи, що дозволяє ефективно вирішувати задачу визначення вартості комерційних та житлових приміщень на ринку нерухомості з урахуванням різноманітних факторів впливу [24].

2.2 Розробка взаємодії програмних модулів системи визначення вартості нерухомості

Ефективна взаємодія між програмними модулями є ключовим фактором для забезпечення надійності та продуктивності системи визначення вартості нерухомості. У розробленій системі реалізовано структурований підхід до організації взаємодії компонентів, що дозволяє забезпечити чітку послідовність

обробки даних та формування результатів. Взаємодія між модулями здійснюється за принципом "конвеєра", де дані послідовно проходять через усі необхідні етапи обробки. Кожен модуль виконує свою специфічну функцію та передає результати наступному компоненту системи. Для забезпечення гнучкості та можливості паралельної роботи модулів використовується архітектура, що базується на обміні даними через спільні структури та події.

Процес взаємодії модулів починається з ініціалізації системи та завантаження даних. При запуску програми відбувається ініціалізація головного вікна програми через клас Main Application. Модуль обробки даних виконує завантаження початкових даних з CSV-файлу методом load_data(). Відбувається підготовка даних, включаючи фільтрацію некоректних записів, кодування категоріальних змінних та формування необхідних структур даних.

Після підготовки даних відбувається навчання моделей прогнозування та виявлення аномалій. Ініціалізуються моделі Random Forest та Isolation Forest. Модуль прогнозування вартості виконує навчання моделі Random Forest на підготовлених даних методом train_models(). Модуль виявлення аномалій паралельно навчає модель Isolation Forest для ідентифікації нетипових об'єктів. При взаємодії з користувачем інтерфейс приймає введені параметри об'єкта нерухомості та передає їх для аналізу. Модуль валідації даних перевіряє коректність введених користувачем значень методом validate_input_data() та генерує відповідні попередження при необхідності. Підготовлені дані передаються модулям прогнозування та виявлення аномалій.

На етапі формування та візуалізації результатів модуль прогнозування вартості генерує прогноз ціни на основі введених параметрів. Модуль виявлення аномалій визначає, чи є об'єкт типовим для ринку. Модуль формування результатів обчислює метрики якості моделі та підготовлює дані для візуалізації. Модуль візуалізації відображає результати у вигляді графіків та числових показників. Для забезпечення ефективної взаємодії між модулями розроблені механізми передачі даних та обробки подій. Використовуються спільні структури даних (DataFrame, словники) для передачі інформації. Застосовуються методи класів для доступу до

даних між різними компонентами системи. Реалізовані механізми зворотного зв'язку для повідомлення про помилки та статус виконання операцій.

При обробці подій інтерфейсу користувача використовується система реагування на дії користувача, такі як натискання кнопок, введення даних або зміна налаштувань моделі. Кожна подія ініціює відповідну функцію або метод, що дозволяє миттєво обробляти введену інформацію та адаптувати поведінку системи до нових умов. Реалізовано також механізм автоматичного оновлення інтерфейсу при зміні параметрів моделі або після введення нових даних користувачем, що забезпечує динамічний зворотний зв’язок у реальному часі. Крім того забезпечується синхронізація модулів через послідовне виконання операцій, які залежать одна від одної, з урахуванням логіки обробки даних. Для уникнення конфліктів і забезпечення стабільності роботи у модулі було реалізовано механізми завершення тривалих або асинхронних операцій (наприклад, навчання моделі або виявлення аномалій) перед переходом до наступного етапу обробки. Такий підхід гарантує узгодженість стану системи та покращує загальну взаємодію з користувачем. [24].

Для візуалізації взаємодії модулів розроблено схему алгоритму функціонування програмного забезпечення, представлену на рисунках 2.2-2.5.

На поданій на рисунку 2.2 блок-схемі зображено роботу модуля зчитування та попередньої обробки даних, який є першим етапом функціонування системи. Основне завдання цього модуля полягає у завантаженні даних з зовнішнього джерела у форматі CSV, перевірці коректності значень, очищенні записів від некоректних або пропущених даних. Після попередньої обробки інформації відбувається кодування категоріальних змінних, таких як тип нерухомості чи місцезнаходження, у числовий формат, придатний для аналізу. Після цього дані нормалізуються, розподіляються на тренувальну та тестову вибірки і передаються до наступного модуля для подальшої обробки. Модуль призначений для попередньої обробки даних та підготовки їх для подальшої роботи програми.

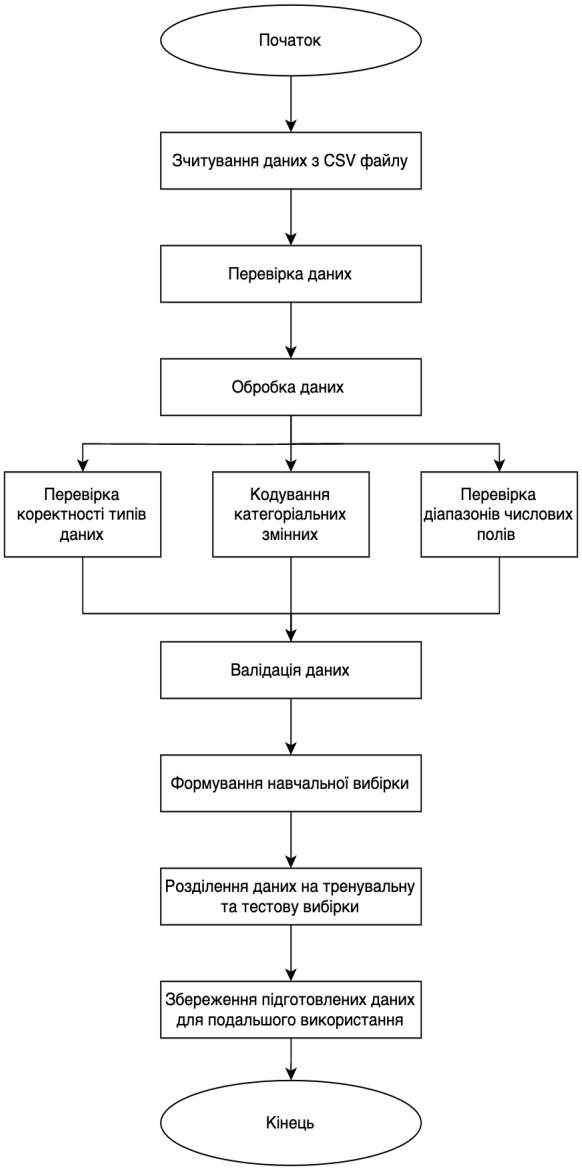

Рисунок 2.2 – Блок-схема модуля зчитування та обробки даних

Функціонування інтелектуального модуля прийняття рішень зображено схематично на рисунку 2.3. Модуль засновано на алгоритмі Random Forest. Він приймає на вхід підготовлені дані, проходить етап ініціалізації моделі з визначенням кількості дерев та інших параметрів, після чого здійснює тренування на тренувальній вибірці. Кожне дерево формує власний прогноз вартості, а результат усереднюється для отримання фінального значення. Після навчання модель готова до прийому нових характеристик об’єкта нерухомості, на основі яких генерується прогноз ціни. Такий підхід забезпечує високу точність та стійкість до перенавчання.

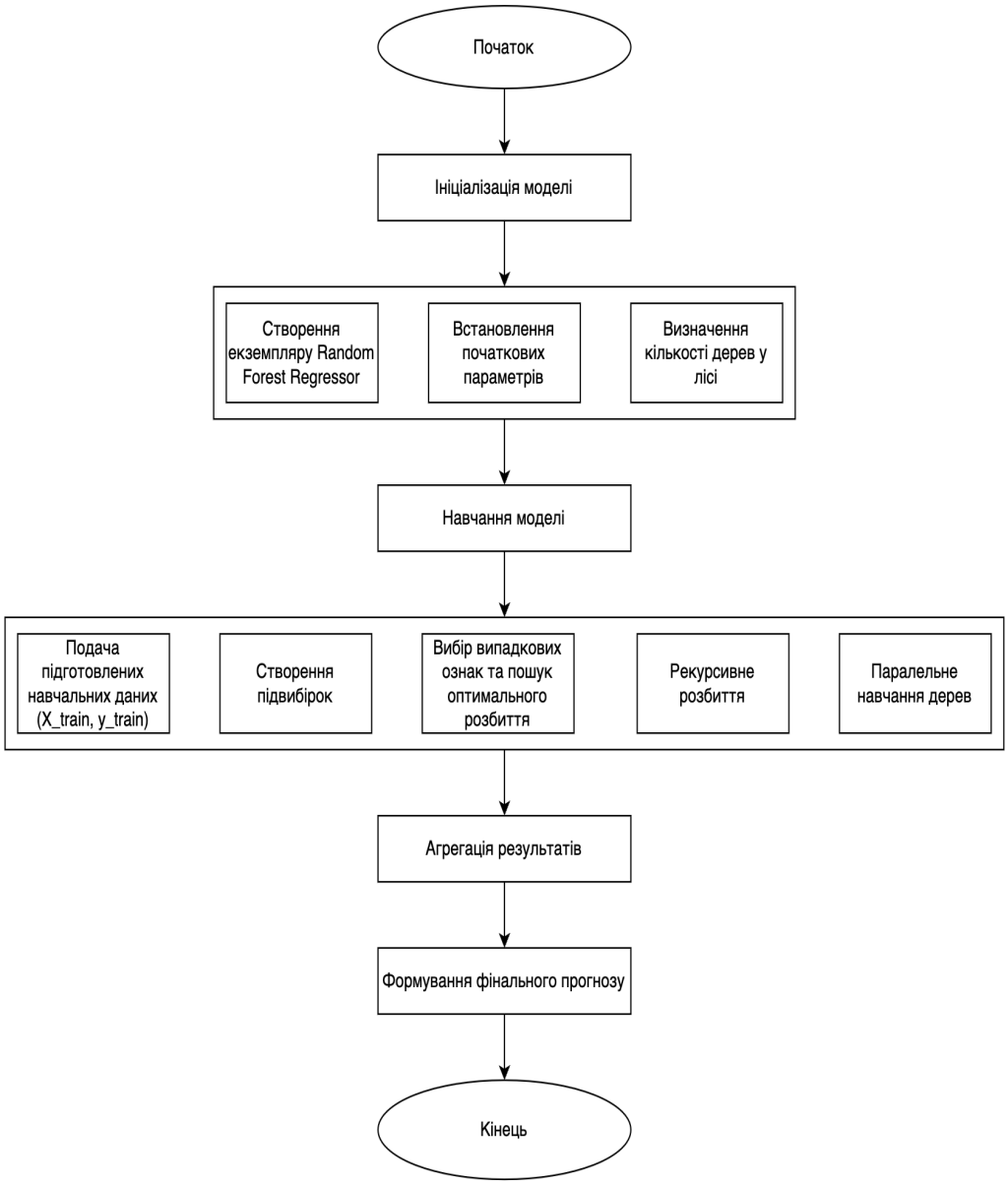

Рисунок 2.3 – Блок-схема інтелектуального модуля прийняття рішень на основі Random Forest

На схемі з рисунка 2.4 зображено процес роботи модуля аналізу та виявлення аномалій, який реалізований за допомогою алгоритму Isolation Forest. Цей модуль функціонує паралельно з основною моделлю прогнозування та призначений для виявлення нетипових або потенційно помилкових об’єктів у даних.

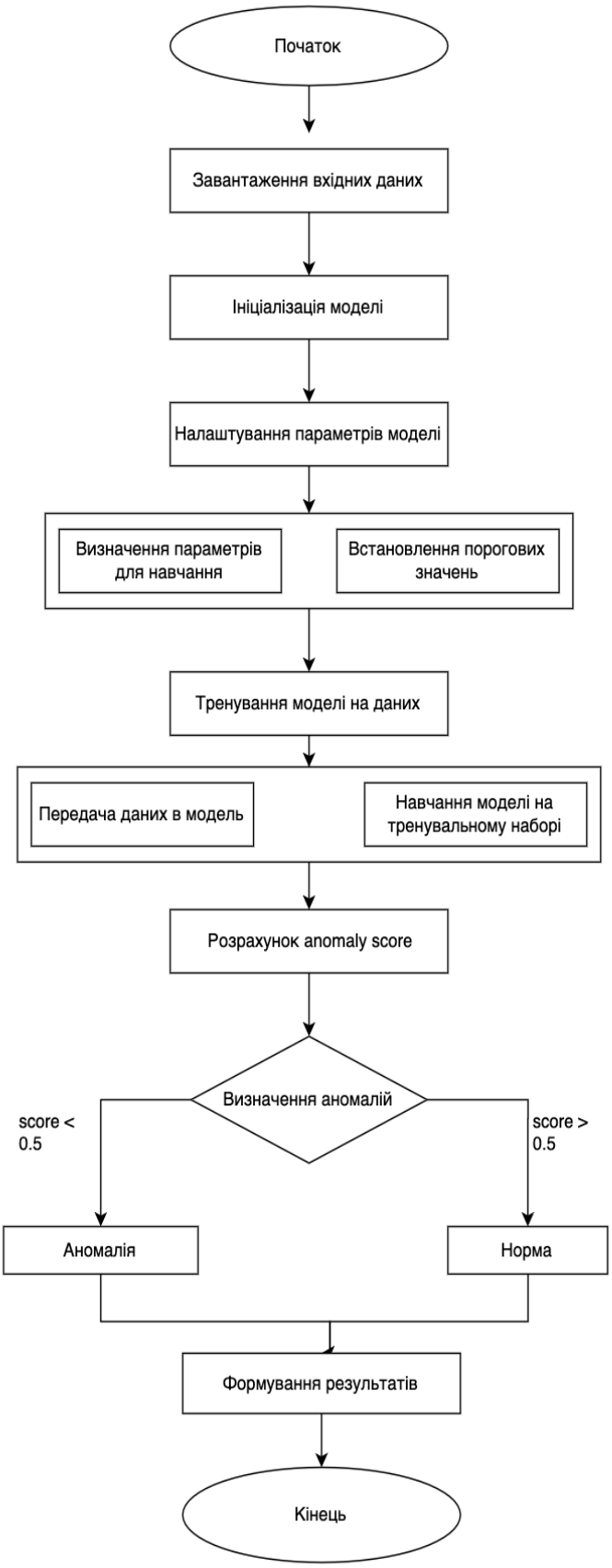

Рисунок 2.4 – Блок-схема модуля аналізу та виявлення аномалій

На рисунку 2.5 наведено блок-схему, яка демонструє роботу модуля генерації та формування результатів, що відповідає за об'єднання прогнозів, розрахунок метрик ефективності моделі (MAE, RMSE, R^2 тощо), а також за підготовку інформації до візуалізації. Отримані результати обробляються для подальшого подання у вигляді графіків, таблиць чи текстових повідомлень. Крім того, цей модуль взаємодіє з інтерфейсом користувача, оновлюючи відображення після кожного запиту. Завдяки цьому користувач отримує зручну й наочну інтерпретацію розрахованої вартості об’єкта та відповідних показників якості прогнозу.

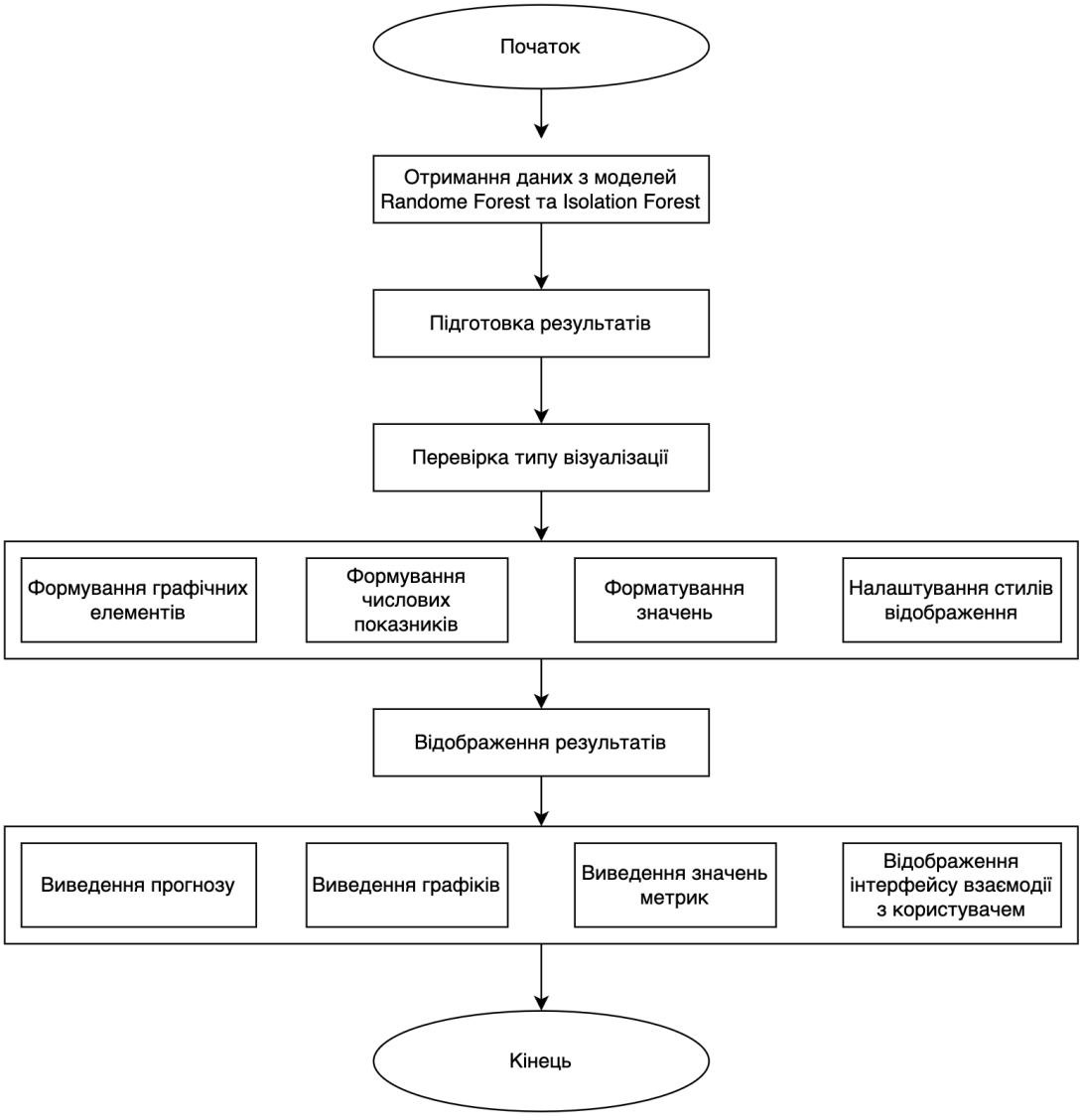

Рисунок 2.5 – Блок-схема модуля генерації та формування результатів

Реалізація взаємодії модулів виконана з використанням об'єктно-орієнтованого підходу, де кожен модуль представлений відповідним класом з чітко визначеним інтерфейсом. Основним класом, що координує роботу всіх компонентів, є Main Application, який відповідає за ініціалізацію системи, створення інтерфейсу користувача та обробку основних подій.

Для роботи з даними використовується клас з методами для завантаження, обробки та валідації даних. Модель прогнозування реалізована через клас, що інкапсулює функціональність Random Forest, а виявлення аномалій — через клас, що взаємодіє з алгоритмом Isolation Forest.

Візуалізація результатів забезпечується класом GraphicsWindow, який використовує методи бібліотек matplotlib та seaborn для створення графіків та діаграм. Цей клас також відповідає за оновлення графічних елементів при зміні параметрів моделі або введенні нових даних користувачем.

Така організація взаємодії модулів забезпечує гнучкість системи, можливість її розширення та модифікації, а також високу надійність роботи. Модульний підхід також спрощує тестування та підтримку системи, оскільки кожен компонент може бути протестований окремо, а помилки локалізовані в межах конкретного модуля.

2.3 Розробка алгоритму визначення вартості комерційних та житлових приміщень на ринку нерухомості

У процесі розробки інтелектуального модуля для прогнозування вартості нерухомості було реалізовано комплексну архітектуру програмного забезпечення, яка базується на використанні алгоритму Random Forest. Цей алгоритм є ключовим елементом системи, оскільки він дозволяє отримувати високоточні прогнози на основі великої кількості вхідних параметрів, притаманних як житловим, так і комерційним об'єктам.

Архітектура розробленого програмного забезпечення складається з трьох функціональних компонентів: модуля попередньої обробки даних, модуля побудови моделі прогнозування на основі Random Forest та модуля аналізу

результатів. Така модульна структура забезпечує масштабованість, простоту підтримки та можливість подальшої адаптації системи до нових типів об’єктів або додаткових характеристик.

Модуль попередньої обробки даних виконує зчитування вхідних даних із CSV-файлу, їх перевірку на коректність (наприклад, фільтрацію нульових чи від’ємних значень), кодування категоріальних ознак (тип будівлі, місце розташування, наявність інфраструктури) та нормалізацію числових параметрів. У цьому ж модулі реалізовано валідацію введених користувачем характеристик, що гарантує узгодженість з тренувальними даними. На рисунку 2.6 зображено загальну схему функціонування програмного забезпечення.

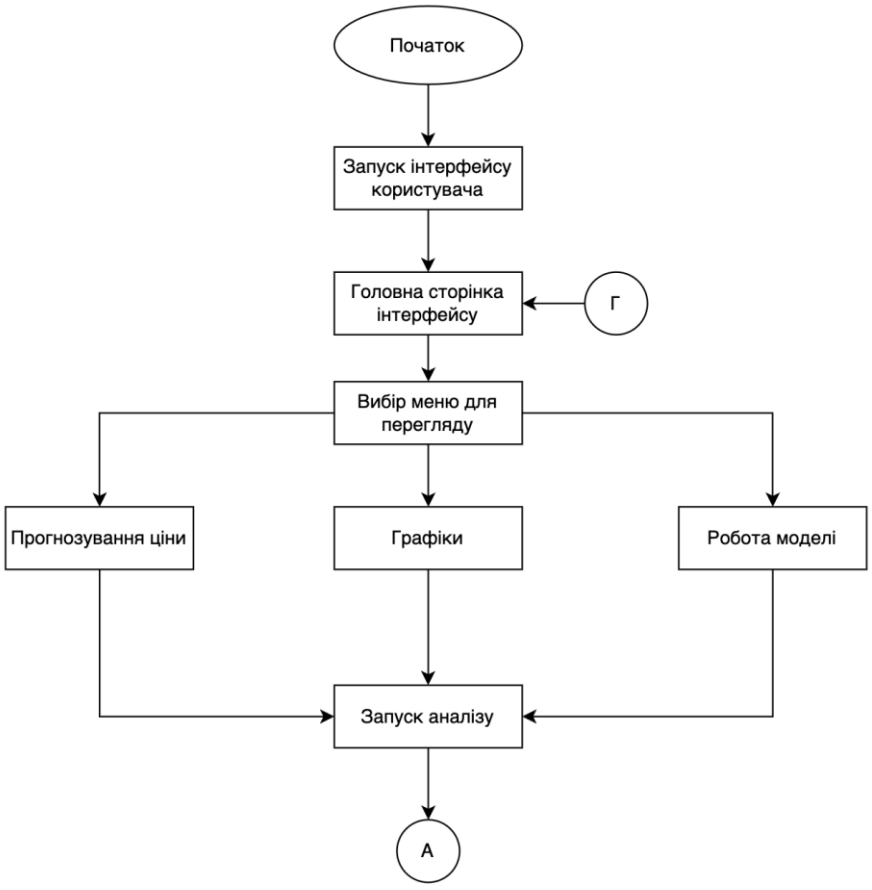

Рисунок 2.6 – Схема алгоритму функціонування програмного забезпечення, частина 1

Модуль побудови моделі є центральним компонентом системи. Алгоритм Random Forest використовується для реалізації багатьох паралельних дерев рішень, кожне з яких будується на різних підмножинах даних із випадково обраними ознаками. Кожне дерево виконує незалежний аналіз, що дозволяє уникнути перенавчання та покращити узагальнення. На виході результати усіх дерев агрегуються шляхом усереднення — отримується фінальне прогнозоване значення вартості об’єкта. Для побудови моделі використовується 80% даних як тренувальна вибірка та 20% як тестова. На рисунку 2.7 наведено продовження схеми функціонування, яке деталізує логіку навчання моделі.

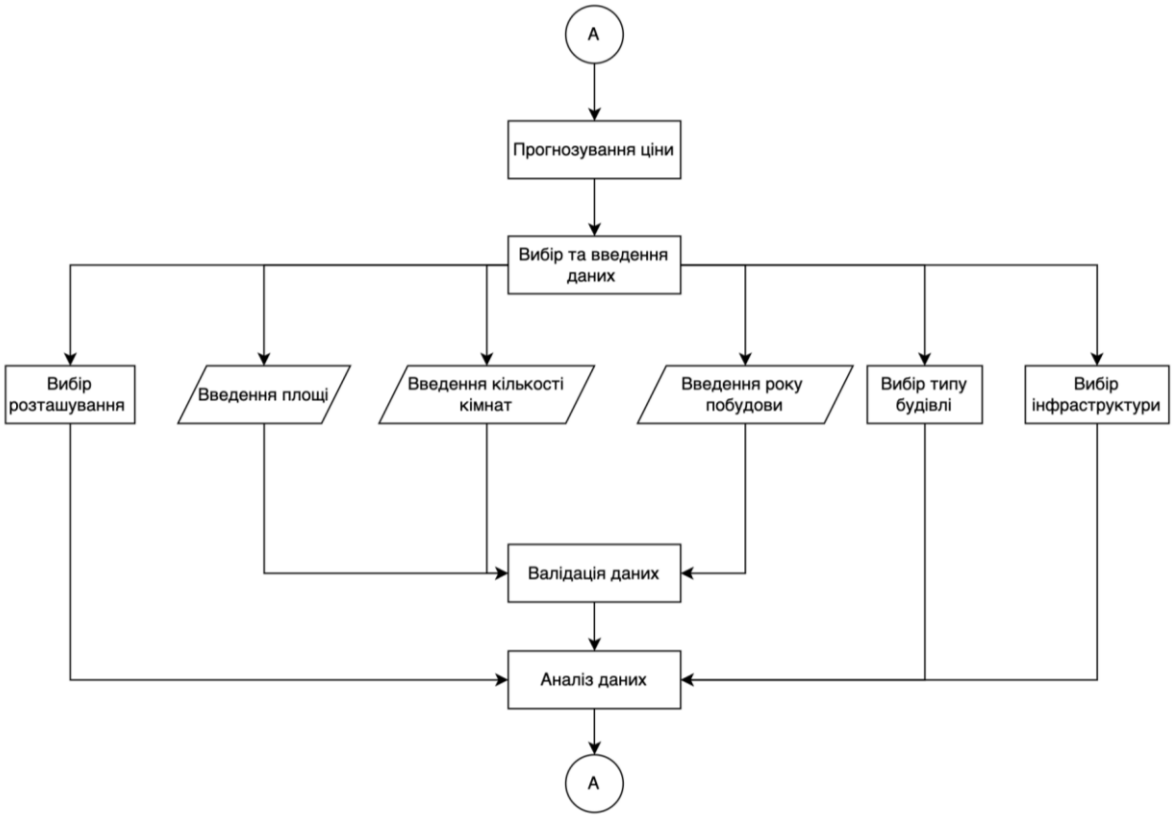

Рисунок 2.7 – Схема алгоритму функціонування програмного забезпечення, частина 2

Особливістю запропонованої архітектури є реалізація гнучкого налаштування моделі: користувач має можливість змінювати кількість дерев у Random Forest, що дозволяє адаптувати точність та швидкодію моделі залежно від поставлених цілей. Також система дозволяє оцінити ефективність побудованих

прогнозів за допомогою чотирьох метрик: MAE, MSE, RMSE та R^2. Ці метрики оновлюються в реальному часі та надають користувачу розуміння якості моделі залежно від вхідних параметрів. Статистичні показники формуються після кожного запуску алгоритму прогнозування. На рисунку 2.8 наведено продовження схеми функціонування, що демонструє відповідну логіку модулів обробки даних та побудови прогнозу.

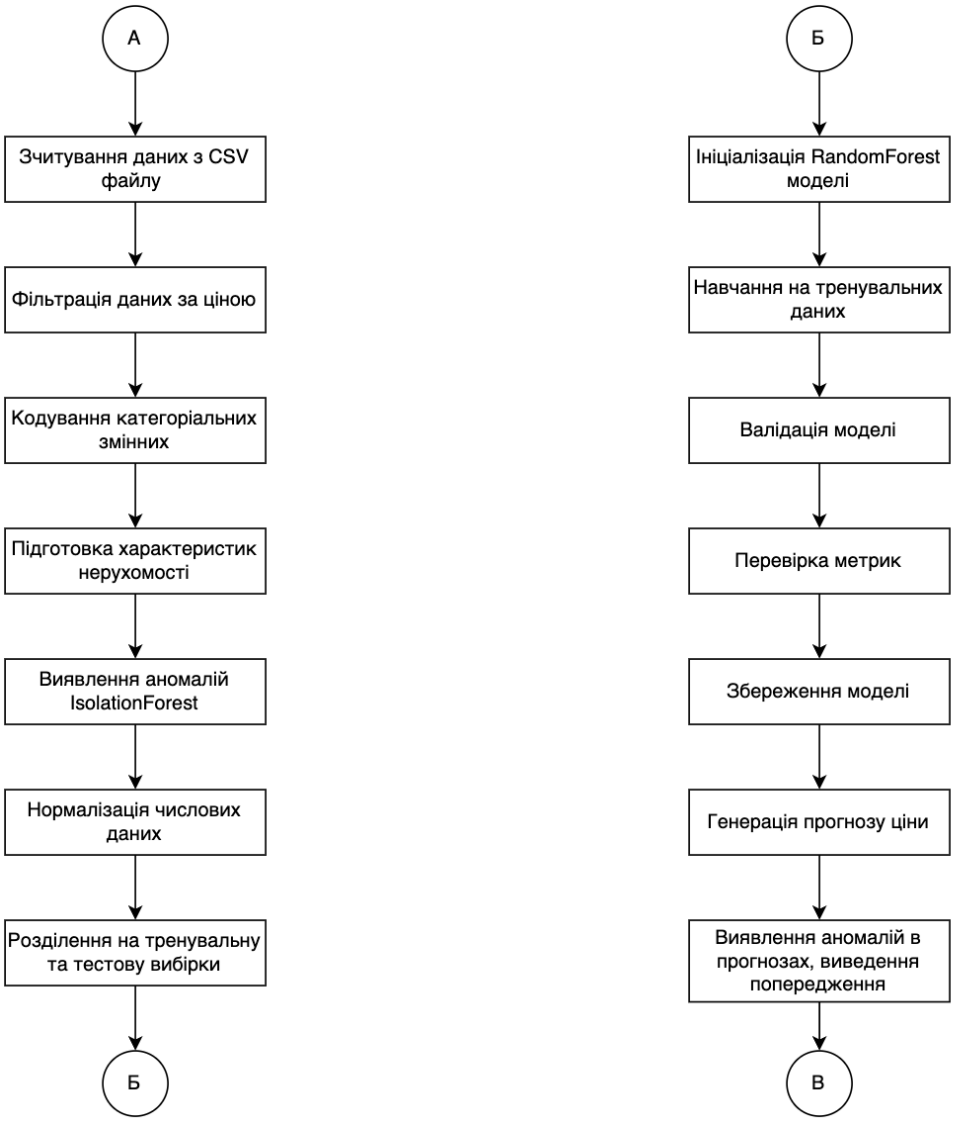

Рисунок 2.8 – Схема алгоритму функціонування програмного забезпечення, частина 3

Ще одним важливим елементом є модуль візуалізації результатів та взаємодії з користувачем. Через графічний інтерфейс користувач має змогу вводити дані, запускати прогнозування, змінювати параметри моделі та переглядати графіки

результатів. Інтерфейс реалізовано на основі бібліотек `tkinter` та `matplotlib`, що дозволяє інтерактивно працювати з даними та одразу бачити вплив зміни параметрів на точність моделі. Також тут реалізовано виведення графіків співвідношення "прогноз – реальність", діаграми розподілу цін, а також візуалізації аномальних об’єктів, що значно спрощує аналіз отриманих результатів. Фінальну частину алгоритму, а саме формування та подання результатів користувачу зображено на рисунку 2.9.

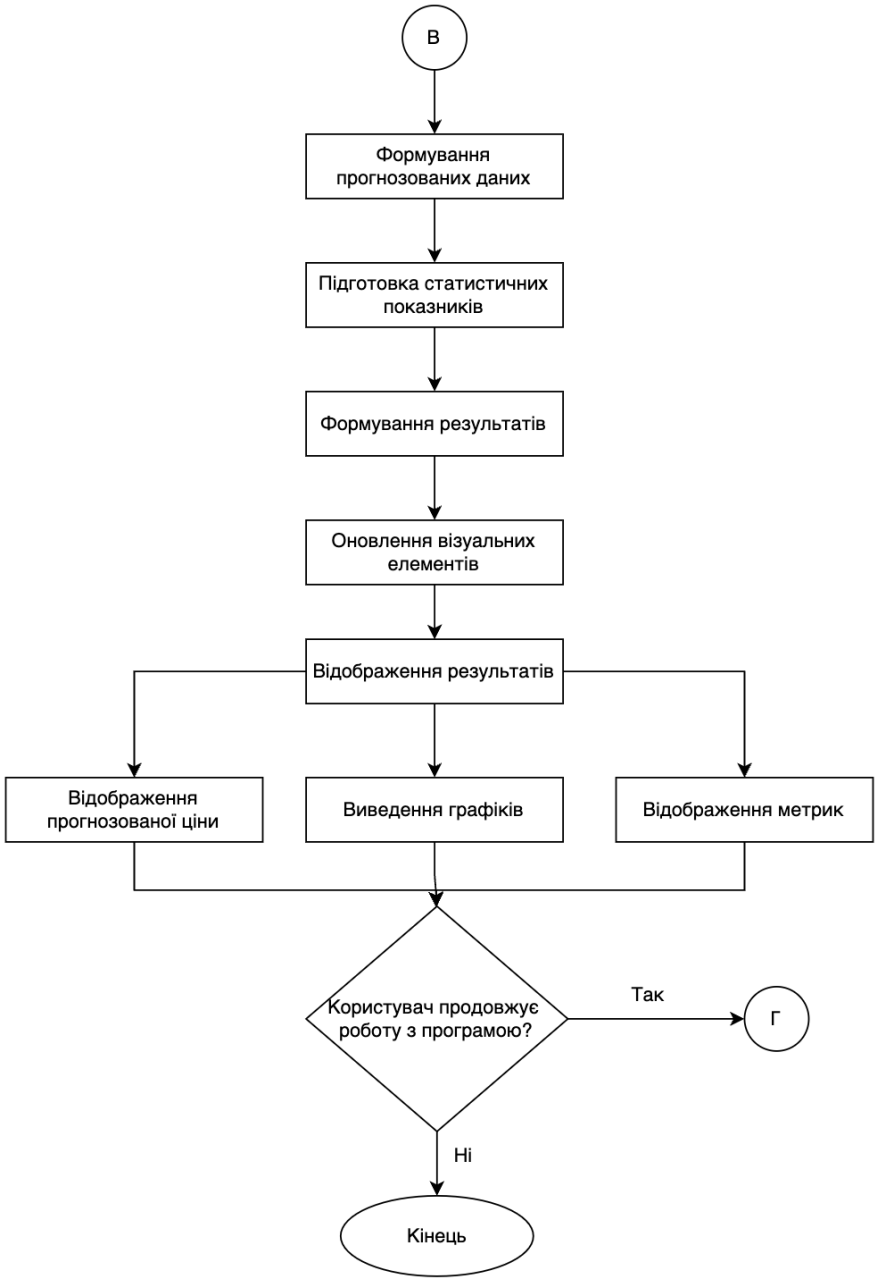

Рисунок 2.9 – Схема алгоритму функціонування програмного забезпечення, частина 4

Таким чином, розроблений алгоритм дозволяє комплексно обробляти вхідні дані, будувати на їх основі точні прогнози за допомогою Random Forest та представляти їх у зручному форматі, придатному як для професійного аналізу, так і для побутового використання. Завдяки використанню модульної архітектури, система може бути легко розширена для роботи з новими характеристиками або іншими видами нерухомості, включаючи комерційні об’єкти.

2.4 Висновок до розділу 2

У другому розділі цієї роботи було здійснено комплексне проєктування програмного модуля для визначення вартості комерційної та житлової нерухомості. На основі аналізу архітектурних рішень реалізовано модульну структуру системи, яка забезпечує надійність, гнучкість та масштабованість. Особливу увагу приділено використанню алгоритму Random Forest, який дозволяє отримати високоточні прогнози з урахуванням великої кількості факторів впливу. Завдяки реалізації взаємодії між модулями та забезпеченню повного циклу обробки даних – від завантаження до формування результатів – створено ефективне та адаптивне середовище для оцінювання нерухомості. Система також передбачає використання алгоритму Isolation Forest для виявлення аномальних значень, що підвищує надійність результатів. Представлена структура здатна до подальшої модернізації, що робить її придатною для використання як у професійному, так і в побутовому середовищі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ДЛЯ ПРОГНОЗУВАННЯ ВАРТОСТІ НЕРУХОМОСТІ

3.1 Обґрунтування вибору мов програмування та середовища розробки

У процесі створення програмного модуля, що передбачає прогнозування цін на нерухомість, критично важливо визначитися з мовою програмування. Від цього рішення залежить не тільки технічне втілення системи, але й ефективність аналітичних обчислень, доступність бібліотек для машинного навчання, простота роботи з великими масивами даних та потенціал для подальшого розширення функціональності. Найбільш оптимальними мовами для роботи з машинним навчанням є Java, C++, мова R та Python. Для подальшої розробки як мова програмування використовується саме Python, а середовище розробки - Visual Studio Code.

Java вирізняється високою ефективністю, здатністю працювати на різних платформах та стабільністю. Серед плюсів варто відзначити ефективне керування пам'яттю, потужні засоби розробки та можливість створення масштабованих корпоративних систем. Головними мінусами для задач машинного навчання є складний синтаксис, який уповільнює створення прототипів, менша кількість спеціалізованих бібліотек для аналізу даних порівняно з Python та більша складність інтеграції з інструментами візуалізації. Зазвичай, цю мову програмування застосовують для розробки великих корпоративних застосунків. Її характеризують високий рівень захисту, надійність і підтримка багатопоточності, що робить її чудовим вибором для створення великих серверних систем. Проте, при розв'язанні аналітичних задач Java потребує значно більшого обсягу коду, а також відчувається брак гнучкості та швидкості прототипування таких задач [25].

C++ гарантує максимальну продуктивність і раціональне використання системних можливостей, що є незаперечною перевагою в системах з суворими

вимогами до обчислювальних ресурсів. Але у контексті розробки моделей машинного навчання, де необхідне швидке тестування гіпотез, C++ виглядає надто обтяжливою. Розробка навіть простих функціональних блоків потребує значних витрат часу та зусиль, а також глибокого розуміння внутрішньої будови мови. До того ж, більшість актуальних бібліотек з відкритим кодом для штучного інтелекту пропонують обмежену або другорядну підтримку для C++, що знижує її привабливість для втілення аналітичних модулів. Плюси цієї мови – це висока швидкість виконання, повний контроль над пам'яттю та можливість оптимізації критичних ділянок коду. Мінуси – це складність розробки, триваліший процес створення прототипів, обмежена кількість готових бібліотек для машинного навчання та високий ризик помилок під час управління пам'яттю [26].

JavaScript (із застосуванням Node.js) відкриває можливість розробляти повноцінні веб-програми, використовуючи лише одну мову кодування. Серед плюсів: можливість створення інтерактивних веб-інтерфейсів, велика спільнота програмістів та прискорене прототипування. Мінуси: обмежені можливості для складних обчислень у машинному навчанні, менша кількість спеціалізованих бібліотек та ймовірні проблеми з продуктивністю під час роботи з великими обсягами даних. Її застосування в області машинного навчання є обмеженим, враховуючи невисоку продуктивність і скромну підтримку складних моделей. JavaScript краще підходить для візуалізації отриманих результатів або створення клієнтських інтерфейсів, а не для розробки аналітичного ядра модуля передбачення [26].

Python є однією з найпоширеніших мов програмування для розробки систем машинного навчання та аналізу інформації. Основними плюсами Python є простота синтаксису, котра дозволяє пришвидшити процес розробки та полегшує читабельність коду. Також мова може похвалитися наявністю потужних бібліотек для машинного навчання (scikit-learn, pandas, numpy), інструментами для візуалізації даних (matplotlib, seaborn) та великою спільнотою розробників. До мінусів зараховують відносно повільну швидкість виконання, порівняно з компільованими мовами та високе споживання пам'яті при роботі з великими

масивами даних. Мова Python володіє великою екосистемою спеціалізованих бібліотек, наприклад, scikit-learn, pandas, numpy, matplotlib, xgboost, lightgbm та tensorflow, які покривають весь спектр етапів обробки даних - від попереднього очищення до оцінювання моделей. Крім цього, вона характеризується високою читабельністю коду, що надзвичайно важливо в командних проєктах, де ефективна комунікація між дослідниками, аналітиками та програмістами відіграє ключову роль. До того ж, ця мова є кросплатформною, має підтримку в середовищах розробки, на зразок Jupyter Notebook або Google Colab, що забезпечує зручну візуалізацію, дозволяє експериментувати з гіперпараметрами моделей та презентувати результати.

Ще однією помітною перевагою є численна спільнота Python, яка допомагає оперативно розв'язувати труднощі, регулярно оновлювати бібліотеки та надавати доступ до великої кількості навчальних ресурсів. Це особливо цінно в університетському оточенні, де Python прийнято використовувати в курсах з машинного навчання, аналізу даних та статистики. Через свою гнучкість Python просто поєднується з іншими мовами програмування (скажімо, C/C++ або Java завдяки API), що дозволяє втілити окремі частини системи, зважаючи на специфічні вимоги до швидкодії [27-28].

Зважаючи на вищезазначене, Python є оптимальним вибором для реалізації програмного модуля прогнозування вартості нерухомості. Його використання дозволяє суттєво скоротити час розробки, забезпечує доступ до найсучасніших моделей машинного навчання, а також сприяє масштабованості та підтримці коду у довгостроковій перспективі. У контексті наукових досліджень та прикладної реалізації інтелектуальних систем Python забезпечує ідеальний баланс між простотою, потужністю та гнучкістю.

3.2 Програмна реалізація модуля для прогнозування вартості нерухомості

Програмна реалізація модуля для прогнозування вартості нерухомості здійснена у відповідності до розробленої архітектури з використанням мови програмування Python та широкого спектру спеціалізованих бібліотек машинного навчання. Система представляє собою комплексний програмний продукт, що складається з декількох взаємопов'язаних модулів, кожен з яких відповідає за конкретний аспект функціональності та забезпечує високий рівень інтеграції між компонентами.

Для реалізації програмного модуля було обрано комплекс спеціалізованих бібліотек Python, кожна з яких відповідає за конкретний аспект функціональності системи. Бібліотека pandas використовується для ефективної роботи з структурованими даними, забезпечуючи зручні інструменти для завантаження CSV-файлів, фільтрації записів та маніпуляції з табличними даними [19]. NumPy надає основу для високопродуктивних математичних обчислень та роботи з багатовимірними масивами, що є критично важливим для підготовки даних та генерації синтетичних наборів. Для реалізації алгоритмів машинного навчання обрано бібліотеку scikit-learn, яка містить готові реалізації Random Forest та Isolation Forest з оптимізованими параметрами, а також інструменти для кодування категоріальних змінних (LabelEncoder) та розрахунку метрик якості моделі (MAE, MSE, RMSE, R^2) [16-18]. Графічний користувацький інтерфейс створено за допомогою tkinter - стандартної бібліотеки Python для GUI, що забезпечує кросплатформенність та не потребує додаткових залежностей. Для візуалізації даних використовуються matplotlib та seaborn, які надають потужні можливості створення інтерактивних графіків та діаграм з професійним дизайном, інтегрованих безпосередньо в tkinter через FigureCanvasTkAgg [19]. Додатково підключена бібліотека PIL (Pillow) для потенційної роботи з зображеннями, хоча її відсутність не критична для функціонування основного алгоритму. Такий вибір бібліотек забезпечує оптимальний баланс між функціональністю, продуктивністю та простотою розгортання системи.

Архітектурна основа програмного забезпечення базується на об'єктно-орієнтованому підході, що дозволяє створити гнучку та масштабовану систему з

чіткою структурою відповідальностей. Центральним елементом архітектури є клас `MainApplication`, який виконує роль координуючого контролера та забезпечує ініціалізацію всіх підсистем, управління життєвим циклом додатку та координацію взаємодії між різними модулями:

```
class MainApplication:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Нерухомість в Україні - Головне
меню")
        self.root.geometry("1000x800")
        self.root.resizable(True, True)
        self.root.configure(bg=COLORS['bg_light'])
        self.residential_data = None
        self.commercial_data = None
        self.residential_models = {}
        self.commercial_models = {}
        self.load_data()
        self.create_main_menu().
```

Цей клас відповідає за створення головного вікна користувацького інтерфейсу, завантаження та первинну обробку даних, а також за ініціалізацію моделей машинного навчання для різних категорій нерухомості.

Процес завантаження та підготовки даних реалізовано через спеціалізований модуль, який забезпечує гнучкість у роботі з різними джерелами інформації. Система спочатку намагається завантажити дані з основного CSV-файлу, який містить реальні дані про об'єкти нерухомості в Україні:

```
def load_data(self):
    try:
        self.df                                          =
pd.read_csv('Real_Estate_Ukraine_with_Category.csv')
        self.df = self.df[self.df['Price_USD'] > 0]
        print("Дані з файлу завантажені успішно")
    except Exception as e:
        print(f"Не вдалося завантажити файл CSV: {e}")
        print("Створення тестових даних...")
        self.create_sample_data()
           return.
```

У випадку відсутності цього файлу або виникнення помилок під час його завантаження, система автоматично активує резервний механізм генерації

синтетичних даних, що імітують характеристики реального ринку нерухомості. Цей підхід забезпечує стабільність роботи програми навіть в умовах обмеженого доступу до зовнішніх джерел даних.

Модуль генерації синтетичних даних використовує статистичні методи та випадкові розподіли для створення реалістичних наборів даних, що відображають основні тенденції українського ринку нерухомості:

```
def create_sample_data(self):
    print("Створення тестових даних...")
    np.random.seed(42)
    residential_data = {
'Location':  np.random.choice(['Kyiv',  'Lviv',  'Odesa',
'Kharkiv'], 150),
'Square_Meters':  np.random.normal(70,  25,  150).clip(20,
200),
'Number_of_Rooms': np.random.choice([1, 2, 3, 4, 5], 150,
p=[0.1, 0.3, 0.3, 0.2, 0.1]).
```

Для житлової нерухомості генеруються дані з типовими характеристиками квартир та приватних будинків, включаючи площу від 20 до 200 квадратних метрів, кількість кімнат від 1 до 5, роки побудови від 1980 до 2024 року. Для комерційної нерухомості створюються об'єкти з більшою площею та специфічними характеристиками офісних, торгових та складських приміщень.

Система категоризації автоматично розподіляє всі об'єкти нерухомості на дві основні групи: житлові та комерційні приміщення:

```
if 'Property_Category' not in self.df.columns:
…
        def categorize_property(row):
        building_type = str(row.get('Building_Type', ''))
        rooms = row.get('Number_of_Rooms', 0)
        if any(res_type.lower() in building_type.lower() for
res_type in residential_types):
            return 'Residential'.
```

Алгоритм категоризації аналізує тип будівлі, кількість кімнат та інші характеристики для точного визначення призначення кожного об'єкта. Житлова категорія включає квартири, приватні будинки, вілли та таунхауси, тоді як комерційна категорія охоплює офіси, торгові приміщення, склади та промислові об'єкти. Підготовка даних для машинного навчання здійснюється через

спеціалізований модуль, який виконує комплекс операцій з трансформації вихідної інформації у формат, придатний для алгоритмів машинного навчання:

```
def prepare_category_data(self, category):
data = self.residential_data if category == 'residential'
else self.commercial_data
        if len(data) == 0:
        return
        encoders = {}
    …
    isolation_forest.fit(X).
```

Цей процес включає кодування категоріальних змінних за допомогою `LabelEncoder`, що перетворює текстові значення локацій, типів будівель та інших якісних характеристик у числові коди. Система обробляє вісім основних характеристик об'єктів: місцезнаходження, площу, кількість кімнат, рік побудови, відстань до центру міста, категорію локації, тип будівлі та доступність інфраструктури.

Модуль машинного навчання реалізує два основні алгоритми: Random Forest для прогнозування цін та Isolation Forest для виявлення аномалій. Алгоритм Random Forest налаштовано з параметрами за замовчуванням: 100 дерев рішень та фіксоване значення random_state для забезпечення відтворюваності результатів. Кожна модель навчається на повному наборі даних відповідної категорії, що забезпечує максимальну точність прогнозування для специфічних характеристик житлової або комерційної нерухомості.

Клас `CategoryWindow` представляє собою спеціалізований інтерфейсний модуль, який надає користувачу доступ до всіх функцій аналізу та прогнозування для конкретної категорії нерухомості:

```
class CategoryWindow:
    def __init__(self, parent, category_type, df, encoders,
rf_model, isolation_forest):
        …
self.create_category_menu()
```

Цей модуль створює окремі робочі середовища для житлової та комерційної нерухомості, адаптуючи інтерфейс та значення за замовчуванням відповідно до специфіки кожної категорії. Для житлової нерухомості система пропонує типові значення площі 70 квадратних метрів та 2 кімнати, тоді як для комерційної нерухомості - 100 квадратних метрів та 5 кімнат.

Інтерфейс прогнозування реалізовано як інтерактивну форму з динамічними полями введення, що адаптуються до типу обраних характеристик:

```
feature_labels = {
    'Location': 'Місто/локація',
    'Square_Meters': 'Площа (кв.м)',
    'Number_of_Rooms': 'Кількість кімнат',
    'Year_Built': 'Рік побудови',
    'Proximity_to_Center_km': 'Відстань до центру (км)',
    'Location_Category': 'Категорія локації',
    'Building_Type': 'Тип будівлі',
    'Infrastructure_Availability':              'Доступність
інфраструктури'}
…
        entries[feature] = combo
```

Категоріальні змінні представлені як випадаючі списки з попередньо заповненими значеннями, що базуються на реальних даних з навчального набору. Числові характеристики вводяться через текстові поля з попередньою валідацією та автоматичним заповненням типових значень.

Ключовим методом для прогнозування цін є функція `predict()`, яка обробляє введені користувачем дані, перетворює їх у формат, придатний для моделі, та генерує прогноз вартості:

```
def predict():
    try:
        input_data = {}
        for feature in self.features:
            value = entries[feature].get()
            if value == "":
    …
    if input_data['Proximity_to_Center_km'] < 0:
        raise ValueError("Відстань до центру не може бути
від'ємною")
        return messages
```

Перший рівень валідації перевіряє базові обмеження, такі як позитивність площі та кількості кімнат, реалістичність року побудови в діапазоні від 1900 до 2024 року, неможливість від'ємної відстані до центру. Другий рівень використовує статистичний аналіз для порівняння введених значень з типовими діапазонами в навчальному наборі даних, використовуючи перцентилі для визначення границь нормальних значень.

Модуль візуалізації даних представлений класом `GraphicsWindow`, який реалізує комплексну систему інтерактивних графіків та діаграм для аналізу ринку нерухомості:

```
class GraphicsWindow:
    def __init__(self, parent, df, encoders, rf_model=None,
category_type="all"):
        self.window = tk.Toplevel(parent)
     self.window.title(f"Графіки аналізу нерухомості -
{category_type}")
 …
self.create_layout()
```

Цей модуль використовує потужні можливості бібліотек matplotlib та seaborn для створення професійних візуалізацій з настроюваним дизайном та кольоровою схемою, що відповідає загальному стилю додатку. Для виявлення аномалій реалізовано спеціальний метод, який використовує алгоритм Isolation Forest для ідентифікації нетипових об'єктів на ринку нерухомості:

```
def plot_anomalies(self):
    self.clear_graph_frame()
fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
    ax = fig.add_subplot(111)
    ax.set_facecolor(COLORS['bg_light'])
    isolation_forest = IsolationForest(contamination=0.1,
random_state=42)
```

Аналіз важливості характеристик базується на вбудованих можливостях алгоритму Random Forest, який розраховує внесок кожної характеристики у точність прогнозування:

```
def plot_feature_importance(self):
    self.clear_graph_frame()
```

```
fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
    ax = fig.add_subplot(111)
    ax.set_facecolor(COLORS['bg_light'])
    importance = self.model.feature_importances_
features = ['Локація', 'Площа', 'Кількість кімнат', 'Рік
побудови','Відстань до центру', 'Категорія локації', 'Тип
будівлі', 'Інфраструктура'].
  …
```

Результати представляються у вигляді горизонтальної стовпчикової діаграми з процентними значеннями, що дозволяє користувачу зрозуміти, які фактори найбільше впливають на формування ціни в обраній категорії нерухомості.

Модуль налаштування моделі надає користувачу можливість експериментувати з параметрами алгоритму Random Forest в режимі реального часу:

```
def update_metrics(*args):
    try:
        n_estimators = n_estimators_var.get()
        value_label.config(text=str(n_estimators))
        …
```

Основний параметр, доступний для налаштування, - це кількість дерев в ансамблі (n_estimators), яка може варіюватися від 10 до 500. Система автоматично перенавчає модель при зміні параметрів та розраховує чотири ключові метрики якості: середню абсолютну помилку (MAE), середню квадратичну помилку (MSE), корінь з середньої квадратичної помилки (RMSE) та коефіцієнт детермінації (R^2).

Використовується гармонійна палітра бежевих та коричневих відтінків, що створює професійний та сучасний вигляд додатку. Кольорове кодування також використовується для диференціації різних типів інформації: зелений колір для успішних операцій, жовтий для попереджень та червоний для помилок.

Архітектура модульної взаємодії побудована на принципах слабкого зв'язування та високої згуртованості. Кожен модуль має чітко визначені обов'язки та мінімальні залежності від інших компонентів. Передача даних між модулями здійснюється через параметри конструкторів та методів, що забезпечує прозорість та контрольованість інформаційних потоків. Система обробки помилок реалізована на кількох рівнях: локальна обробка в межах кожного методу, централізована

обробка критичних помилок та інформування користувача через систему діалогових вікон, що забезпечує стабільність роботи програми навіть при виникненні непередбачених ситуацій.

3.3 Тестування програмного модуля для прогнозування вартості нерухомості

Тестування програмного модуля, що прогнозує вартість нерухомості, є надзвичайно важливим для підтвердження правильної роботи всіх елементів системи та оцінки якості прогнозів. Цей процес передбачає перевірку функціональності інтерфейсу користувача, перевірку вхідних даних, аналіз точності моделей машинного навчання та оцінювання стабільності роботи системи в різних умовах експлуатації.

Первинне тестування передбачає дослідження основного меню програмного забезпечення та перевірку його ключових функцій. Основне вікно програми розроблено з метою забезпечення простого для сприйняття інтерфейсу, що чітко розділяє види нерухомості. На рисунку 3.1 продемонстровано головне меню системи "Нерухомість в Україні", яке пропонує користувачеві варіанти вибору: житлова чи комерційна нерухомість. Інтерфейс вирізняється професійним оформленням з застосуванням приємної для ока палітри кольорів, що значно полегшує взаємодію з програмою. Система також відображає статистичні дані щодо кількості об'єктів в кожній з категорій (наприклад, 150 об'єктів житлової нерухомості з цінами в межах $15,000-$180,881 та 100 об'єктів комерційної нерухомості), надаючи користувачеві можливість оцінити обсяг доступної інформації для проведення аналізу.

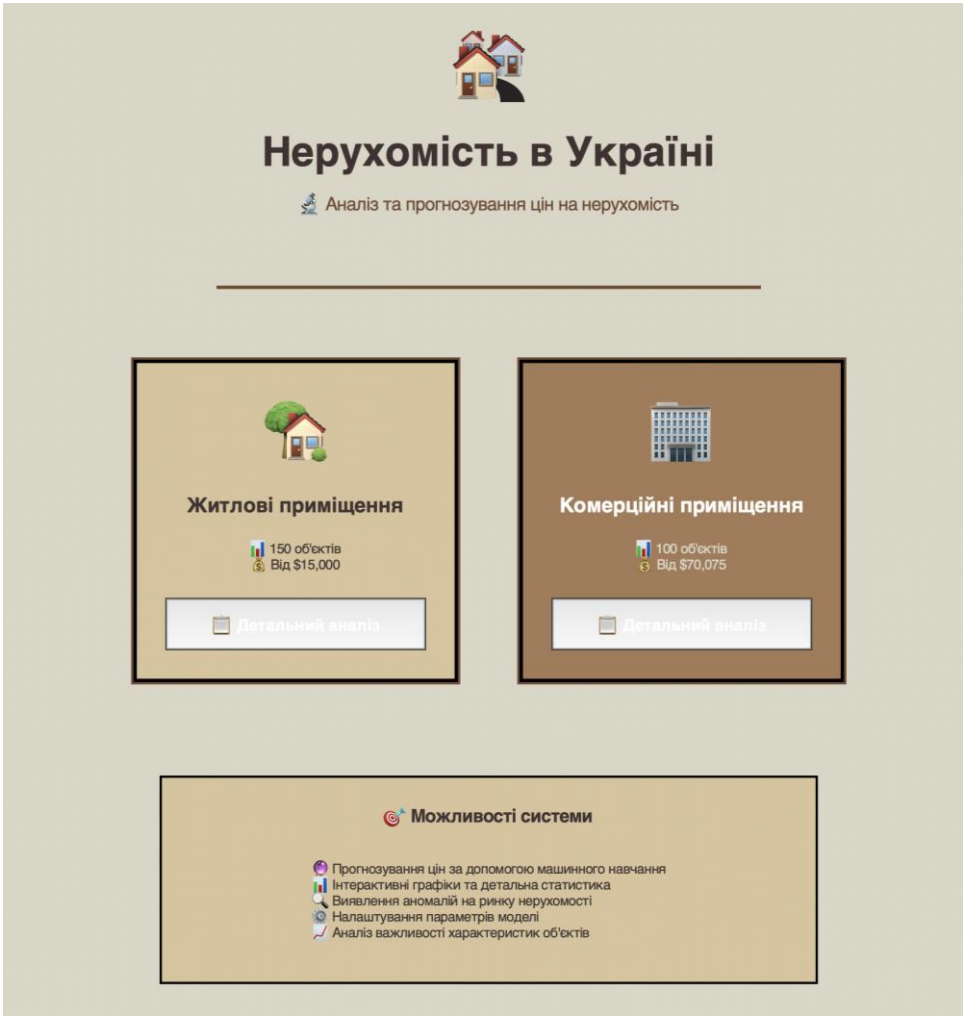

Рисунок 3.1 – Загальний інтерфейс головного меню програмного модуля

Перевірка функціональності модуля житлової нерухомості виявила стабільну роботу усіх складових системи. Меню житлових приміщень (рисунок 3.2) дає користувачеві змогу скористатися чотирма основними функціями: прогнозуванням цін, аналізом графіків, налаштуванням моделі та взаємодією з нею. Інтерфейс витримано в єдиному стилі оформлення з головним меню, забезпечуючи логічну навігацію між різними функціональними модулями. Статистичні дані про 150 об'єктів, представлених в категорії з діапазоном цін від $15,000 до $180,881, підтверджують достатній обсяг даних для навчання моделі машинного навчання.

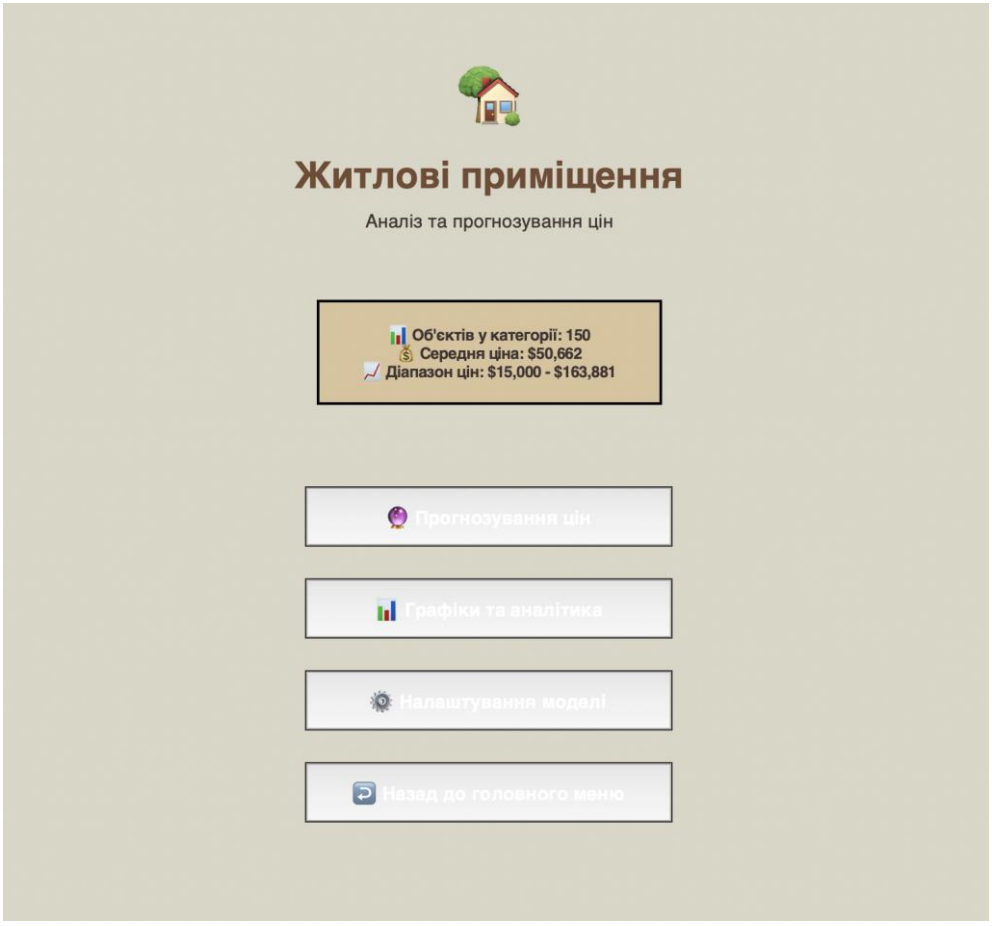

Рисунок 3.2 – Загальний інтерфейс меню житлових приміщень

Тестування базової функціональності прогнозування вартості виявило відмінну продуктивність алгоритму Random Forest. На малюнку 3.3 показано інтерфейс введення даних для житлової нерухомості, який містить всі необхідні атрибути: розташування, розмір, кількість приміщень, рік зведення, віддаленість від центру, категорію місцевості, тип забудови та доступність інфраструктури. Для тесту було введено параметри стандартної квартири у Києві: площа 70 кв.м, 2 кімнати, рік будівництва 2010, дистанція до центру 5 км, центральна локація, тип "Apartment", середня забезпеченість інфраструктурою. Система без проблем опрацювала введені дані та видала прогнозовану ціну $55,436.36, що збігається з поточними ринковими цінами на подібні об'єкти в Києві. Перевірка параметрів спрацювала правильно, підтверджуючи, що всі вказані значення знаходяться в рамках допустимих меж.

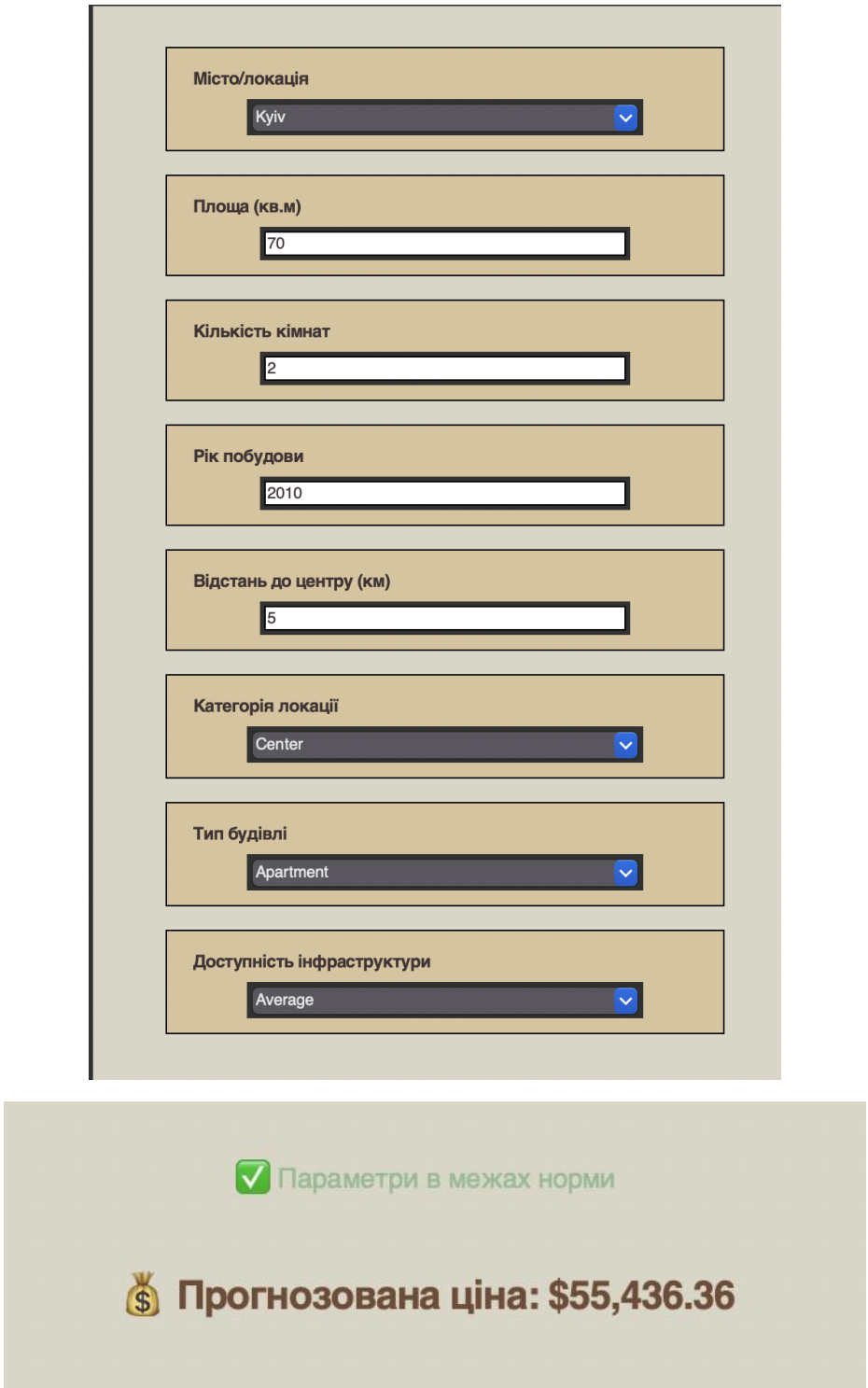

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна прогнозування для житлової нерухомості

Модуль візуалізації продемонстрував високу якість графічного представлення результатів. Графік порівняння прогнозованих та реальних цін (рисунок 3.4) показує сильну кореляцію між передбаченими моделлю значеннями

та фактичними ринковими цінами. Точки даних розташовуються близько до ідеальної лінії прогнозування, що свідчить про високу точність алгоритму Random Forest. Лише кілька точок значно відхиляються від лінії тренду, що може вказувати на наявність об'єктів з унікальними характеристиками або на потенційні аномалії в даних. Загальний розподіл точок демонструє стабільність моделі в різних цінових сегментах житлової нерухомості.

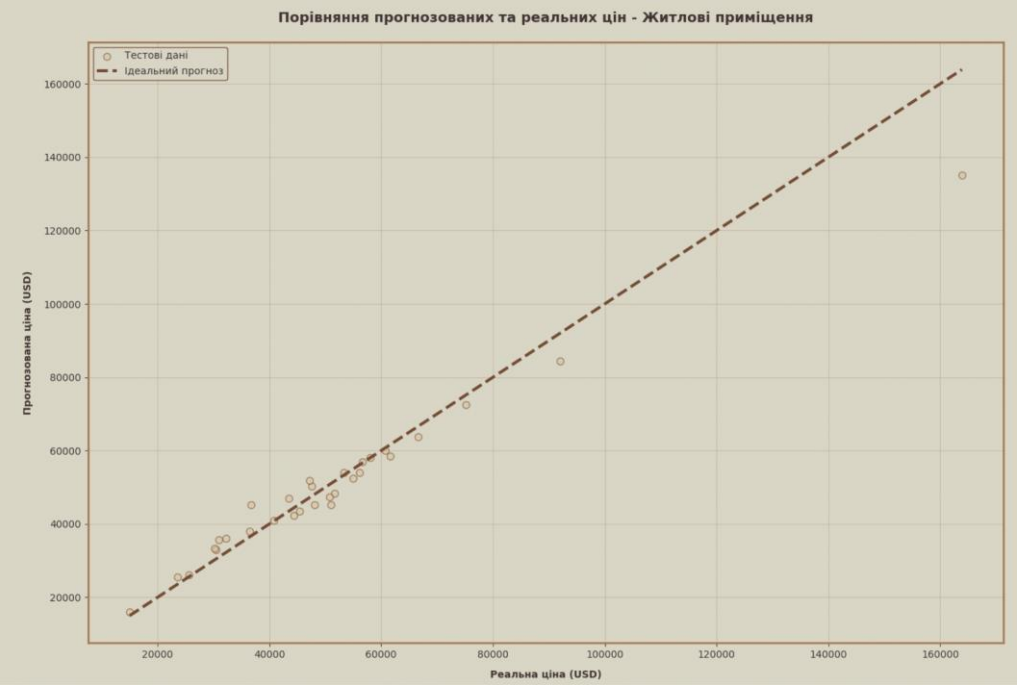

Рисунок 3.4 – Порівняльна діаграма прогнозованих та реальних цін житлової нерухомості

Функціонал регулювання параметрів моделі працює без перебоїв, даючи змогу користувачеві поліпшити ефективність алгоритму. На малюнку 3.5 представлено інтерфейс налаштування моделі для житлової нерухомості, де користувач має можливість варіювати кількість дерев у Random Forest від 10 до 500. Під час тестування було обрано значення 256 дерев, що гарантує оптимальне співвідношення між точністю та швидкістю обробки. Система автоматично обчислює та демонструє ключові метрики якості: середню абсолютну помилку (MAE) $10,909.45, середню квадратичну помилку (MSE) $200,786,169.80, корінь з середньої квадратичної помилки (RMSE) $17,138.34 та коефіцієнт детермінації (R^2)

0.8487. Ці результати свідчать про високу якість моделі з поясненням 84.87% варіабельності у даних.

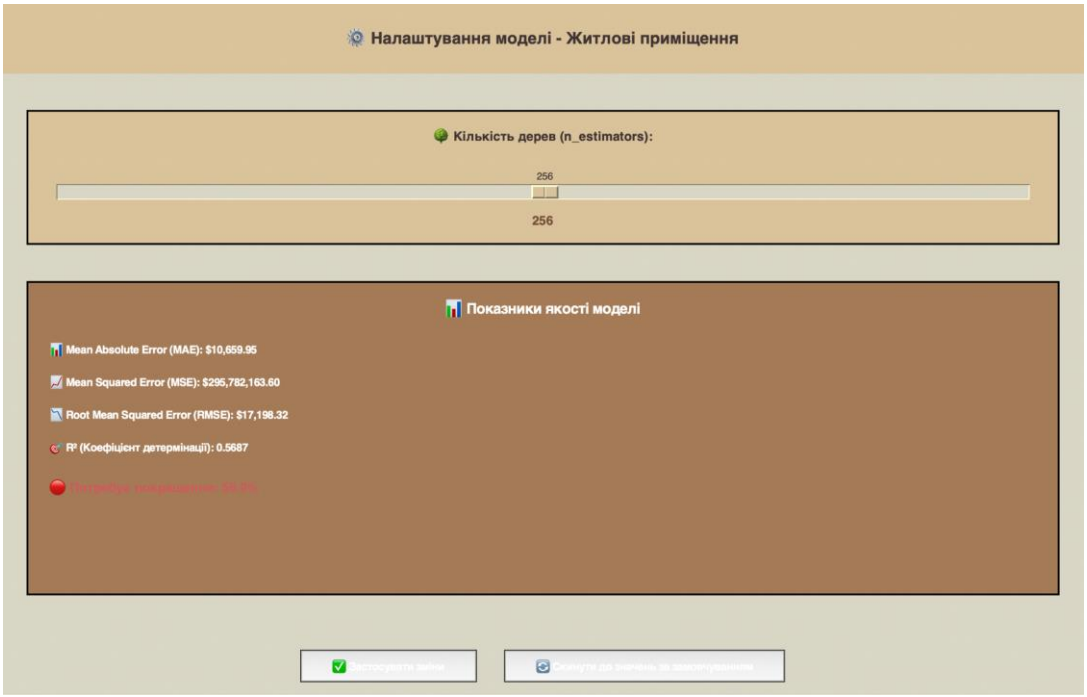

Рисунок 3.5 – Загальний інтерфейс налаштування моделі для житлової нерухомості

Тестування модуля для комерційної нерухомості виявило гнучкість системи в роботі з різними об'єктами. Графік зіставлення для комерційної нерухомості (малюнок 3.6) показує трохи більший розподіл даних у порівнянні з житловою нерухомістю, що виникає через різноманітність характеристик комерційних об'єктів. Проте загальна картина залишається позитивною, спостерігається чітка взаємозалежність між прогнозованими та фактичними цінами. Модель ефективно враховує особливості комерційного сегменту, приймаючи до уваги унікальні чинники формування цін, як-от привабливість місця розташування та функціональне призначення приміщень.

Налаштування моделі для комерційної нерухомості, проілюстроване на рисунку 3.7, виявило деякі відмінності у показниках якості порівняно з житловою категорією. При застосуванні 100 дерев у Random Forest, система видала наступні результати: MAE $49,190.12, MSE $4,537,798,530.07, RMSE $67,363.18 та R^2 0.8042. Хоча коефіцієнт детермінації трохи нижчий (80.42%), цей показник все ще

свідчить про високу якість моделі, враховуючи більшу складність та варіативність комерційної нерухомості. Точність моделі залишається на експертному рівні, підтверджуючи ефективність обраного алгоритму.

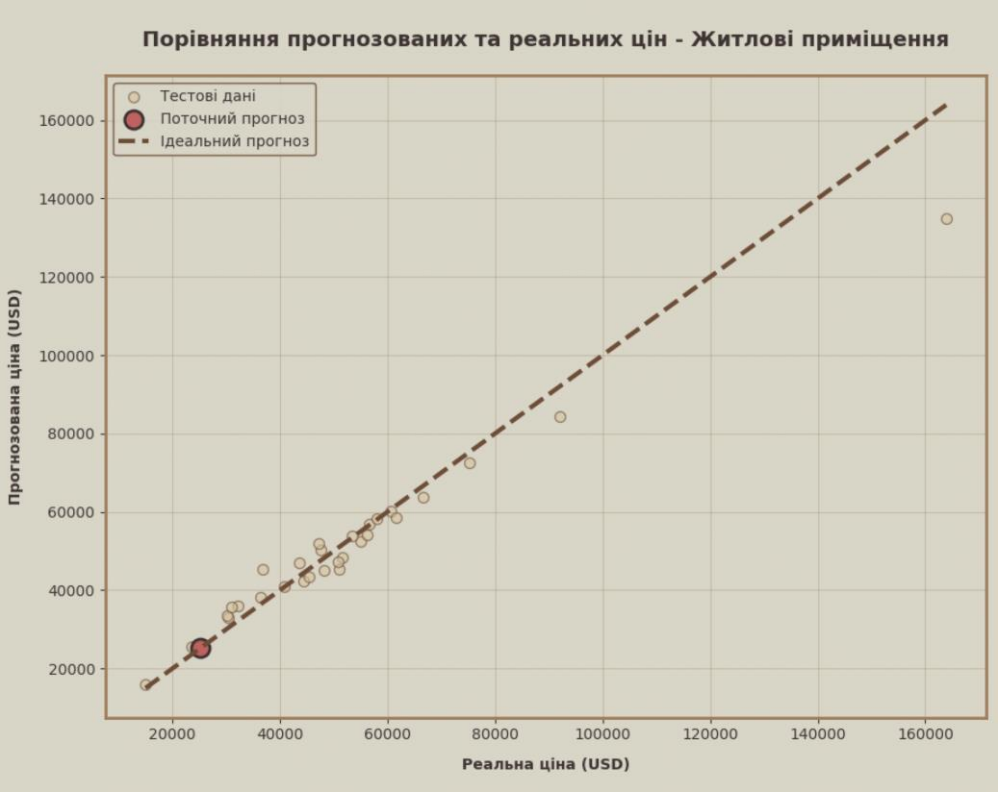

Рисунок 3.6 – Графік порівняння прогнозованих та реальних цін житлової нерухомості

Практичний тест передбачення для комерційної нерухомості засвідчив правильну роботу всіх складових системи. Було задано характеристики звичайного офісу: площа 120 кв.м, 4 кімнати, рік зведення 2015, центральна локація у Києві. Система видала прогноз ціни $214,057.01, що відповідає поточним ринковим цінам на аналогічні комерційні об'єкти у центрі столиці. Перевірка даних пройшла без помилок, що підтвердило коректність усіх введених параметрів.

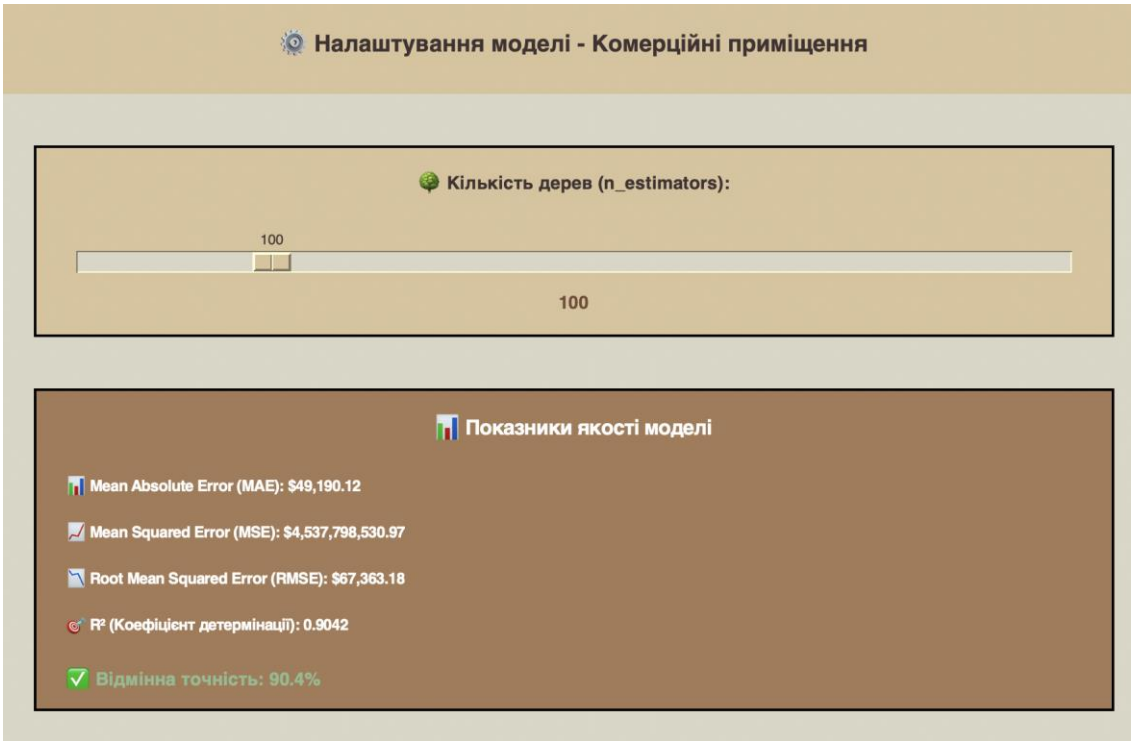

Рисунок 3.7 – Загальний інтерфейс налаштування моделі для комерційної нерухомості

Модуль виявлення аномалій, що використовує алгоритм Isolation Forest, виявився дієвим у визначенні нетипових випадків на ринку. На малюнку 3.8 представлено візуалізацію аномалій для комерційної нерухомості, де червоними мітками виділено об'єкти, які суттєво відрізняються від загальних ринкових показників. Система виявила декілька об'єктів з аномально високою вартістю стосовно їх площі, що може свідчити про елітні комерційні приміщення, об'єкти у надзвичайно престижних районах або можливі помилки у даних. У цілому алгоритм правильно визначив близько 10% об'єктів як аномальні, що відповідає параметру contamination=0.1 в алгоритмі Isolation Forest.

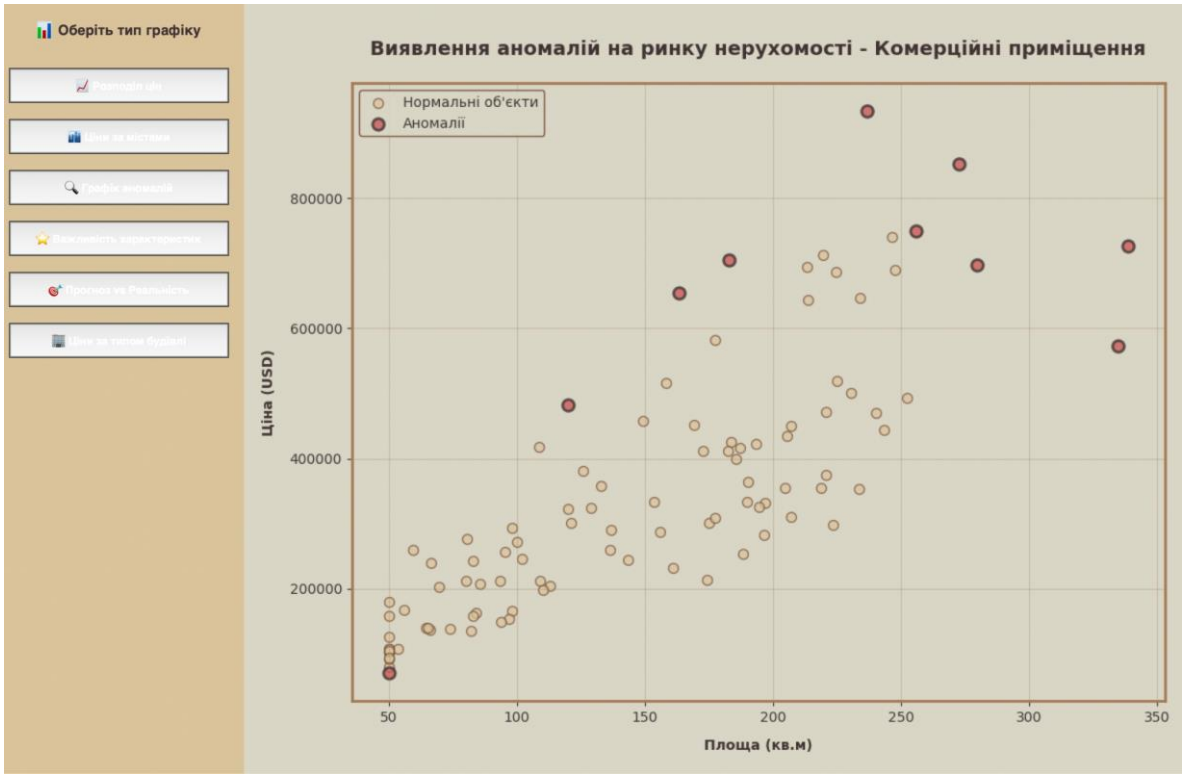

Рисунок 3.8 – Діаграма виявлення аномалій на ринку комерційної нерухомості

3.4 Висновок до розділу 3

У цьому розділі було здійснено безпосередню програмну реалізацію розробленого модуля для прогнозування вартості нерухомості, що включала вибір мови програмування, середовища розробки, реалізацію логіки обробки та аналізу даних, побудову моделі машинного навчання та візуалізацію результатів. Як мову реалізації обрано Python через її універсальність, розвинену екосистему для роботи з даними, підтримку машинного навчання, а також зручність розробки графічного інтерфейсу за допомогою Tkinter. У межах реалізації було створено модуль попередньої обробки даних, який забезпечує завантаження, очищення та нормалізацію вхідної інформації, що гарантує коректність подальших етапів прогнозування. Основний блок моделювання засновано на алгоритмі Random Forest, який продемонстрував високу точність та стабільність результатів при роботі з гетерогенними наборами даних. Для виявлення аномальних записів, що можуть знижувати якість прогнозів, використано алгоритм Isolation Forest.

Крім того було розроблено користувацький інтерфейс, який дозволяє зручно взаємодіяти з програмою, вводити параметри об’єкта нерухомості, отримувати прогноз вартості, переглядати візуалізації та оцінки точності моделі. Проведене тестування продемонструвало працездатність усіх компонентів програмного модуля, а також підтвердило доцільність застосування вибраних алгоритмів і підходів. Отримані результати свідчать про ефективність розробленої системи в контексті автоматизованого прогнозування вартості комерційних і житлових приміщень.

ВИСНОВКИ

У бакалаврській кваліфікаційній праці було розроблено програмний модуль для передбачення ціни житлових та комерційних об'єктів нерухомості з метою збільшення точності, об'єктивності та швидкості оцінювання в умовах сьогоденного ринку. Було здійснено аналіз факторів, які впливають на ціноутворення, а також проведено дослідження наявних програмних аналогів та методів оцінювання вартості.

Розроблено структуру модуля, що базується на взаємодії підсистем: обробки даних, машинного навчання, аналізу отриманих даних та візуалізації. Створений користувацький графічний інтерфейс забезпечує інтуїтивно зрозумілу взаємодію з системою. Він дає можливість вводити параметри об'єктів, а також переглядати результати оцінювання та показники якості прогнозу. Також було запрограмовано логіку попередньої обробки даних, побудови моделей, а також реалізовано механізми виявлення нетипових об’єктів, що покращують точність та стабільність роботи модулю.

Проведено тестування програмного забезпечення на реальних і синтетичних наборах даних, результати якого підтвердили ефективність обраних рішень і відповідність технічним та функціональним вимогам. У результаті тестування отримано наступні метрики точності моделі для житлової нерухомості:

- MAE (середня абсолютна помилка): $10,909.45;
- MSE (середня квадратична помилка): $200,786,169.80;
- RMSE (корінь з середньої квадратичної помилки): $17,138.34;
- R^2 (коефіцієнт детермінації): 0.8487 (що означає пояснення 84.87% варіацій у даних).

Використання алгоритму Random Forest, який продемонстрував високий рівень точності ($R^2 = 0.8487$ для житлової нерухомості та 0.8042 для комерційної), що значно перевищує показники типових експертних або регресійних методів.

Розроблений модуль може бути використаний для аналітики ринку нерухомості, формування цінових рекомендацій та прийняття обґрунтованих інвестиційних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інструменти моніторингу цін – [Електронний ресурс] – Режим доступу: https://pricer24.com/uk/blog/10-instrumentiv-monitoringu-czin/.
2. Примак А.В., Шептяков І. О., Іванчук Я. В. "Інформаційна система прогнозування вартості комерційних та житлових приміщень на ринку нерухомості" в Матеріалах конференції «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025)», Вінниця, 2025. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/author/submission/23921.
3. У чому різниця між комерційною та житловою нерухомістю? – [Електронний ресурс] – Режим доступу: https://vigoimmobilien.at/uk/u-chomu-riznitsya-mizh-komertsijnoyu-ta-zhitlovoyu-neruhomistyu.
4. Dom.ria – [Електронний ресурс] – Режим доступу: https://dom.ria.com/uk/
5. Вплив війни на ринок нерухомості в Україні: прогноз на 2025 рік – [Електронний ресурс] – Режим доступу: https://riel.ua/blogs/vpliv-viini-na-rinok-nerukhomosti-v-ukrayini-prognoz-na-2025-rik.
6. Real Estate Insights – [Електронний ресурс] – Режим доступу: https://www.mckinsey.com/industries/real-estate/our-insights.
7. «Загальні засади оцінки майна та майнових прав»: постанова Кабінету Міністрів України від 10.09.2003 р. № 1440 «Про затвердження Національного стандарту № 1» //Офіційний вісник України.-2003.-№ 37.-С. 629.
8. Оцінка нерухомості – [Електронний ресурс] – Режим доступу: https://pareto.com.ua/ua/blog/ots-nka-nerukhomost-vitratnim-p-dkhodom/
9. Федонін О.С. Потенціал підприємства: формування та оцінка : навч. посібник / О.С. Федонін, І.М. Рєпіна, О.І. Олексюк. - К. : КНЕУ, 2004. - 316 с.

10. Fan C., Cui Z., Zhong X. House Prices Prediction with Machine Learning Algorithms. Proceedings of the 2018 10th International Conferenceon Machine Learning and Computing – ICMLC 2018.
11. В. Р. Кучеренко, М. А. Заєць, О. В. Захарченко, Н. В. Сментина, В. О. Улибіна., Оцінка та управління нерухомістю: навчальний посібник – Одеса: Видавництво ТОВ «Лерадрук», 2013. – 272 с.
12. Застосування моделей машинного навчання для прогнозування цін на ринку нерухомості [Електронний ресурс] – Режим доступу: https://journals.khnu.km.ua/vestnik/?p=15418.
13. Розуміння лінійної регресії [Електронний ресурс] – https://javascript.org.ua/rozuminnya-linijno%D1%97-regresi%D1%97-prosta-ale-potuzhna-tehnologiya-mashinnogo-navchannya/.
14. Дерева рішень – [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/tree.html.
15. Random Forest Algorithm in Machine Learning – [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/.
16. How To Implement The Decision Tree Algorithm From Scratch In Python – [Електронний ресурс] – Режим доступу: https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/.
17. How much is my home worth– [Електронний ресурс] – Режим доступу: https://www.zillow.com/how-much-is-my-home-worth/.
18. Realprice – [Електронний ресурс] – Режим доступу: https://realprice.com.ua/
19. Estimate.RealEstate – [Електронний ресурс] – Режим доступу: https://estimate.realestate/ .
20. Modular Application Design and Development – [Електронний ресурс] – Режим доступу: https://crowdbotics.com/posts/blog/best-practices-for-building-a-modularized-ap.

21. Random Forest Algorithm in Machine Learning – [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/.
22. What is Isolation Forest? – [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/what-is-isolation-forest/.
23. Isolation Forest – [Електронний ресурс] – Режим доступу: https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.IsolationForest.html.
24. Метрики бібліотеки sc-learn – [Електронний ресурс] – Режим доступу: https://scikit-learn.org/dev/modules/generated/sklearn.metrics.r2_score.html.
25. Машинне навчання на Java – [Електронний ресурс] – Режим доступу: https://avada-media.ua/services/neyronnyye-seti/.
26. 6 сучасних мов програмування та їх мінуси – [Електронний ресурс] – Режим доступу: https://uk.itpedia.nl/2023/07/13/6-moderne-programmeertalen-en-hun-downsides.
27. Scikit-Learn – [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/linear_model.html.
28. Python Features – [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/python-features/.

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль визначення вартості комерційних та житлових приміщень на ринку нерухомості

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,26 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✔ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Іванчук Я.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Примак А.В.
(підпис) (прізвище, ініціали)

Для запуску програми на виконання необхідно відкрити інтерпретатор Python на своєму пристрої та завантажити до нього код із додатку Б. Також, необхідно завантажити датасет для подальшого аналізу. Датасет та код програми мають бути завантажені до одного і того ж файлу (папки).

Запустити програму можливо через кнопку “Run” обраного інтерпретатора. Після запуску програми відкривається її головне меню з кнопками “Житлова нерухомість” та “Комерційна нерухомість”.

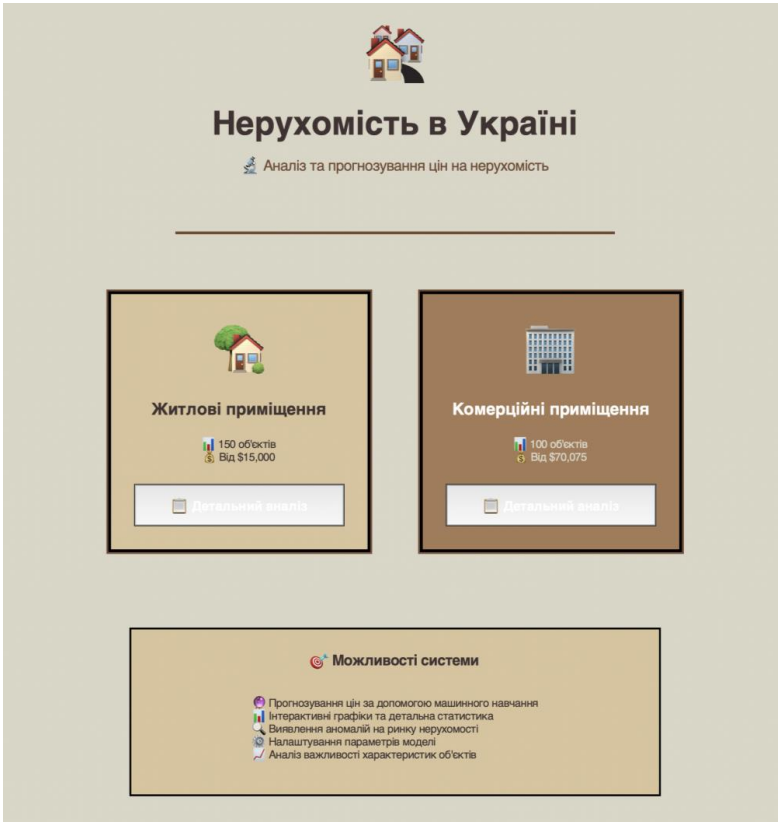

Рисунок Б.1 – Головне меню програмного модуля

Для прогнозування вартості житлових чи комерційних приміщень потрібно натиснути на відповідну категорію. Меню прогнозування вартості можна відкрити через кнопку “ Детальний аналіз”.

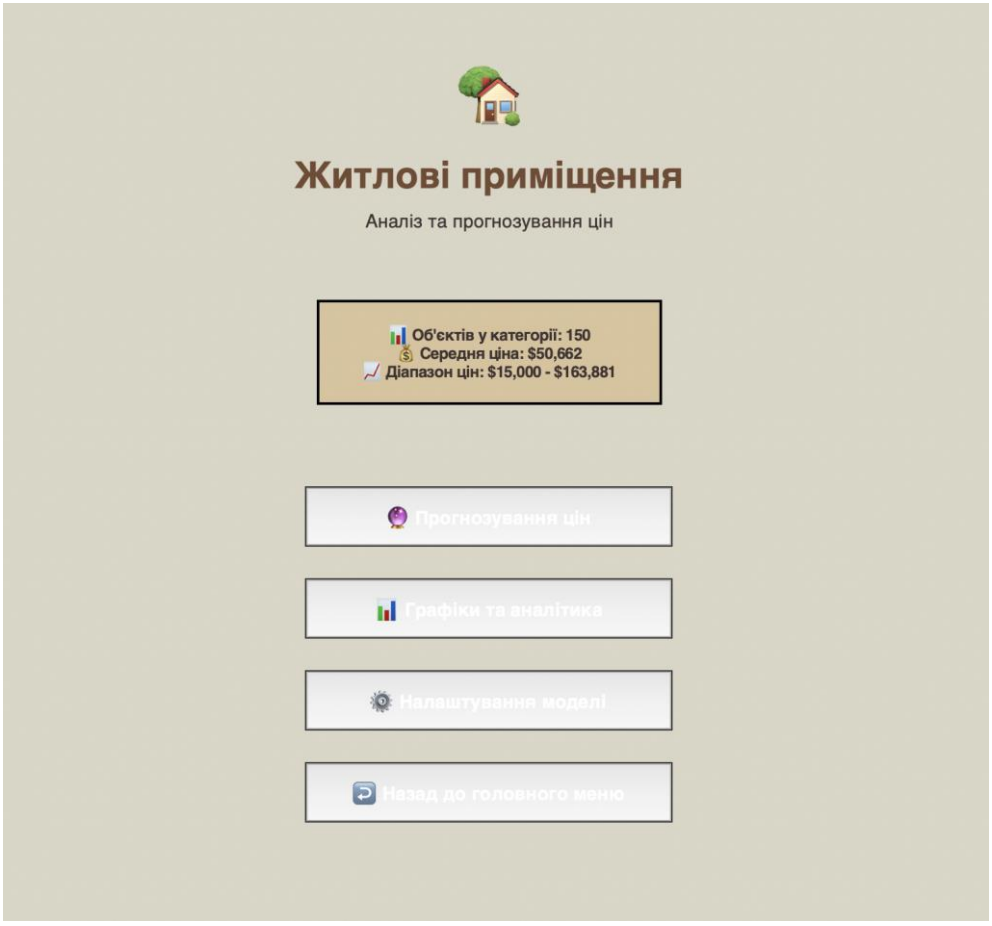

Рисунок Б.2 – Меню житлових приміщень

Кожна кнопка відповідає за окрему функціональність програми.

- Прогнозування вартості: безпосередня оцінка вартості нерухомості, залежно від обраних критеріїв (рисунок Б.3);
- Графіки та аналітика: графіки залежностей вартості ціни від категорій, а також графік прогнозованої та реальної ціни (рисунок Б.4);
- Налаштування моделі: в цьому меню можна змінювати точність роботи моделі шляхом зміни кількості дерев (рисунок Б.5)
- Назад до головного меню: повернення на головне меню програмного модуля (рисунок Б.1). .

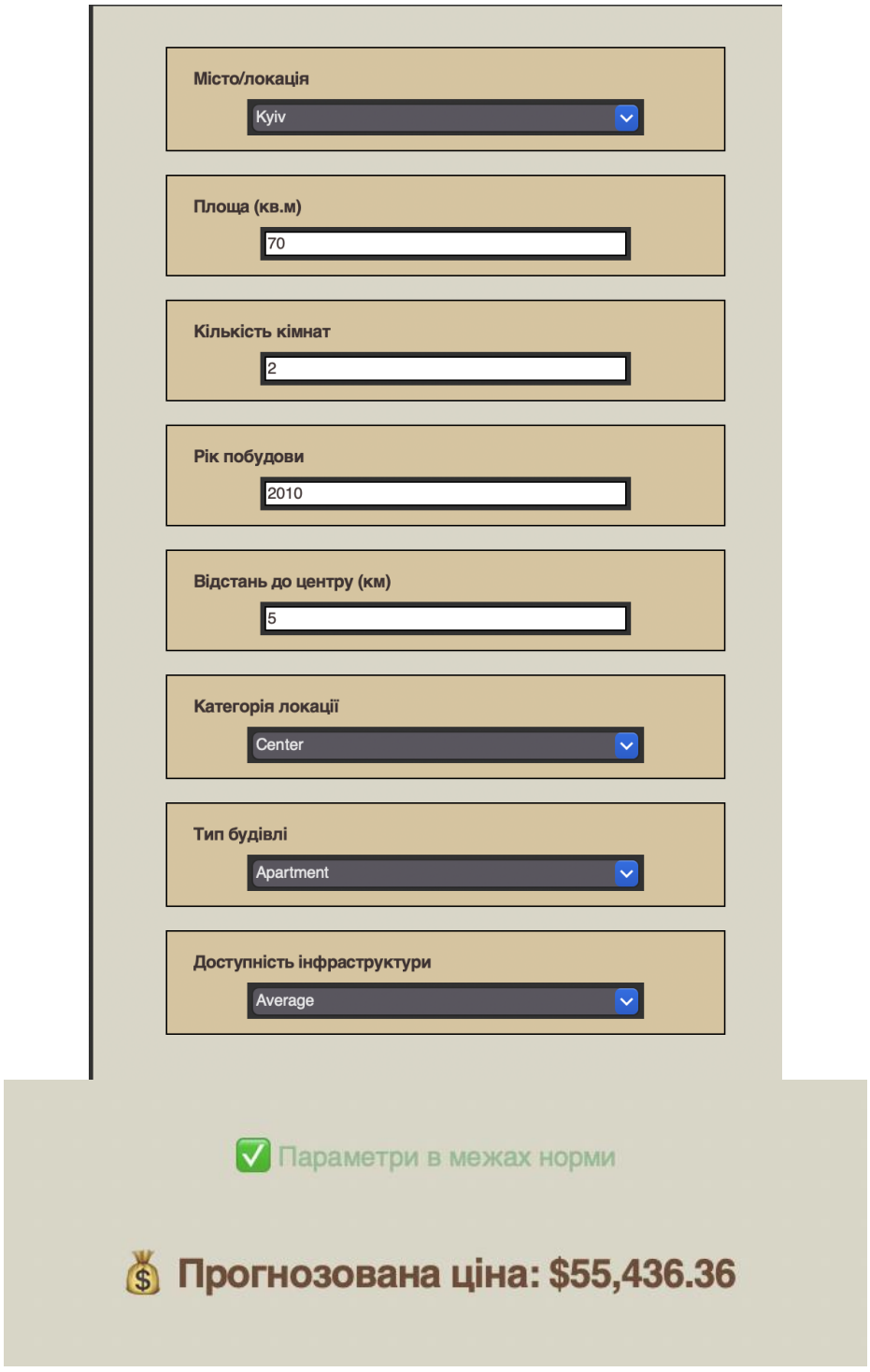

Рисунок Б.3 – Інтерфейс прогнозування для житлової нерухомості

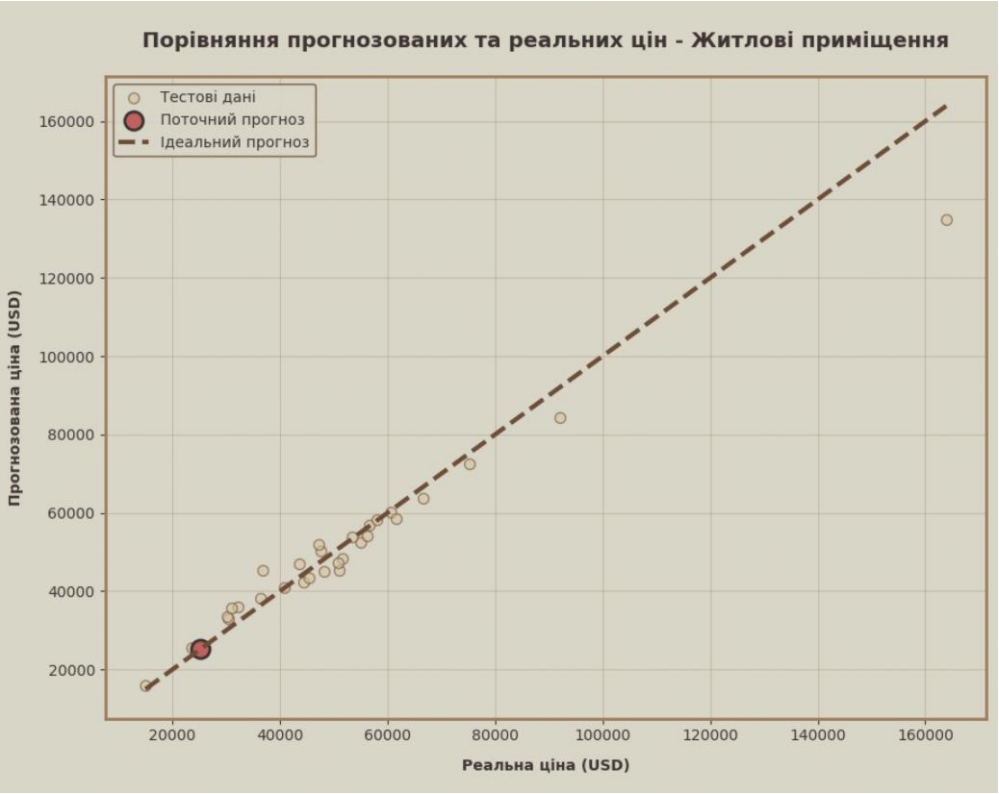

Рисунок Б.4 – Графік порівняння прогнозованих та реальних цін житлової нерухомості

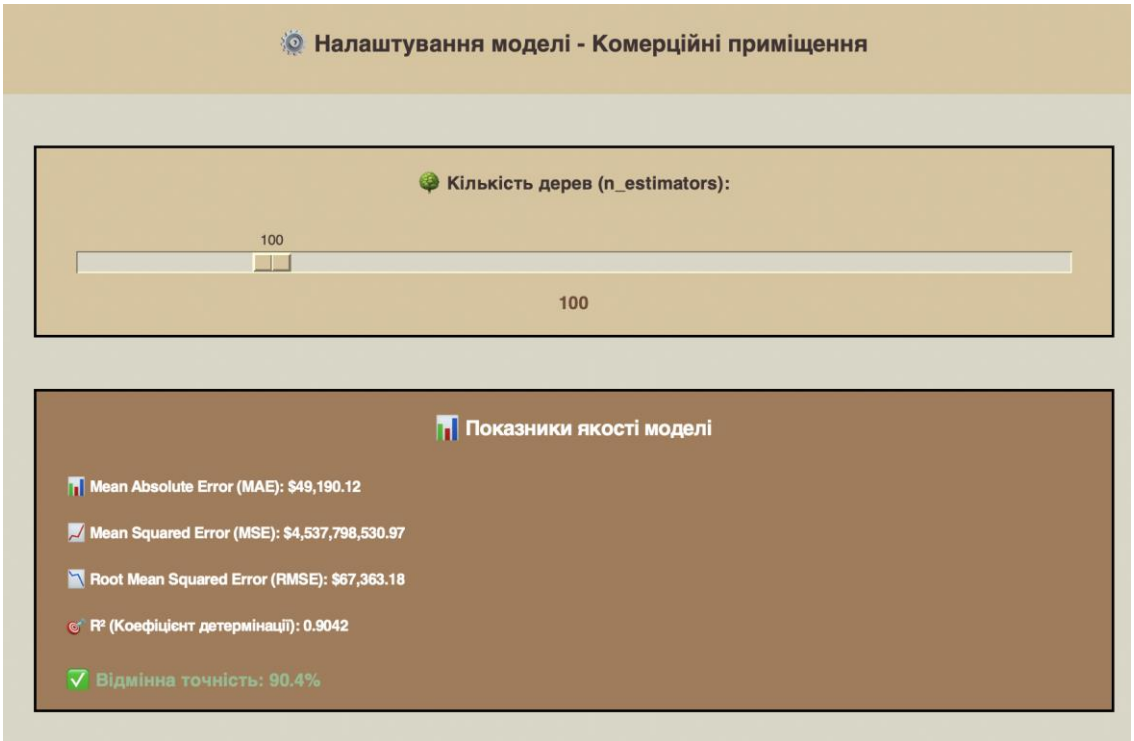

Рисунок Б.5 – Налаштування моделі для комерційної нерухомості

ДОДАТОК В

Фрагмент лістингу програмного коду

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor, IsolationForest
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import tkinter as tk
from tkinter import ttk, messagebox
from tkinter import Tk, Frame, Label, Button, Canvas
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import seaborn as sns
try:
    from PIL import Image, ImageTk
    PIL_AVAILABLE = True
except ImportError:
    PIL_AVAILABLE = False
    print("PIL не встановлено, фонове зображення не буде завантажено")
COLORS = {
    'bg_light': '#D9D6C7',      # Світлий бежевий
    'bg_medium': '#D9C49C',     # Середній бежевий
    'bg_dark': '#A67B56',       # Темний бежевий
    'accent': '#734B34',        # Коричневий акцент
    'text_dark': '#403332',     # Темний текст
    'text_light': '#FFFFFF',    # Світлий текст
    'success': '#8FBC8F',       # Зелений для успіху
    'warning': '#DAA520',       # Золотий для попереджень
    'error': '#CD5C5C'          # Червоний для помилок}
class GraphicsWindow:
    def __init__(self, parent, df, encoders, rf_model=None, category_type="all"):
        self.window = tk.Toplevel(parent)
        self.window.title(f"Графіки аналізу нерухомості - {category_type}")
        self.window.geometry("1200x800")
        self.window.configure(bg=COLORS['bg_light'])
        self.window.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.df = df
        self.encoders = encoders
        self.rf_model = rf_model
        self.category_type = category_type
        self.features = ['Location', 'Square_Meters', 'Number_of_Rooms',
'Year_Built',   'Proximity_to_Center_km',   'Location_Category',   'Building_Type',
'Infrastructure_Availability']
        self.X = self.df[self.features]
        self.y = self.df['Price_USD']
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
            self.X, self.y, test_size=0.2, random_state=42)
        if self.rf_model is None:
            self.model = RandomForestRegressor(n_estimators=100, random_state=42)
            self.model.fit(self.X_train, self.y_train)
        else:
            self.model = rf_model
            self.create_layout()
            def on_closing(self):
        self.window.destroy()
            def create_layout(self):
             style = ttk.Style()
        style.theme_use('clam')
        style.configure('Custom.TButton',
                        background=COLORS['accent'],
```

```
                        foreground=COLORS['text_light'],
                        font=('Helvetica', 11, 'bold'),
                        padding=(15, 10))
        style.map('Custom.TButton',
background=[('active', COLORS['bg_dark']), ('pressed', COLORS['text_dark'])])
      self.button_frame = tk.Frame(self.window, bg=COLORS['bg_medium'], width=250)
        self.button_frame.pack(side=tk.LEFT, fill=tk.Y, padx=0, pady=0)
        self.button_frame.pack_propagate(False)
        self.graph_frame = tk.Frame(self.window, bg=COLORS['bg_light'])
 self.graph_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=10, pady=10)
 self.create_buttons()
            def create_buttons(self):
             title_label = tk.Label(self.button_frame,
                                    text="□ Оберіть тип графіку",
                                    font=('Helvetica', 14, 'bold'),
                                    fg=COLORS['text_dark'],
                                    bg=COLORS['bg_medium'])
                                    title_label.pack(pady=20, padx=10)
                buttons = [
            ("□ Розподіл цін", self.plot_price_distribution),
            ("□ Ціни за містами", self.plot_prices_by_city),
            ("□ Графік аномалій", self.plot_anomalies),
("⋆ Важливість характеристик", self.plot_feature_importance),
            ("□ Прогноз vs Реальність", self.plot_prediction_vs_actual),
            ("□ Ціни за типом будівлі", self.plot_prices_by_building_type)]
            for text, command in buttons:
            btn = tk.Button(self.button_frame,
                            text=text,
                            command=command,
                            font=('Helvetica', 11, 'bold'),
                            bg=COLORS['accent'],
                            fg=COLORS['text_light'],
                            activebackground=COLORS['bg_dark'],
                            activeforeground=COLORS['text_light'],
                            relief='flat',
                            borderwidth=0,
                            cursor='hand2',
                            pady=8)
            btn.pack(fill=tk.X, pady=8, padx=15)
    btn.bind('<Enter>', lambda e, b=btn: b.configure(bg=COLORS['bg_dark']))
            btn.bind('<Leave>', lambda e, b=btn: b.configure(bg=COLORS['accent']))
        def clear_graph_frame(self):
        for widget in self.graph_frame.winfo_children():
            widget.destroy()
        def plot_price_distribution(self):
        self.clear_graph_frame()
        fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
        ax = fig.add_subplot(111)
        ax.set_facecolor(COLORS['bg_light'])
 ax.hist(self.df['Price_USD'],    bins=50,    color=COLORS['accent'],    alpha=0.8,
edgecolor=COLORS['text_dark'])
title = f'Розподіл цін на нерухомість - {self.category_type}'
ax.set_title(title,   color=COLORS['text_dark'],   fontsize=14,   fontweight='bold',
pad=20)
ax.set_xlabel('Ціна (USD)', color=COLORS['text_dark'], fontweight='bold')
ax.set_ylabel('Кількість об\'єктів', color=COLORS['text_dark'], fontweight='bold')
        ax.tick_params(colors=COLORS['text_dark'])
        ax.grid(True, alpha=0.3, color=COLORS['bg_dark'])
                for spine in ax.spines.values():
            spine.set_color(COLORS['bg_dark'])
            spine.set_linewidth(2)
            fig.tight_layout()
        canvas = FigureCanvasTkAgg(fig, self.graph_frame)
```

```
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
        def plot_prices_by_city(self):
        self.clear_graph_frame()
        fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
        ax = fig.add_subplot(111)
        ax.set_facecolor(COLORS['bg_light'])
                df_temp = self.df.copy()
df_temp['Location']=
self.encoders['Location'].inverse_transform(self.df['Location'])city_avg_prices   =
df_temp.groupby('Location')['Price_USD'].mean().round(2)
bars = ax.bar(city_avg_prices.index, city_avg_prices.values,
color=COLORS['bg_medium'], edgecolor=COLORS['accent'], linewidth=2)
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height,
                    f'${height:,.0f}',
                    ha='center', va='bottom', color=COLORS['text_dark'],
                    fontweight='bold', fontsize=10)
        title = f'Середні ціни на нерухомість за містами - {self.category_type}'
ax.set_title(title,   color=COLORS['text_dark'],   fontsize=14,   fontweight='bold',
pad=20)
ax.set_xlabel('Місто', color=COLORS['text_dark'], fontweight='bold', labelpad=10)
ax.set_ylabel('Середня ціна (USD)', color=COLORS['text_dark'], fontweight='bold',
labelpad=10)
        ax.tick_params(colors=COLORS['text_dark'])
        ax.grid(True, alpha=0.3, color=COLORS['bg_dark'])
        for spine in ax.spines.values():
            spine.set_color(COLORS['bg_dark'])
            spine.set_linewidth(2)
        for tick in ax.get_xticklabels():
            tick.set_rotation(45)
        fig.tight_layout()
        canvas = FigureCanvasTkAgg(fig, self.graph_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
def plot_prices_by_building_type(self):
        self.clear_graph_frame()
        fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
        ax = fig.add_subplot(111)
        ax.set_facecolor(COLORS['bg_light'])
        df_temp = self.df.copy()
df_temp['Building_Type']                                                    =
self.encoders['Building_Type'].inverse_transform(self.df['Building_Type'])
        building_avg_prices                                                 =
df_temp.groupby('Building_Type')['Price_USD'].mean().round(2)
        bars = ax.bar(building_avg_prices.index, building_avg_prices.values,
                      color=COLORS['bg_dark'],          edgecolor=COLORS['accent'],
linewidth=2)
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height,
                    f'${height:,.0f}',
                    ha='center', va='bottom', color=COLORS['text_dark'],
                    fontweight='bold', fontsize=10)
        title = f'Середні ціни за типом будівлі - {self.category_type}'
ax.set_title(title,   color=COLORS['text_dark'],   fontsize=14,   fontweight='bold',
pad=20)
ax.set_xlabel('Тип   будівлі',   color=COLORS['text_dark'],   fontweight='bold',
labelpad=10)
ax.set_ylabel('Середня ціна (USD)', color=COLORS['text_dark'], fontweight='bold',
labelpad=10)
        ax.tick_params(colors=COLORS['text_dark'])
```

```
        ax.grid(True, alpha=0.3, color=COLORS['bg_dark'])
        for spine in ax.spines.values():
            spine.set_color(COLORS['bg_dark'])
            spine.set_linewidth(2)
                for tick in ax.get_xticklabels():
            tick.set_rotation(45)
        fig.tight_layout()
                canvas = FigureCanvasTkAgg(fig, self.graph_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
            def plot_anomalies(self):
        self.clear_graph_frame()
        fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
        ax = fig.add_subplot(111)
        ax.set_facecolor(COLORS['bg_light'])
    isolation_forest = IsolationForest(contamination=0.1, random_state=42)
 anomalies = isolation_forest.fit_predict(self.df[['Price_USD', 'Square_Meters']])
                normal_mask = anomalies == 1
        anomaly_mask = anomalies == -1
                ax.scatter(self.df.loc[normal_mask, 'Square_Meters'],
                  self.df.loc[normal_mask, 'Price_USD'],
               c=COLORS['bg_medium'], label='Нормальні об\'єкти', alpha=0.7, s=50,
                  edgecolors=COLORS['accent'], linewidth=1)
        ax.scatter(self.df.loc[anomaly_mask, 'Square_Meters'],
                  self.df.loc[anomaly_mask, 'Price_USD'],
                  c=COLORS['error'], label='Аномалії', alpha=0.8, s=70,
                  edgecolors=COLORS['text_dark'], linewidth=2)
        title = f'Виявлення аномалій на ринку нерухомості - {self.category_type}'
ax.set_title(title,  color=COLORS['text_dark'],  fontsize=14,  fontweight='bold',
pad=20)
        ax.set_xlabel('Площа (кв.м)', color=COLORS['text_dark'], fontweight='bold',
labelpad=10)
        ax.set_ylabel('Ціна  (USD)',  color=COLORS['text_dark'],  fontweight='bold',
labelpad=10)
        ax.tick_params(colors=COLORS['text_dark'])
        ax.grid(True, alpha=0.3, color=COLORS['bg_dark'])
        for spine in ax.spines.values():
            spine.set_color(COLORS['bg_dark'])
            spine.set_linewidth(2)
      legend = ax.legend(facecolor=COLORS['bg_light'], edgecolor=COLORS['accent'],
                         framealpha=0.9, loc='upper left')
        for text in legend.get_texts():
            text.set_color(COLORS['text_dark'])
                fig.tight_layout()
        canvas = FigureCanvasTkAgg(fig, self.graph_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
    def plot_feature_importance(self):
        self.clear_graph_frame()
        fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
        ax = fig.add_subplot(111)
        ax.set_facecolor(COLORS['bg_light'])
                importance = self.model.feature_importances_
        features = ['Локація', 'Площа', 'Кількість кімнат', 'Рік побудови',
                  'Відстань до центру', 'Категорія локації',
                  'Тип будівлі', 'Інфраструктура']
     features_importance = sorted(zip(importance, features), reverse=True)
        importance_sorted = [imp for imp, _ in features_importance]
        features_sorted = [feat for _, feat in features_importance]
colors  =  [COLORS['accent']  if  i  %  2  ==  0  else  COLORS['bg_dark']  for  i  in
range(len(features_sorted))]
        bars = ax.barh(range(len(features_sorted)), importance_sorted,
olor=colors, alpha=0.8, edgecolor=COLORS['text_dark'], linewidth=1)
```

```
                for i, bar in enumerate(bars):
            width = bar.get_width()
            ax.text(width + 0.005, i, f'{width*100:.1f}%',
                    ha='left', va='center', color=COLORS['text_dark'],
                    fontweight='bold', fontsize=11)
        ax.set_yticks(range(len(features_sorted)))
        ax.set_yticklabels(features_sorted)
title = f'Важливість характеристик у прогнозуванні ціни - {self.category_type}'
ax.set_title(title,  color=COLORS['text_dark'],  fontsize=14,  fontweight='bold',
pad=20)      ax.set_xlabel('Важливість      (%)',      color=COLORS['text_dark'],
fontweight='bold', labelpad=10)
ax.tick_params(colors=COLORS['text_dark'])
        ax.grid(True, alpha=0.3, color=COLORS['bg_dark'], axis='x')
        for spine in ax.spines.values():
            spine.set_color(COLORS['bg_dark'])
            spine.set_linewidth(2)
                fig.tight_layout()
        canvas = FigureCanvasTkAgg(fig, self.graph_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
    def plot_prediction_vs_actual(self, new_point=None):
        try:
            self.clear_graph_frame()
            fig = Figure(figsize=(10, 6), facecolor=COLORS['bg_light'])
            ax = fig.add_subplot(111)
            ax.set_facecolor(COLORS['bg_light'])
            y_pred = self.model.predict(self.X_test)
ax.scatter(self.y_test, y_pred, alpha=0.6, color=COLORS['bg_medium'],
label='Тестові дані', s=50, edgecolors=COLORS['accent'], linewidth=1)
if new_point is not None:
ax.scatter([new_point[0]],    [new_point[1]],    color=COLORS['error'],    s=150,
label='Поточний прогноз', edgecolors=COLORS['text_dark'], linewidth=2)
                        max_val = max(max(self.y_test), max(y_pred))
            min_val = min(min(self.y_test), min(y_pred))
            if new_point:
                max_val = max(max_val, new_point[0], new_point[1])
                min_val = min(min_val, new_point[0], new_point[1])
                        ax.plot([min_val, max_val], [min_val, max_val],
                    color=COLORS['accent'], linewidth=3, linestyle='--',
                    label='Ідеальний прогноз')
title = f'Порівняння прогнозованих та реальних цін - {self.category_type}'
ax.set_title(title,  color=COLORS['text_dark'],  fontsize=14,  fontweight='bold',
pad=20)
ax.set_xlabel('Реальна ціна (USD)', color=COLORS['text_dark'], fontweight='bold',
labelpad=10)
ax.set_ylabel('Прогнозована      ціна      (USD)',      color=COLORS['text_dark'],
fontweight='bold', labelpad=10)
ax.tick_params(colors=COLORS['text_dark'])
            ax.grid(True, alpha=0.3, color=COLORS['bg_dark'])
            for spine in ax.spines.values():
                spine.set_color(COLORS['bg_dark'])
                spine.set_linewidth(2)
legend = ax.legend(facecolor=COLORS['bg_light'], edgecolor=COLORS['accent'],
                                framealpha=0.9, loc='upper left')
            for text in legend.get_texts():
                text.set_color(COLORS['text_dark'])
                        fig.tight_layout()
            canvas = FigureCanvasTkAgg(fig, self.graph_frame)
            canvas.draw()
            canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
                    except Exception as e:
            print(f"Помилка при створенні графіку прогнозів: {e}")
            error_label = tk.Label(self.graph_frame,
```

```
                text=f"✗Помилка створення графіку:₩n{str(e)}",
                                font=('Helvetica', 14),
                                fg=COLORS['error'],
                                bg=COLORS['bg_light'],
                                justify='center') error_label.pack(expand=True)
class CategoryWindow:
def __init__(self, parent, category_type, df, encoders, rf_model, isolation_forest):
        self.window = tk.Toplevel(parent)
        self.window.title(f"Нерухомість в Україні - {category_type}")
        self.window.geometry("900x700")
        self.window.configure(bg=COLORS['bg_light'])
        self.window.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.category_type = category_type
        self.df = df
        self.encoders = encoders
        self.rf_model = rf_model
        self.isolation_forest = isolation_forest
        self.graphics_window = None
        self.features = ['Location', 'Square_Meters', 'Number_of_Rooms',
    'Year_Built', 'Proximity_to_Center_km', 'Location_Category',
     'Building_Type', 'Infrastructure_Availability']
        self.create_category_menu()
    def on_closing(self):
        if self.graphics_window:
            self.graphics_window.on_closing()
        self.window.destroy()
    def create_category_menu(self):
        main_frame = tk.Frame(self.window, bg=COLORS['bg_light'])
        main_frame.place(relx=0.5, rely=0.5, anchor='center')
        title_frame = tk.Frame(main_frame, bg=COLORS['bg_light'])
        title_frame.pack(pady=30)
                if self.category_type == 'Житлові приміщення':
            category_color = COLORS['accent']
            icon = '□'
        else:
            category_color = COLORS['bg_dark']
            icon = '□'
        icon_label = tk.Label(title_frame, text=icon,
                              font=('Helvetica', 48),
                              fg=category_color, bg=COLORS['bg_light'])
        icon_label.pack()
        title = tk.Label(title_frame, text=f"{self.category_type}",
                         font=('Helvetica', 28, 'bold'),
                         fg=category_color, bg=COLORS['bg_light'])
        title.pack(pady=(10, 0))
        subtitle = tk.Label(title_frame, text="Аналіз та прогнозування цін",
                            font=('Helvetica', 14),
                            fg=COLORS['text_dark'], bg=COLORS['bg_light'])
        subtitle.pack(pady=(5, 0))
        stats_frame = tk.Frame(main_frame, bg=COLORS['bg_medium'],
                               relief='solid', borderwidth=2)
        stats_frame.pack(pady=20, padx=20, fill='x')
                stats_text = self.get_category_stats()
        stats_label = tk.Label(stats_frame, text=stats_text,
                               font=('Helvetica', 12, 'bold'),
                               fg=COLORS['text_dark'], bg=COLORS['bg_medium'],
                               justify='center', pady=15)
        stats_label.pack()
        button_frame = tk.Frame(main_frame, bg=COLORS['bg_light'])
        button_frame.pack(pady=30)
                button_style = {
            'font': ('Helvetica', 14, 'bold'),
            'relief': 'flat',
```

```
            'borderwidth': 0,
            'cursor': 'hand2',
            'padx': 25,
            'pady': 12,
            'width': 25 }
                buttons = [
            ("  Прогнозування цін", self.open_prediction, COLORS['accent']),
            ("  Графіки та аналітика", self.open_graphics, COLORS['bg_dark']),
("  Налаштування моделі", self.create_model_metrics_window, COLORS['bg_medium']),
      ("  Назад до головного меню", self.window.destroy, COLORS['error'])]
        for text, command, color in buttons:
            btn = tk.Button(button_frame, text=text, command=command,
                            bg=color, fg=COLORS['text_light'],
                            activebackground=COLORS['text_dark'],
                            activeforeground=COLORS['text_light'],
                            **button_style)
            btn.pack(pady=12)
            btn.bind('<Enter>', lambda e, b=btn, c=color: b.configure(
                bg=COLORS['text_dark'] if c != COLORS['error'] else '#B22222'))
            btn.bind('<Leave>', lambda e, b=btn, c=color: b.configure(bg=c))
        def get_category_stats(self):
        total_objects = len(self.df)
        avg_price = self.df['Price_USD'].mean()
        min_price = self.df['Price_USD'].min()
        max_price = self.df['Price_USD'].max()
                return f"""  Об'єктів у категорії: {total_objects}
  Середня ціна: ${avg_price:,.0f}
  Діапазон цін: ${min_price:,.0f} - ${max_price:,.0f}"""
        def open_prediction(self):
        prediction_window = tk.Toplevel(self.window)
        prediction_window.title(f"Прогнозування цін - {self.category_type}")
        prediction_window.geometry("900x800")
        prediction_window.configure(bg=COLORS['bg_light'])
    header_frame = tk.Frame(prediction_window, bg=COLORS['bg_medium'], height=80)
        header_frame.pack(fill='x', pady=0)
        header_frame.pack_propagate(False)
        header_label = tk.Label(header_frame,
     text=f"  Прогнозування цін - {self.category_type}",
                                font=('Helvetica', 18, 'bold'),
                                fg=COLORS['text_dark'], bg=COLORS['bg_medium'])
        header_label.pack(expand=True)
        entries = {}
        canvas = tk.Canvas(prediction_window, bg=COLORS['bg_light'])
        scrollbar     =     ttk.Scrollbar(prediction_window,     orient="vertical",
command=canvas.yview)
        scrollable_frame = tk.Frame(canvas, bg=COLORS['bg_light'])
        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all")))
                canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)
                canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")
                input_frame = tk.Frame(scrollable_frame, bg=COLORS['bg_light'])
        input_frame.pack(pady=20, padx=40, fill='x')
                feature_labels = {
            'Location': 'Місто/локація',
            'Square_Meters': 'Площа (кв.м)',
            'Number_of_Rooms': 'Кількість кімнат',
            'Year_Built': 'Рік побудови',
            'Proximity_to_Center_km': 'Відстань до центру (км)',
            'Location_Category': 'Категорія локації',
            'Building_Type': 'Тип будівлі',
```

```
            'Infrastructure_Availability': 'Доступність інфраструктури'
                for i, feature in enumerate(self.features):
            field_frame = tk.Frame(input_frame, bg=COLORS['bg_medium'],
                                   relief='solid', borderwidth=1)
            field_frame.pack(fill='x', pady=8, padx=10)
                        label = tk.Label(field_frame,
                            text=feature_labels.get(feature, feature),
                            font=('Helvetica', 12, 'bold'),
                            fg=COLORS['text_dark'], bg=COLORS['bg_medium'],
                            anchor='w')
            label.pack(fill='x', padx=15, pady=(10, 5))
 if    feature    in    ['Location',    'Location_Category',    'Building_Type',
'Infrastructure_Availability']:
                unique_values = self.encoders[feature].classes_
                combo = ttk.Combobox(field_frame, values=list(unique_values),
                                     font=('Helvetica', 11), width=35)
                combo.pack(padx=15, pady=(0, 10))
                                    if feature == 'Location':
combo.set('Kyiv' if 'Kyiv' in unique_values else unique_values[0])
                elif feature == 'Building_Type':
                    if self.category_type == 'Житлові приміщення':
combo.set('Apartment' if 'Apartment' in unique_values else unique_values[0])
                    else:
combo.set('Office' if 'Office' in unique_values else unique_values[0])
                else:
                    combo.set(unique_values[0])
                entries[feature] = combo
            else:
                entry = tk.Entry(field_frame, font=('Helvetica', 11), width=35,
                                 bg='white', fg=COLORS['text_dark'],
                                 relief='solid', borderwidth=1)
                entry.pack(padx=15, pady=(0, 10))
                            if feature == 'Square_Meters':
default_value = '70' if self.category_type == 'Житлові приміщення' else '100'
                    entry.insert(0, default_value)
                elif feature == 'Number_of_Rooms':
default_value = '2' if self.category_type == 'Житлові приміщення' else '5'
                    entry.insert(0, default_value)
                elif feature == 'Year_Built':
                    entry.insert(0, '2010')
                elif feature == 'Proximity_to_Center_km':
                    entry.insert(0, '5')
                                    entries[feature] = entry
        result_frame = tk.Frame(scrollable_frame, bg=COLORS['bg_light'])
        result_frame.pack(pady=20, fill='x', padx=40)
                warning_label = tk.Label(result_frame, text="",
                                 font=('Helvetica', 11),
                                 fg=COLORS['warning'], bg=COLORS['bg_light'],
                                 wraplength=700, justify='left')
        warning_label.pack(pady=10)
                result_label = tk.Label(result_frame, text="",
                                font=('Helvetica', 16, 'bold'),
                                fg=COLORS['accent'], bg=COLORS['bg_light'])
        result_label.pack(pady=15)
                def predict():
            try:
                input_data = {}
                for feature in self.features:
                    value = entries[feature].get()
 if value == "":
raise ValueError(f"Поле '{feature_labels.get(feature, feature)}' не може бути
порожнім")
```

```
 if    feature    in    ['Location',    'Location_Category',    'Building_Type',
'Infrastructure_Availability']:
if value not in self.encoders[feature].classes_:
raise ValueError(f"Некоректне значення для {feature_labels.get(feature, feature)}")
value = self.encoders[feature].transform([value])[0]
                    else: value = float(value)
input_data[feature] = value
arning_messages = self.validate_input_data(input_data, self.df)
input_df = pd.DataFrame([input_data])
anomaly_score = self.isolation_forest.predict(input_df)
predicted_price = self.rf_model.predict(input_df)[0]
  warning_text = ""
if warning_messages:
warning_text = "□ Попередження:\n" + "\n".join(warning_messages)
                    if anomaly_score[0] == -1:
  warning_text += "\n□ Увага: введені значення можуть бути нетиповими для ринку!"
                    warning_label.config(text=warning_text, fg=COLORS['warning'])
                else:
                    if anomaly_score[0] == -1:
warning_label.config(text="□ Увага: введені значення можуть бути нетиповими для
ринку!",
fg=COLORS['warning'])
                    else:
warning_label.config(text="✓Параметри в межах норми", fg=COLORS['success'])
result_label.config(text=f"□ Прогнозована ціна: ${predicted_price:,.2f}")
if self.graphics_window is not None and hasattr(self.graphics_window, 'window') and
self.graphics_window.window.winfo_exists():
                    self.graphics_window.plot_prediction_vs_actual(
                        new_point=(predicted_price, predicted_price))
                            except ValueError as ve:
                messagebox.showerror("Помилка", str(ve))
            except Exception as e:
                messagebox.showerror("Помилка", f"Виникла помилка: {str(e)}")
        button_frame = tk.Frame(scrollable_frame, bg=COLORS['bg_light'])
        button_frame.pack(pady=25)
predict_button = tk.Button(button_frame, text="□ Прогнозувати ціну",
                                   command=predict,
                                   font=('Helvetica', 14, 'bold'),
                                   bg=COLORS['accent'], fg=COLORS['text_light'],
                                   activebackground=COLORS['text_dark'],
                                   relief='flat', borderwidth=0,
                                   cursor='hand2', padx=20, pady=10)
        predict_button.pack(side=tk.LEFT, padx=15)
                clear_button = tk.Button(button_frame, text="□ Очистити поля",
 command=lambda:  [entry.delete(0,  tk.END)  if  hasattr(entry,  'delete')  else
entry.set('') for entry in entries.values()],
                                 font=('Helvetica', 14, 'bold'),
                                 bg=COLORS['bg_dark'], fg=COLORS['text_light'],
                                 activebackground=COLORS['text_dark'],
                                 relief='flat', borderwidth=0,
                                 cursor='hand2', padx=20, pady=10)
        clear_button.pack(side=tk.LEFT, padx=15)
        predict_button.bind('<Enter>',                    lambda                    e:
predict_button.configure(bg=COLORS['text_dark']))
        predict_button.bind('<Leave>',                    lambda                    e:
predict_button.configure(bg=COLORS['accent']))
        clear_button.bind('<Enter>',                    lambda                    e:
clear_button.configure(bg=COLORS['text_dark']))
        clear_button.bind('<Leave>',                    lambda                    e:
clear_button.configure(bg=COLORS['bg_dark']))
        def validate_input_data(self, input_data, df):
        messages = []
                if input_data['Year_Built'] > 2024:
```

```
            raise ValueError("Рік побудови не може бути більшим за 2024")
        elif input_data['Year_Built'] < 1900:
            raise ValueError("Рік побудови не може бути меншим за 1900")
                max_area = df['Square_Meters'].quantile(0.99)
        min_area = df['Square_Meters'].quantile(0.01)
        if input_data['Square_Meters'] > max_area:
            messages.append(f"Площа  ({input_data['Square_Meters']}  кв.м)  значно
більша за типові значення "
f"(максимум в вибірці: {max_area:.2f} кв.м)")
        elif input_data['Square_Meters'] <= 0:
            raise ValueError("Площа має бути більшою за 0")
        elif input_data['Square_Meters'] < min_area:
messages.append(f"Площа ({input_data['Square_Meters']} кв.м) значно менша за типові
значення "
                        f"(мінімум в вибірці: {min_area:.2f} кв.м)")
                max_rooms = df['Number_of_Rooms'].quantile(0.99)
        min_rooms = df['Number_of_Rooms'].quantile(0.01)
        if input_data['Number_of_Rooms'] > max_rooms:
            messages.append(f"Кількість  кімнат  ({input_data['Number_of_Rooms']})
значно більша за типові значення "
                        f"(максимум в вибірці: {int(max_rooms)})")
        elif input_data['Number_of_Rooms'] <= 0:
            raise ValueError("Кількість кімнат має бути більшою за 0")
        elif input_data['Number_of_Rooms'] < min_rooms:
            messages.append(f"Кількість  кімнат  ({input_data['Number_of_Rooms']})
значно менша за типові значення "
                        f"(мінімум в вибірці: {int(min_rooms)})")
                if input_data['Proximity_to_Center_km'] < 0:
            raise ValueError("Відстань до центру не може бути від'ємною")
                return messages
        def open_graphics(self):
        if  self.graphics_window  is  None  or  not  hasattr(self.graphics_window,
'window') or not self.graphics_window.window.winfo_exists():
            self.graphics_window    =    GraphicsWindow(self.window,    self.df,
self.encoders, self.rf_model, self.category_type)
        def create_model_metrics_window(self):
        metrics_window = tk.Toplevel(self.window)
        metrics_window.title(f"Характеристики моделі - {self.category_type}")
        metrics_window.geometry("900x600")
        metrics_window.configure(bg=COLORS['bg_light'])
        header_frame = tk.Frame(metrics_window, bg=COLORS['bg_medium'], height=80)
        header_frame.pack(fill='x')
        header_frame.pack_propagate(False)
                header_label = tk.Label(header_frame,
    text=f" Налаштування моделі - {self.category_type}",
                                font=('Helvetica', 18, 'bold'),
                                fg=COLORS['text_dark'], bg=COLORS['bg_medium'])
        header_label.pack(expand=True)
        content_frame = tk.Frame(metrics_window, bg=COLORS['bg_light'])
        content_frame.pack(fill='both', expand=True, padx=30, pady=20)
        slider_frame = tk.Frame(content_frame, bg=COLORS['bg_medium'],
                                relief='solid', borderwidth=2)
        slider_frame.pack(fill='x', pady=20)
        tk.Label(slider_frame,
                 text=" Кількість дерев (n_estimators):",
                 font=('Helvetica', 14, 'bold'),
                 fg=COLORS['text_dark'], bg=COLORS['bg_medium']).pack(pady=15)
        n_estimators_var = tk.IntVar(value=100)
                slider_container = tk.Frame(slider_frame, bg=COLORS['bg_medium'])
        slider_container.pack(fill='x', padx=30, pady=10)
                slider = tk.Scale(slider_container,
                          from_=10, to=500,
                          variable=n_estimators_var,
```

```
                    orient='horizontal',
                    bg=COLORS['bg_medium'],
                    fg=COLORS['text_dark'],
                    activebackground=COLORS['accent'],
                    highlightbackground=COLORS['bg_medium'],
                    troughcolor=COLORS['bg_light'],
                    font=('Helvetica', 11))
  slider.pack(fill='x')
  value_label = tk.Label(slider_frame, text="100",
                         font=('Helvetica', 14, 'bold'),
                         fg=COLORS['accent'], bg=COLORS['bg_medium'])
  value_label.pack(pady=(0, 15))
  metrics_frame = tk.Frame(content_frame, bg=COLORS['bg_dark'],
                           relief='solid', borderwidth=2)
  metrics_frame.pack(fill='both', expand=True, pady=20)
  tk.Label(metrics_frame, text="□ Показники якості моделі",
          font=('Helvetica', 16, 'bold'),
          fg=COLORS['text_light'], bg=COLORS['bg_dark']).pack(pady=15)
  mae_label = tk.Label(metrics_frame, text="MAE: ",
                       font=('Helvetica', 12, 'bold'),
                       fg=COLORS['text_light'], bg=COLORS['bg_dark'])
  mae_label.pack(pady=8, anchor='w', padx=20)
  mse_label = tk.Label(metrics_frame, text="MSE: ",
                       font=('Helvetica', 12, 'bold'),
                       fg=COLORS['text_light'], bg=COLORS['bg_dark'])
  mse_label.pack(pady=8, anchor='w', padx=20)
  rmse_label = tk.Label(metrics_frame, text="RMSE: ",
                        font=('Helvetica', 12, 'bold'),
                        fg=COLORS['text_light'], bg=COLORS['bg_dark'])
  rmse_label.pack(pady=8, anchor='w', padx=20)
  r2_label = tk.Label(metrics_frame, text="R²: ",
                      font=('Helvetica', 12, 'bold'),
                      fg=COLORS['text_light'], bg=COLORS['bg_dark'])
  r2_label.pack(pady=8, anchor='w', padx=20)
          accuracy_label = tk.Label(metrics_frame, text="Точність моделі: ",
                            font=('Helvetica', 14, 'bold'),
                            fg=COLORS['bg_light'], bg=COLORS['bg_dark'])
  accuracy_label.pack(pady=15, anchor='w', padx=20)
  def update_metrics(*args):
      try:
          n_estimators = n_estimators_var.get()
          value_label.config(text=str(n_estimators))
          X = self.df[self.features]
          y = self.df['Price_USD']
          X_train, X_test, y_train, y_test = train_test_split(
              X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(n_estimators=n_estimators, random_state=42)
          rf_model.fit(X_train, y_train)
          y_pred = rf_model.predict(X_test)
          mae = mean_absolute_error(y_test, y_pred)
          mse = mean_squared_error(y_test, y_pred)
          rmse = np.sqrt(mse)
          r2 = r2_score(y_test, y_pred)
                            accuracy_percentage = max(0, r2 * 100)
     mae_label.config(text=f"□ Mean Absolute Error (MAE): ${mae:,.2f}")
          mse_label.config(text=f"□ Mean Squared Error (MSE): ${mse:,.2f}")
 rmse_label.config(text=f"□ Root Mean Squared Error (RMSE): ${rmse:,.2f}")
          r2_label.config(text=f"□ R² (Коефіцієнт детермінації): {r2:.4f}")
                            if accuracy_percentage >= 80:
    accuracy_text = f"✓Відмінна точність: {accuracy_percentage:.1f}%"
              accuracy_color = COLORS['success']
          elif accuracy_percentage >= 60:
                            except Exception as e:
```

```
                print(f"Помилка при оновленні метрик: {e}")
        n_estimators_var.trace('w', update_metrics)
        update_metrics()
        def apply_changes():
            try:
                n_estimators = n_estimators_var.get()
                self.rf_model = RandomForestRegressor(n_estimators=n_estimators,
                                                      random_state=42)
                X = self.df[self.features]
                y = self.df['Price_USD']
                self.rf_model.fit(X, y)
                messagebox.showinfo("Успіх",          f"Модель     для     категорії
'{self.category_type}' успішно оновлена з {n_estimators} деревами!")
            except Exception as e:
messagebox.showerror("Помилка", f"Не вдалося оновити модель: {str(e)}")
        button_frame = tk.Frame(metrics_window, bg=COLORS['bg_light'])
        button_frame.pack(pady=20)
        apply_btn = tk.Button(button_frame, text="✓Застосувати зміни",
                              command=apply_changes,
                              font=('Helvetica', 12, 'bold'),
                              bg=COLORS['success'], fg=COLORS['text_light'],
                              activebackground=COLORS['accent'],
                              relief='flat', borderwidth=0,
                              cursor='hand2', padx=20, pady=8)
        apply_btn.pack(side=tk.LEFT, padx=15)
reset_btn = tk.Button(button_frame, text="□ Скинути до значень за замовчуванням",
                              command=lambda: n_estimators_var.set(100),
                              font=('Helvetica', 12, 'bold'),
                              bg=COLORS['bg_dark'], fg=COLORS['text_light'],
                              activebackground=COLORS['text_dark'],
                              relief='flat', borderwidth=0,
                              cursor='hand2', padx=20, pady=8)
        reset_btn.pack(side=tk.LEFT, padx=15)
apply_btn.bind('<Enter>', lambda e: apply_btn.configure(bg=COLORS['accent']))
apply_btn.bind('<Leave>', lambda e: apply_btn.configure(bg=COLORS['success']))
reset_btn.bind('<Enter>', lambda e: reset_btn.configure(bg=COLORS['text_dark']))
reset_btn.bind('<Leave>', lambda e: reset_btn.configure(bg=COLORS['bg_dark']))
class MainApplication:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Нерухомість в Україні - Головне меню")
        self.root.geometry("1000x800")
        self.root.resizable(True, True)
        self.root.configure(bg=COLORS['bg_light'])
        self.residential_data = None
        self.commercial_data = None
        self.residential_models = {}
        self.commercial_models = {}
                self.load_data()
        self.create_main_menu()
        def load_data(self):
        """Завантаження та підготовка даних"""
        try:
            self.df = pd.read_csv('Real_Estate_Ukraine_with_Category.csv')
            self.df = self.df[self.df['Price_USD'] > 0]
        print("✓Дані з файлу завантажені успішно")
                except Exception as e:
            print(f"□ Не вдалося завантажити файл CSV: {e}")
            print("□ Створення тестових даних...")
            self.create_sample_data()
            return
                if 'Property_Category' not in self.df.columns:
```

```
            residential_types = ['Apartment', 'House', 'Villa', 'Townhouse']
            commercial_types = ['Office', 'Retail', 'Warehouse', 'Industrial']
                        def categorize_property(row):
                building_type = str(row.get('Building_Type', ''))
                rooms = row.get('Number_of_Rooms', 0)
if any(res_type.lower() in building_type.lower() for res_type in residential_types):
                    return 'Residential'
elif    any(com_type.lower()    in    building_type.lower()    for    com_type    in
commercial_types): return 'Commercial'
                elif rooms <= 4: return 'Residential'
                else: return 'Commercial'
self.df['Property_Category'] = self.df.apply(categorize_property, axis=1)
self.residential_data       =       self.df[self.df['Property_Category']       ==
'Residential'].copy()
self.commercial_data = self.df[self.df['Property_Category'] == 'Commercial'].copy()
if len(self.residential_data) == 0 or len(self.commercial_data) == 0:
            total_rows = len(self.df)
            residential_indices = self.df.index[:total_rows//2]
            commercial_indices = self.df.index[total_rows//2:]
self.residential_data = self.df.loc[residential_indices].copy()
            self.commercial_data = self.df.loc[commercial_indices].copy()
                self.prepare_category_data('residential')
        self.prepare_category_data('commercial')
        print(f"✓ Дані завантажені успішно:")
        print(f"   Житлові приміщення: {len(self.residential_data)} об'єктів")
        print(f"   Комерційні приміщення: {len(self.commercial_data)} об'єктів")
        def create_sample_data(self):
        """Створення тестових даних якщо основний файл недоступний"""
        print(" Створення тестових даних...")
        np.random.seed(42)
        residential_data = {
'Location':   np.random.choice(['Kyiv',   'Lviv',   'Odesa',   'Kharkiv'],   150),
'Square_Meters': np.random.normal(70, 25, 150).clip(20, 200), 'Number_of_Rooms':
np.random.choice([1, 2, 3, 4, 5], 150, p=[0.1, 0.3, 0.3, 0.2, 0.1]),
            'Year_Built': np.random.randint(1980, 2024, 150),
            'Proximity_to_Center_km': np.random.exponential(5, 150).clip(0.5, 30),
'Location_Category':  np.random.choice(['Center',  'Suburb',  'Outskirts'],  150),
'Building_Type':   np.random.choice(['Apartment',   'House',   'Villa'],   150),
'Infrastructure_Availability': np.random.choice(['Good', 'Average', 'Poor'], 150),
'Property_Category': ['Residential'] * 150}
        commercial_data = {
'Location': np.random.choice(['Kyiv', 'Lviv', 'Odesa', 'Kharkiv'], 100),
            'Square_Meters': np.random.normal(150, 80, 100).clip(50, 500),
            'Number_of_Rooms': np.random.choice([3, 5, 8, 10, 15], 100),
            'Year_Built': np.random.randint(1990, 2024, 100),
            'Proximity_to_Center_km': np.random.exponential(3, 100).clip(0.5, 20),
            'Location_Category': np.random.choice(['Center', 'Business District',
'Industrial'],   100),   'Building_Type':   np.random.choice(['Office',   'Retail',
'Warehouse'], 100), 'Infrastructure_Availability': np.random.choice(['Excellent',
'Good', 'Average'], 100),
            'Property_Category': ['Commercial'] * 100}
        def calculate_price(data, base_price):
            prices = []
            for i in range(len(data['Square_Meters'])):
                price = base_price
                price *= data['Square_Meters'][i] / 70
                price *= (1 + (data['Number_of_Rooms'][i] - 2) * 0.1)
                price *= (1 + (data['Year_Built'][i] - 2000) * 0.01)
                price *= (1 - data['Proximity_to_Center_km'][i] * 0.02)
                price *= np.random.normal(1, 0.15)
                prices.append(max(price, base_price * 0.3))
            return prices
        residential_data['Price_USD'] = calculate_price(residential_data, 50000)
```

```
        commercial_data['Price_USD'] = calculate_price(commercial_data, 100000)
        self.residential_data = pd.DataFrame(residential_data)
        self.commercial_data = pd.DataFrame(commercial_data)
self.df      =      pd.concat([self.residential_data,      self.commercial_data],
ignore_index=True)
                self.prepare_category_data('residential')
        self.prepare_category_data('commercial')
        print("✓Тестові дані створені успішно")
        def prepare_category_data(self, category):
        """Підготовка даних для конкретної категорії"""
data = self.residential_data if category == 'residential' else self.commercial_data
                if len(data) == 0:
            return        encoders = {}
categorical_columns    =    ['Location',    'Location_Category',    'Building_Type',
'Infrastructure_Availability']        data_encoded = data.copy()
        for column in categorical_columns:
            if column in data_encoded.columns:
                encoders[column] = LabelEncoder()
data_encoded[column]                                                                =
encoders[column].fit_transform(data_encoded[column].astype(str))
                features = ['Location', 'Square_Meters', 'Number_of_Rooms',
                   'Year_Built', 'Proximity_to_Center_km', 'Location_Category',
                   'Building_Type', 'Infrastructure_Availability']
                X = data_encoded[features]
        y = data_encoded['Price_USD']
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
        rf_model.fit(X, y)
isolation_forest = IsolationForest(contamination=0.1, random_state=42)
        isolation_forest.fit(X)
                models = {
            'rf_model': rf_model,
            'isolation_forest': isolation_forest,
            'encoders': encoders,
            'data': data_encoded,
            'features': features }
                if category == 'residential':
            self.residential_models = models
        else:
            self.commercial_models = models
        def create_main_menu(self):
        main_container = tk.Frame(self.root, bg=COLORS['bg_light'])
        main_container.place(relx=0.5, rely=0.5, anchor='center')
        title_frame = tk.Frame(main_container, bg=COLORS['bg_light'])
        title_frame.pack(pady=40)
        main_icon = tk.Label(title_frame, text="□",
                             font=('Helvetica', 64),
                             fg=COLORS['accent'], bg=COLORS['bg_light'])
        main_icon.pack()
                main_title = tk.Label(title_frame, text="Нерухомість в Україні",
                              font=('Helvetica', 36, 'bold'),
                              fg=COLORS['text_dark'], bg=COLORS['bg_light'])
        main_title.pack(pady=(10, 0))
subtitle   =   tk.Label(title_frame,   text="□   Аналіз   та   прогнозування   цін   на
нерухомість",font=('Helvetica', 16), fg=COLORS['accent'], bg=COLORS['bg_light'])
        subtitle.pack(pady=(10, 0))
        line_frame = tk.Frame(main_container, bg=COLORS['accent'], height=3)
        line_frame.pack(fill='x', pady=20, padx=100)
        categories_frame = tk.Frame(main_container, bg=COLORS['bg_light'])
        categories_frame.pack(pady=30)
        residential_frame = tk.Frame(categories_frame, bg=COLORS['bg_medium'],
                                     relief='solid', borderwidth=3,
                                     highlightbackground=COLORS['accent'],
                                     highlightthickness=2)
```

```
        residential_frame.pack(side=tk.LEFT, padx=25, pady=10)
        res_content = tk.Frame(residential_frame, bg=COLORS['bg_medium'])
        res_content.pack(padx=25, pady=25)
                residential_icon = tk.Label(res_content, text="□",
                                    font=('Helvetica', 56),
                                    fg=COLORS['accent'], bg=COLORS['bg_medium'])
        residential_icon.pack()
residential_title = tk.Label(res_content, text="Житлові приміщення",
font=('Helvetica', 18, 'bold'), fg=COLORS['text_dark'], bg=COLORS['bg_medium'])
        residential_title.pack(pady=(15, 10))
residential_info  =  tk.Label(res_content,  text=f"□  {len(self.residential_data)}
об'єктів\n□ Від ${self.residential_data['Price_USD'].min():,.0f}",
                                    font=('Helvetica', 12),
                                    fg=COLORS['text_dark'], bg=COLORS['bg_medium'],
                                    justify='center')
        residential_info.pack(pady=5)
residential_btn = tk.Button(res_content, text="□ Детальний аналіз",
command=lambda: self.open_category('Житлові приміщення'),
font=('Helvetica', 14, 'bold'),
bg=COLORS['text_dark'], fg=COLORS['text_light'],
                                    activebackground=COLORS['accent'],
                                    activeforeground=COLORS['text_light'],
                                    relief='flat', borderwidth=0,
                                    cursor='hand2', padx=25, pady=12)
        residential_btn.pack(pady=(15, 0))
        commercial_frame = tk.Frame(categories_frame, bg=COLORS['bg_dark'],
                                    relief='solid', borderwidth=3,
                                    highlightbackground=COLORS['accent'],
                                    highlightthickness=2)
        commercial_frame.pack(side=tk.LEFT, padx=25, pady=10)
        com_content = tk.Frame(commercial_frame, bg=COLORS['bg_dark'])
com_content.pack(padx=25, pady=25)
ommercial_icon   =   tk.Label(com_content,   text="□",   font=('Helvetica',   56),
fg=COLORS['bg_medium'], bg=COLORS['bg_dark'])
        commercial_icon.pack()
commercial_title = tk.Label(com_content, text="Комерційні приміщення",
                                    font=('Helvetica', 18, 'bold'),
                                    fg=COLORS['text_light'], bg=COLORS['bg_dark'])
commercial_title.pack(pady=(15, 10))
ommercial_info   =   tk.Label(com_content,   text=f"□   {len(self.commercial_data)}
об'єктів\n□ Від ${self.commercial_data['Price_USD'].min():,.0f}",
                                font=('Helvetica', 12),
                                fg=COLORS['bg_light'], bg=COLORS['bg_dark'],
                                justify='center')  commercial_info.pack(pady=5)
      commercial_btn = tk.Button(com_content, text="□ Детальний аналіз",
          command=lambda: self.open_category('Комерційні приміщення'),
                                font=('Helvetica', 14, 'bold'),
                                bg=COLORS['text_dark'], fg=COLORS['text_light'],
                                activebackground=COLORS['accent'],
                                activeforeground=COLORS['text_light'],
                                relief='flat', borderwidth=0,
                                cursor='hand2', padx=25, pady=12)
        commercial_btn.pack(pady=(15, 0))
        info_frame = tk.Frame(main_container, bg=COLORS['bg_medium'],
                              relief='solid', borderwidth=2)
        info_frame.pack(pady=40, padx=50, fill='x')
                info_title = tk.Label(info_frame, text="□ Можливості системи",
                              font=('Helvetica', 16, 'bold'),
                              fg=COLORS['text_dark'], bg=COLORS['bg_medium'])
        info_title.pack(pady=(20, 10))
                features_text = """
        """
                info_label = tk.Label(info_frame, text=features_text,
```

```
                                font=('Helvetica', 12),
                                fg=COLORS['text_dark'], bg=COLORS['bg_medium'],
                                justify='left')
        info_label.pack(pady=(0, 20))
        def on_enter_residential(event):
            residential_btn.configure(bg=COLORS['accent'])
            residential_frame.configure(highlightbackground=COLORS['text_dark'])
         def on_leave_residential(event):
            residential_btn.configure(bg=COLORS['text_dark'])
            residential_frame.configure(highlightbackground=COLORS['accent'])
                def on_enter_commercial(event):
            commercial_btn.configure(bg=COLORS['accent'])
            commercial_frame.configure(highlightbackground=COLORS['text_dark'])
                def on_leave_commercial(event):
            commercial_btn.configure(bg=COLORS['text_dark'])
            commercial_frame.configure(highlightbackground=COLORS['accent'])
                residential_btn.bind('<Enter>', on_enter_residential)
        residential_btn.bind('<Leave>', on_leave_residential)
        commercial_btn.bind('<Enter>', on_enter_commercial)
        commercial_btn.bind('<Leave>', on_leave_commercial)
        residential_frame.bind('<Enter>', on_enter_residential)
        residential_frame.bind('<Leave>', on_leave_residential)
        commercial_frame.bind('<Enter>', on_enter_commercial)
        commercial_frame.bind('<Leave>', on_leave_commercial)
footer_frame = tk.Frame(main_container, bg=COLORS['bg_light'])
        footer_frame.pack(pady=30)
        footer_label = tk.Label(footer_frame,
text="□ Real Estate Analysis System v2.0 | Створено для аналізу українського ринку
нерухомості", font=('Helvetica', 10), fg=COLORS['accent'], bg=COLORS['bg_light'])
        footer_label.pack()
        def open_category(self, category_type):
        """Відкриття вікна категорії"""
        if category_type == 'Житлові приміщення':
            models = self.residential_models
            data = self.residential_data       else:
            models = self.commercial_models
            data = self.commercial_data
       if len(data) == 0: messagebox.showwarning("Попередження", f"Немає даних для
категорії: {category_type}")
            return
        CategoryWindow(
            self.root,
            category_type,
            models['data'],
            models['encoders'],
            models['rf_model'],
            models['isolation_forest'])
    def run(self):
        """Запуск застосунку"""
        self.root.mainloop()
if __name__ == "__main__":
    print("□ Запуск застосунку 'Нерухомість в Україні'...")
    app = MainApplication()
    app.run()
```

ДОДАТОК Г

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

«Програмний модуль визначення вартості комерційних та житлових приміщень на ринку нерухомості»

Виконала: студентка 4 курсу, групи 2КН-21б

Спеціальності 122 – Комп'ютерні науки

(шифр і назва напрямку підготовки, спеціальності)

Анастасія ПРИМАК

(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН

Ярослав ІВАНЧУК

(прізвище та ініціали)

«03» червня 2025 р.

Вінниця ВНТУ – 2025 рік

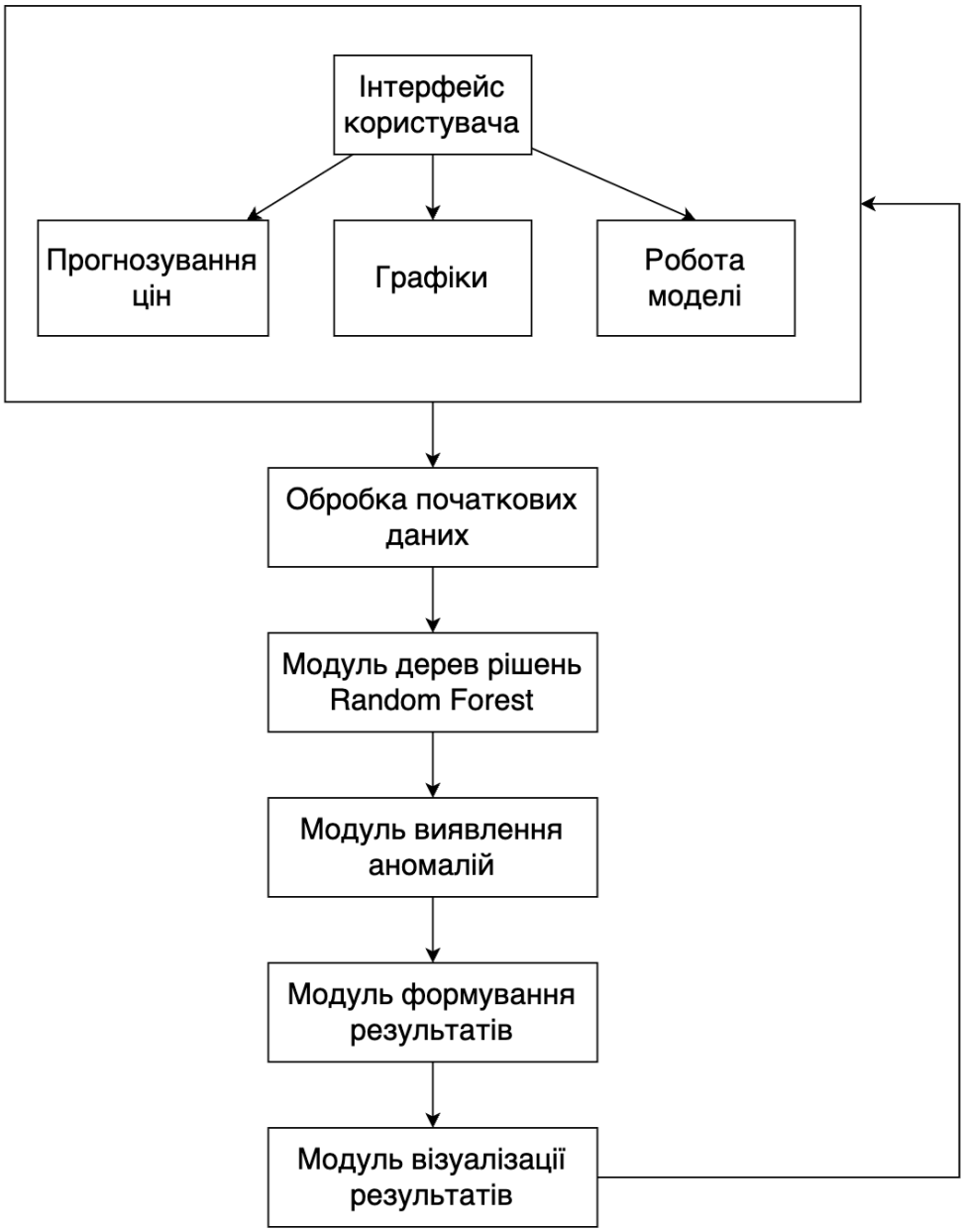

Рисунок Г.1 - Структурна схема програмного забезпечення

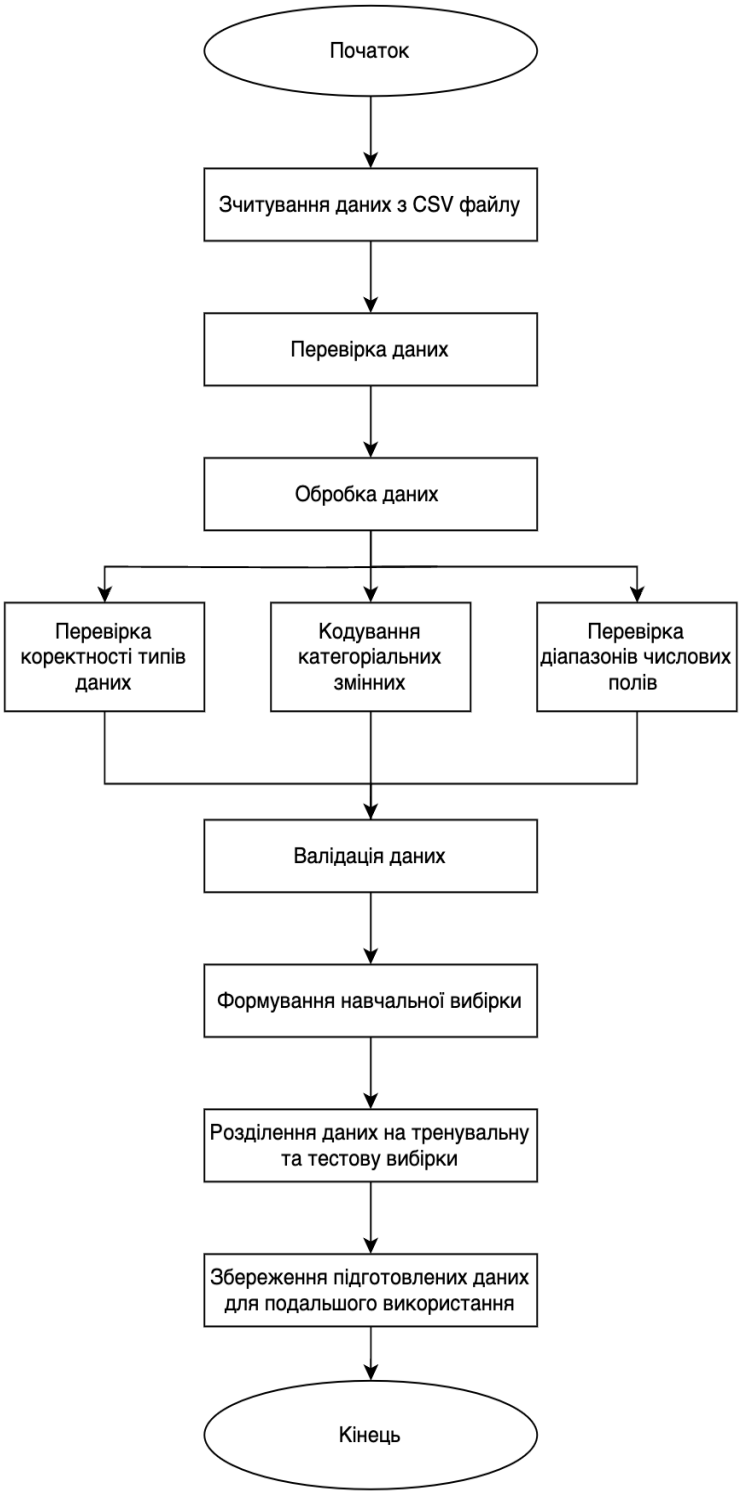

Рисунок Г.2 – Блок-схема модуля зчитування та обробки даних

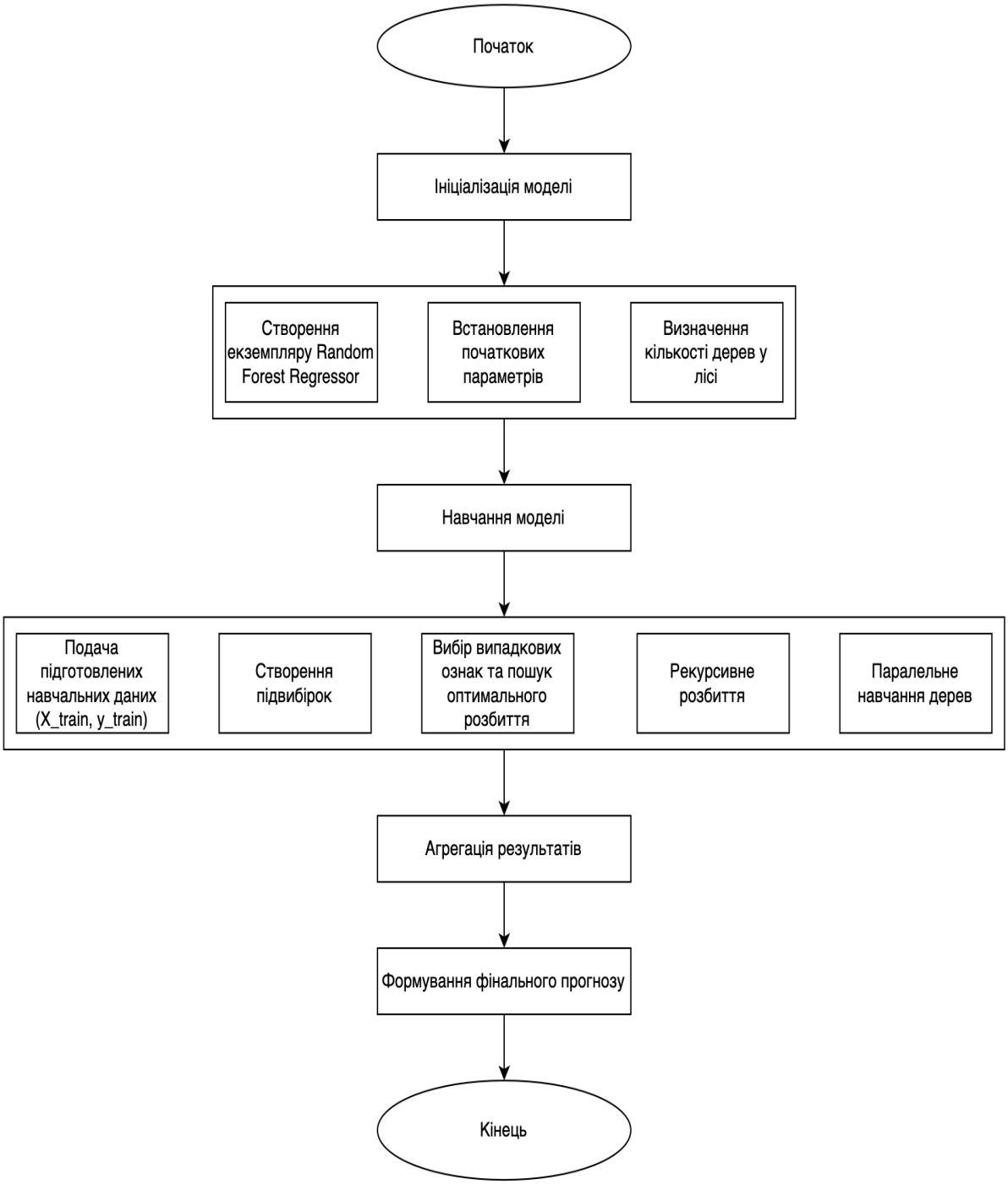

Рисунок Г.3 – Блок-схема інтелектуального модуля прийняття рішень на основі Random Forest

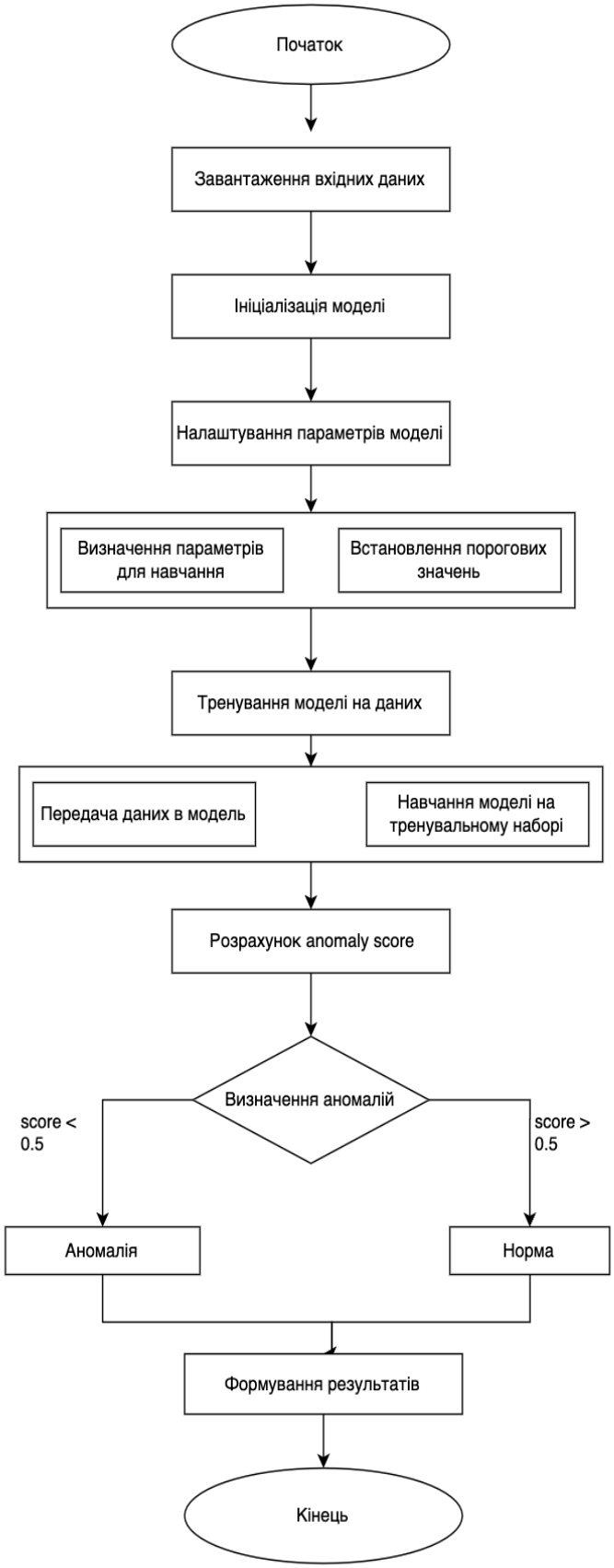

Рисунок Г.4 – Блок-схема модуля аналізу та виявлення аномалій

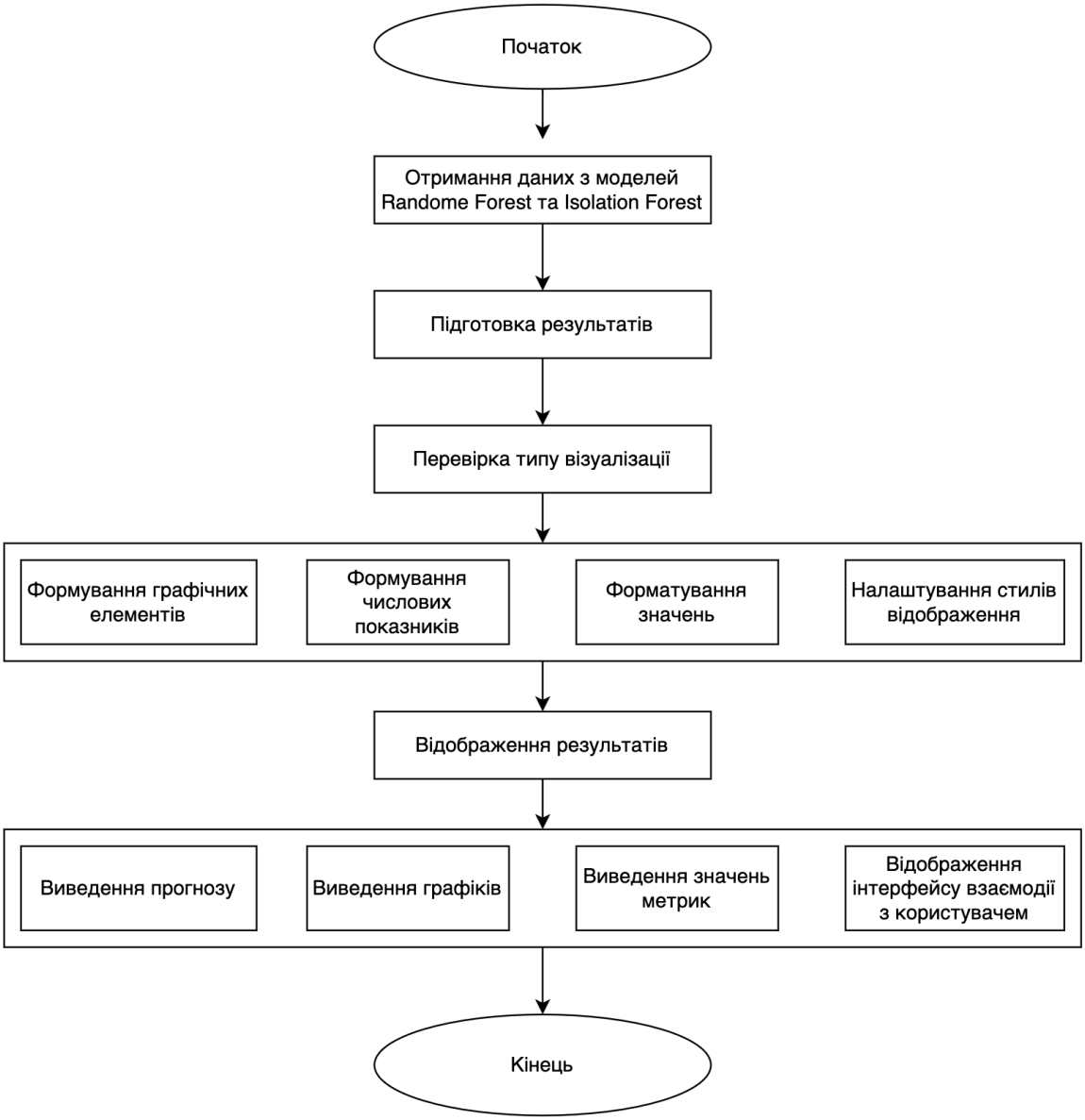

Рисунок Г.5 – Блок-схема модуля генерації та формування результатів

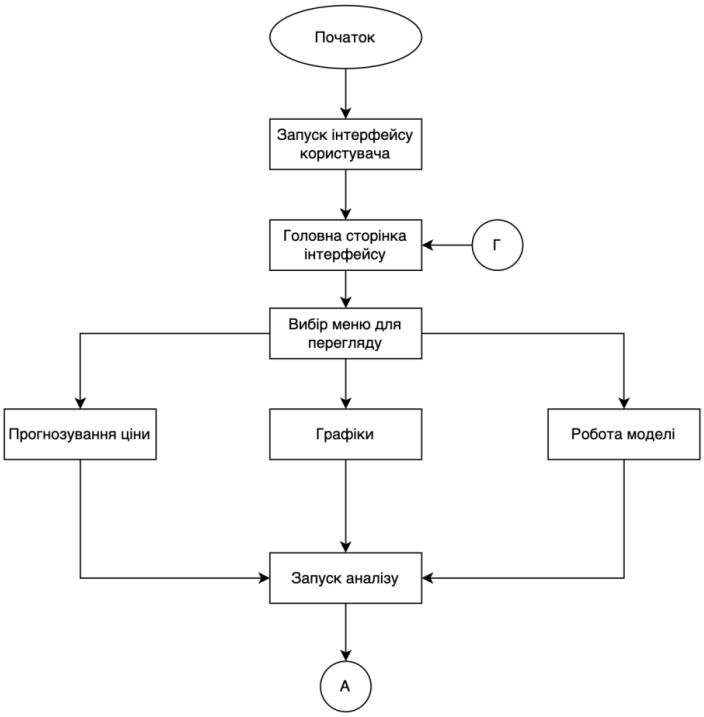

Рисунок Г.6 – Схема алгоритму функціонування програмного забезпечення

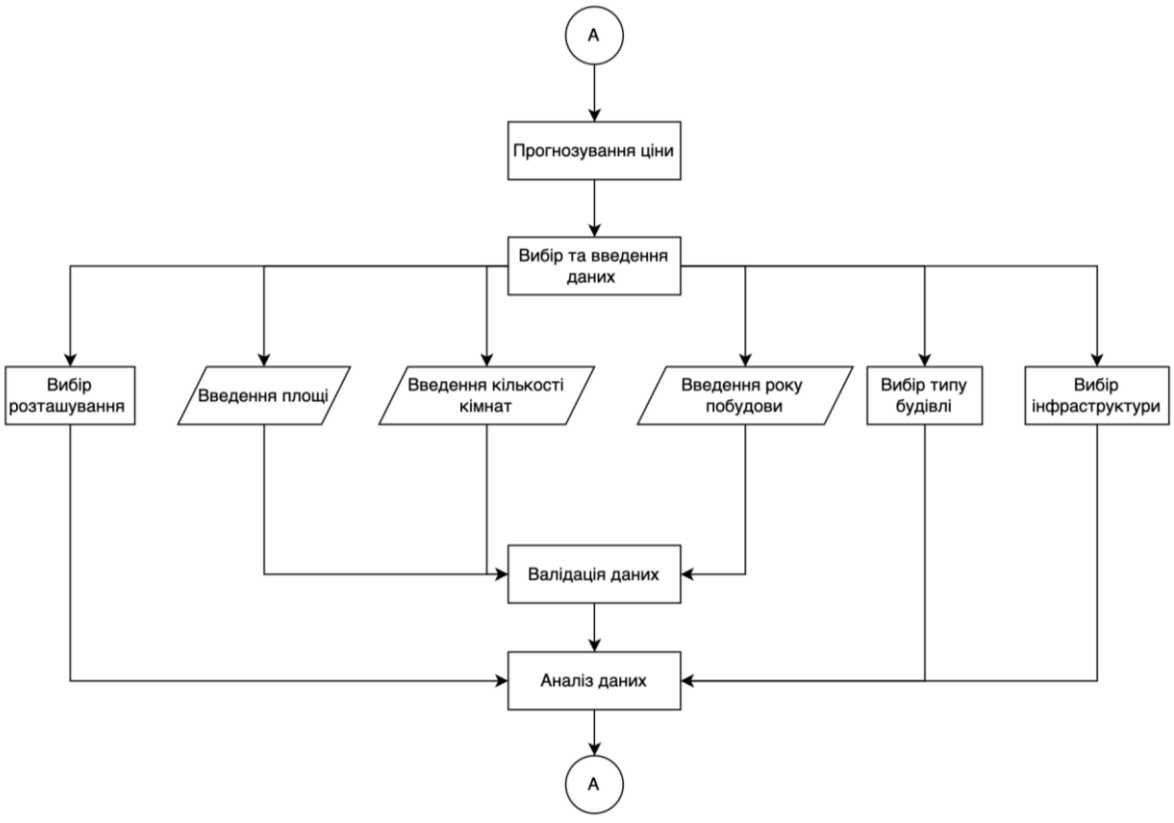

Рисунок Г.7 – Продовження

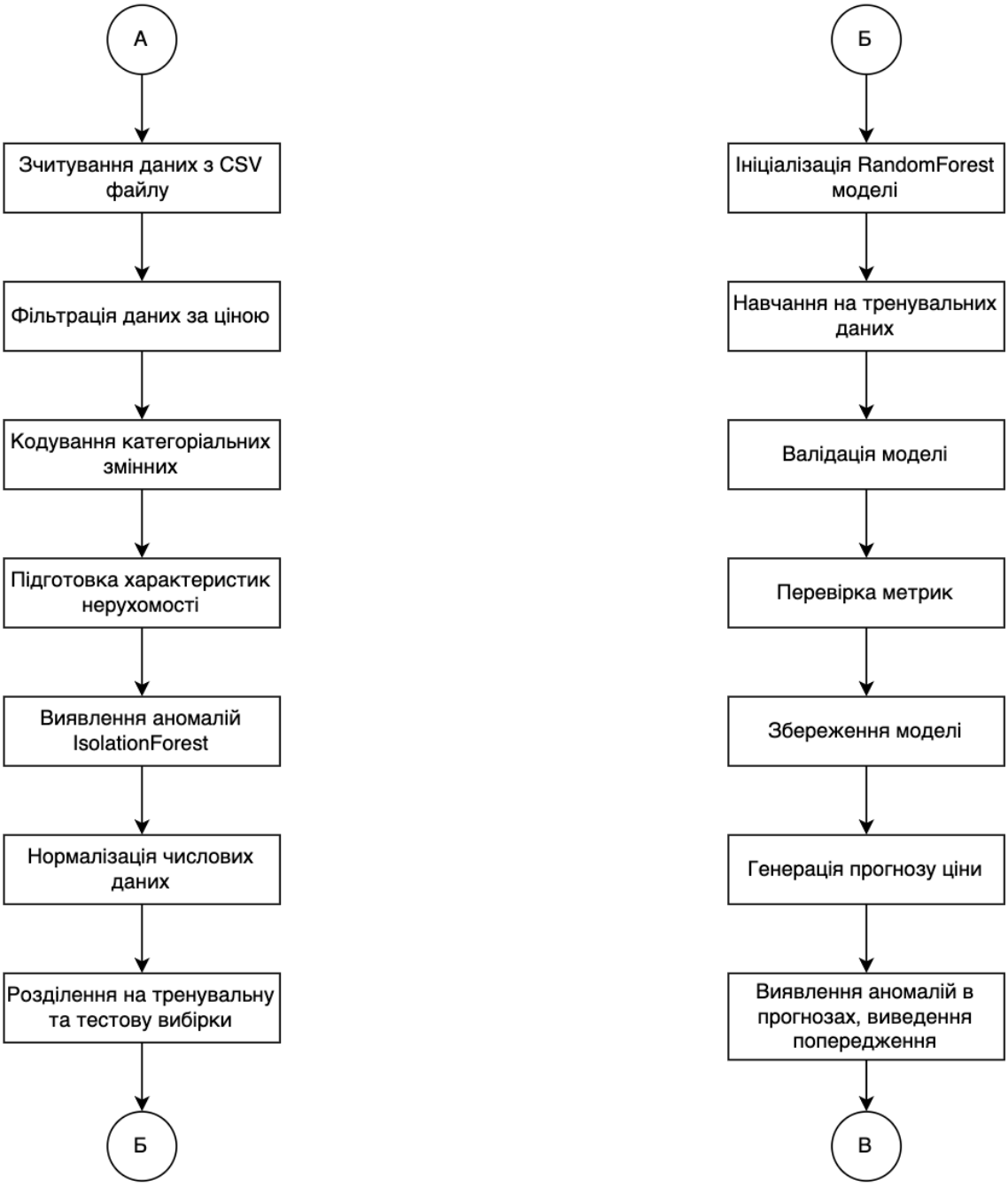

Рисунок Г.8 – Продовження

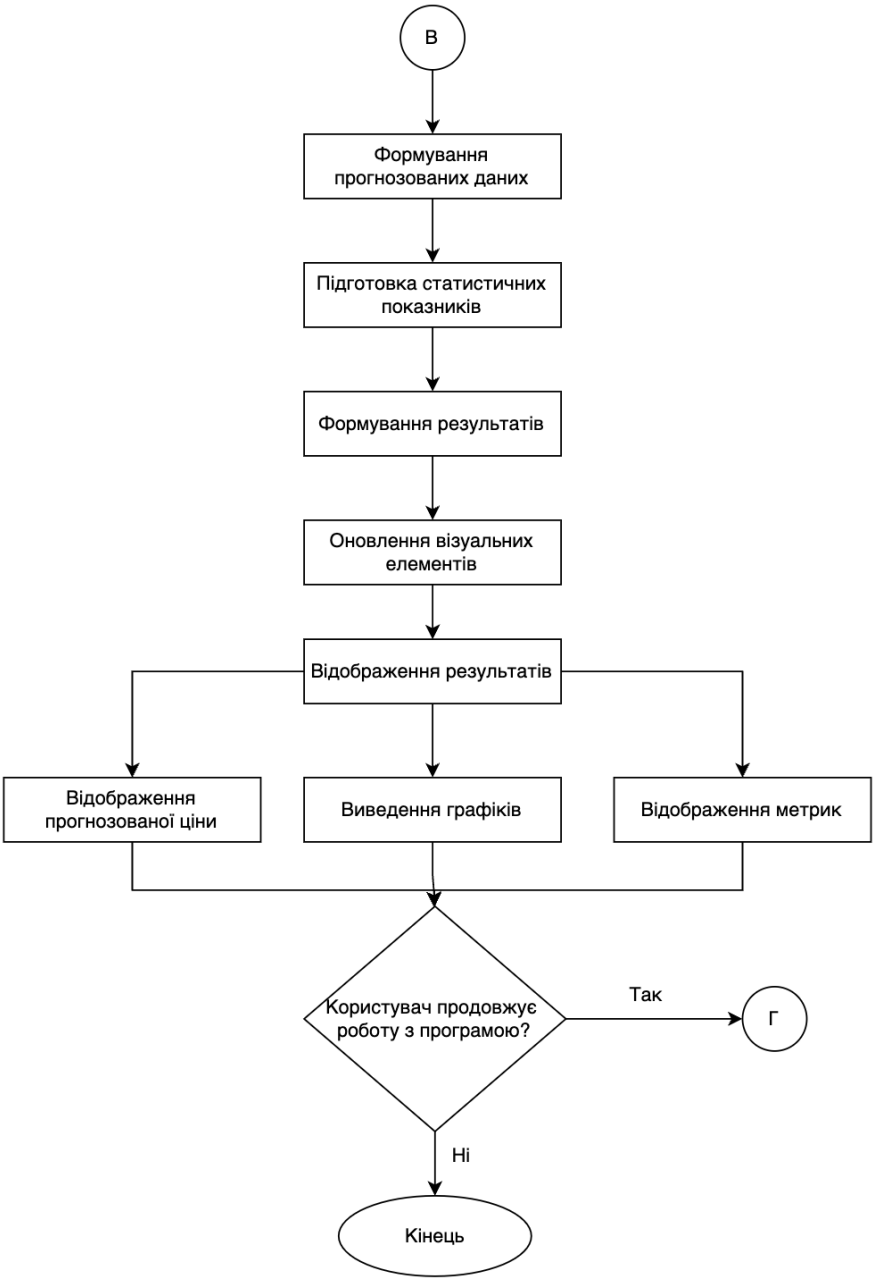

Рисунок Г.9 – Продовження

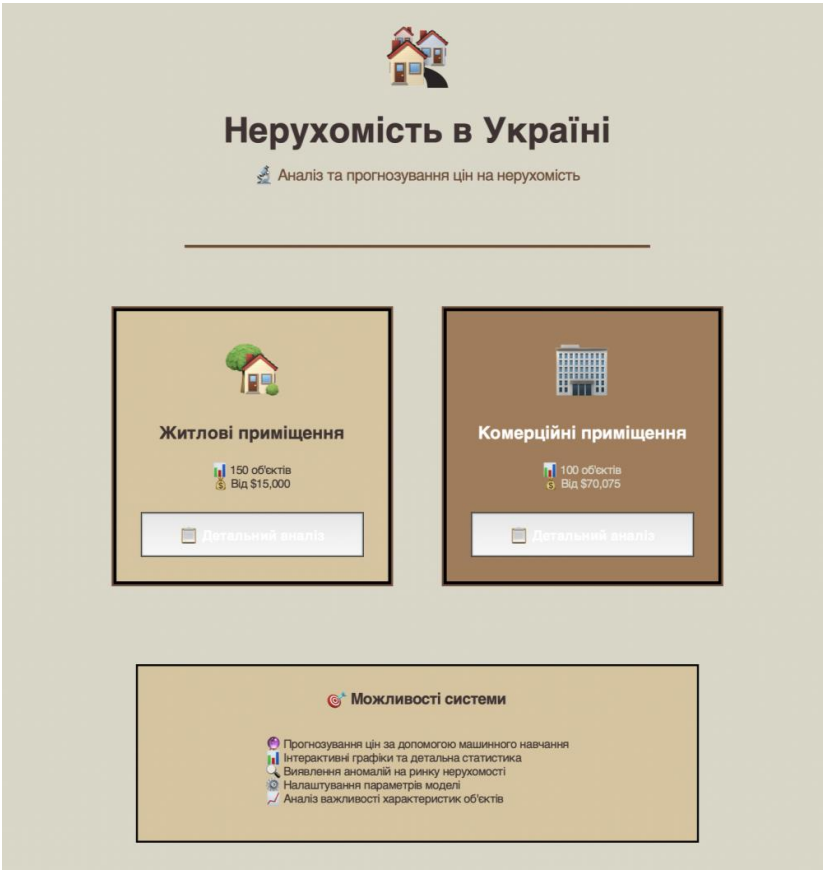

Рисунок Г.10 – Головне меню програмного модуля

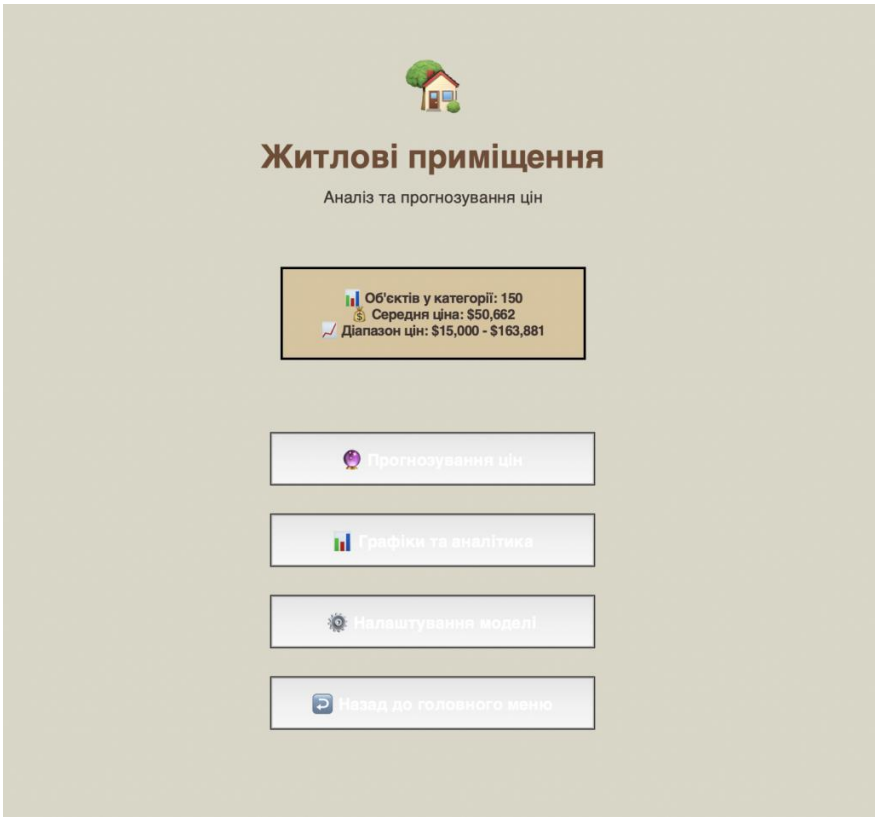

Рисунок Г.11 – Меню житлових приміщень

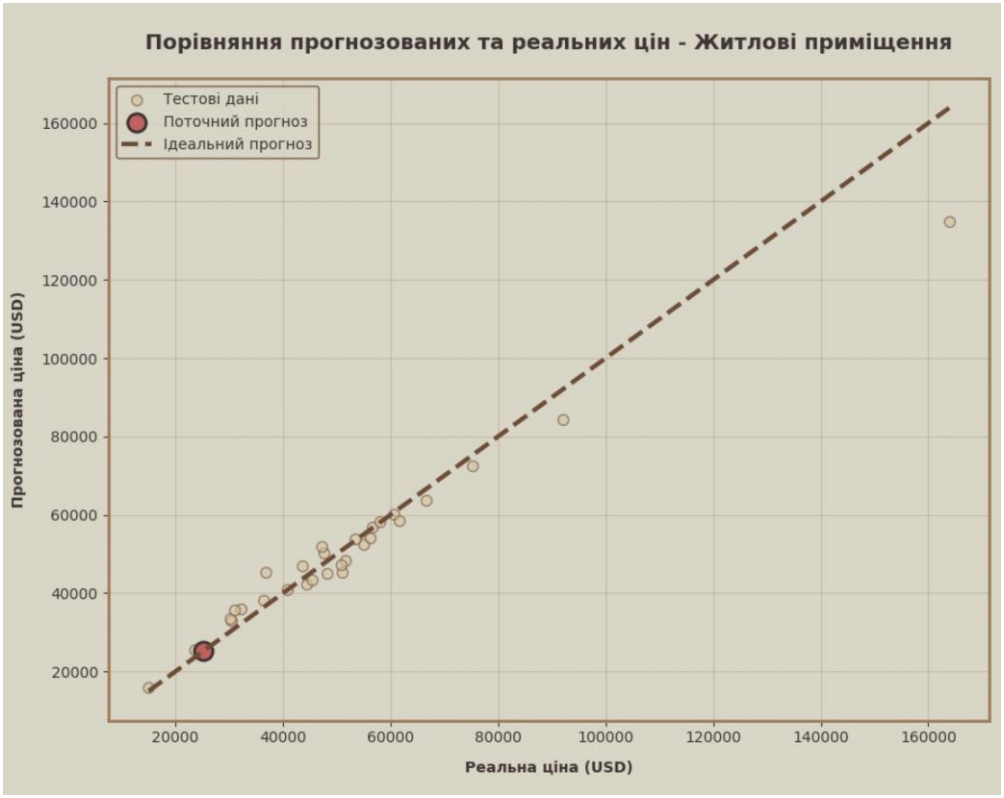

Рисунок Г.12 – Графік порівняння прогнозованих та реальних цін житлової нерухомості

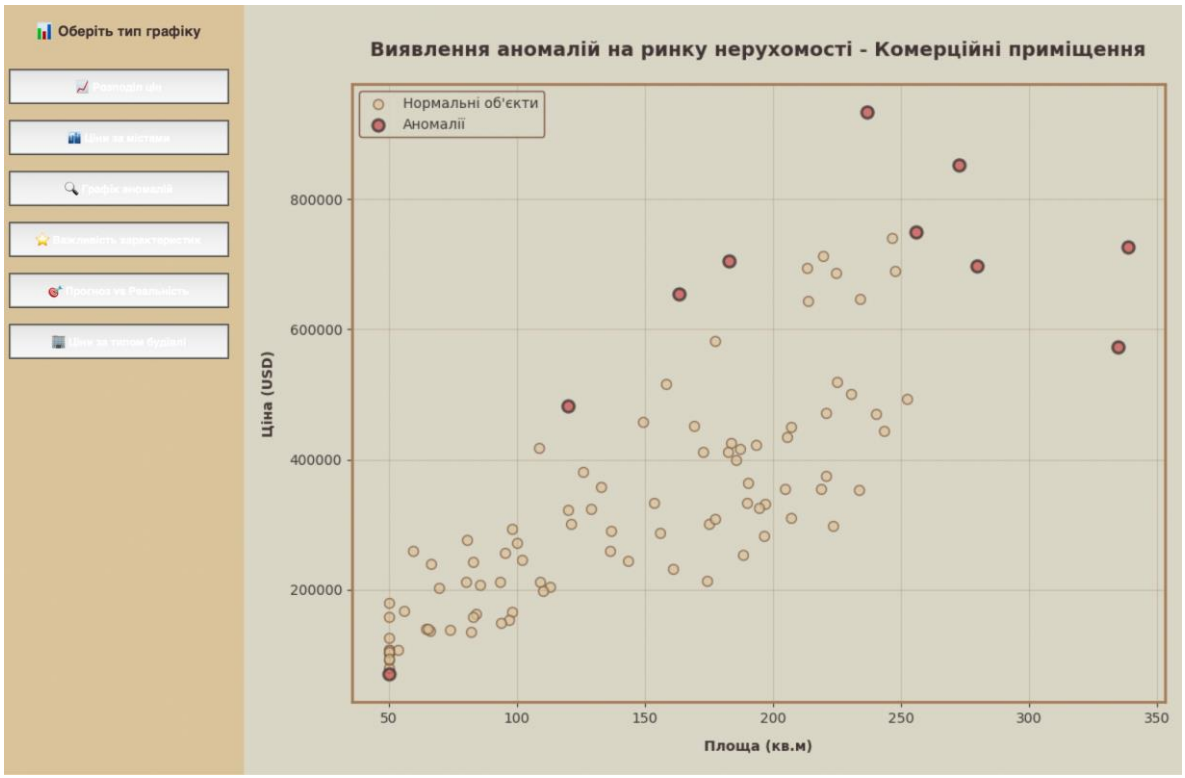

Рисунок Г.13 – Виявлення аномалій на ринку комерційної нерухомості