Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання»

Виконав: студент 2-го курсу, групи
2КН-23м спеціальності 122
«Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Дідик О.С.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Паночишин Ю.М.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., ~~професор~~ каф. КСУ
_____ Биков М. М.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
_____ д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
11.10 2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дідику Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання
керівник роботи: Паночишин., к.т.н., доц. каф. КН
затверджені наказом ВНТУ від «17» 09 2024 року №310
2. Строк подання студентом роботи 11.11. 2024 року
3. Вихідні дані до роботи:

Вхідні дані – Набір даних з платформи Kaggle, який містить інформацію про ціни продажу будинків у місті Бостоні, мова програмування Python, 1460 екземплярів навчальних та 1460 тестових даних датасету.

4. Зміст текстової частини:

Вступ, аналіз предметної області технологій прогнозування цін на будинки, розробка інформаційної технології прогнозування цін на приватні житлові будинки методами машинного навчання, програмна реалізація інформаційної технології, економічна частина, висновки, перелік використаних джерел, додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Граф-схема алгоритму інформаційної технології прогнозування цін на приватні житлові будинки методами машинного навчання, Графіки залежності категоріальних змінних, кореляційна матриця, графіки точності прогнозу моделей.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдан прийн
1-3	Паночишин Ю.М., к.т.н., доц. каф. КН		
4	Адлер О.О. к.т.н., доц. ЕПВМ		

7. Дата видачі завдання 02.09.2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	02.09.24 - 05.09.24	
2	Основні етапи виконання роботи та огляд вхідного набору даних	06.09.24 - 12.09.24	
3	Розроблення інформаційної технології прогнозування цін на приватні житлові будинки	12.09.24 - 18.09.24	
4	Побудування моделей та прогнозування ціни продажу будинків	18.09.24 - 25.09.24	
5	Економічна частина	29.10.24 - 04.11.24	
5	Апробація та/або впровадження результатів дослідження	29.10.24 - 04.11.24	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 - 11.11.24	

Студент _____ Дідик О.С
(підпис) (прізвище та ініціали)

Керівник роботи _____ Паночишин Ю.М
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.8:338+332.628

Дідик О. С. Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма - комп’ютерні науки. Вінниця: ВНТУ, 2024. 111 с

На укр. мові. Бібліогр.:33 назв; рис.:56; табл. 11.

Дана магістерська кваліфікаційна присвячена розробленню інформаційної технології прогнозування цін на приватні житлові будинки методами машинного навчання. Виконано аналіз предметної області, проаналізовані проблеми прогнозування цін на будинки, обґрунтовано актуальність дослідження. Проведено розвідувальний аналіз даних, під час якого було обрано ключові ознаки, які найбільше впливають на ціну будинків.

В роботі використано декілька моделей машинного навчання, виконана їх оптимізація для прогнозування цін на будинки. Створено інформаційну технологію прогнозування цін на приватні житлові будинки методами машинного навчання, яка має точність на 34 % більшу ніж у подібного аналога.

Графічна частина складається з 7 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 759521 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 3.8 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: інформаційна технологія, розвідувальний аналіз даних, прогнозування цін, моделі, машинне навчання.

ABSTRACT

Didyk O. S. Information technology for forecasting prices for private residential buildings using machine learning methods. Master's qualification work on specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2024. 111 p

In Ukrainian speech Bibliography: 33 titles; Fig.:56; table 11.

This master's qualification is devoted to the development of information technology for forecasting prices for private residential buildings using machine learning methods. The analysis of the subject area was performed, the problems of forecasting house prices were analyzed, the relevance of the research was substantiated. An exploratory analysis of the data was carried out, during which the key characteristics that most affect the price of houses were selected. Several machine learning models were used in the work, and their optimization was performed for forecasting house prices. An information technology for forecasting prices for private residential buildings using machine learning methods has been created, which has a forecasting accuracy 34% higher than that of a similar analogue. The graphic part consists of 7 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 1,287,378 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit is calculated, the payback period for the manufacturer is 0.59 years, and the economic effect for the consumer when using this developments.

Keywords: information technology, data intelligence analysis, price forecasting, models, machine learning.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі придбання приватного житла є не лише значною фінансовою інвестицією, але й важливим елементом соціального та економічного розвитку. З ростом попиту на нерухомість та зростаючою конкуренцією на ринку, здатність точно передбачити ціни на житлові будинки стає критично важливою для всіх учасників ринку, включаючи покупців, продавців, агентств нерухомості та інвесторів.

Методи машинного навчання пропонують нові можливості для аналізу великих обсягів даних і виявлення закономірностей, які можуть вплинути на ціни. Вони дозволяють більш точно оцінити вплив різних факторів, таких як місцезнаходження, стан нерухомості, інфраструктура та економічні умови, на ринкову вартість. З використанням передових алгоритмів, таких як регресія, дерева рішень або нейронні мережі, можна досягти вищої точності в прогнозах, що може суттєво знизити ризики при інвестуванні в нерухомість.[1]

Крім того, в умовах швидких змін на ринку нерухомості, спричинених економічними, соціальними та екологічними факторами, створення ефективних інформаційних технологій для прогнозування цін є надзвичайно актуальним. Це дозволить не лише оптимізувати процес купівлі-продажу, але й сприятиме більш раціональному управлінню ресурсами в галузі нерухомості.

Таким чином, дослідження і впровадження інформаційних технологій для прогнозування цін на приватні житлові будинки методом машинного навчання набуває значення як для приватних осіб, так і для професіоналів в галузі нерухомості, що відкриває нові можливості для прийняття обґрунтованих рішень.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного

університету 22 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета і задачі дослідження. Метою даного дослідження є підвищення точності прогнозування цін на приватні житлові будинки за допомогою методів машинного навчання шляхом розробки та впровадження інформаційної технології, що базується на аналізі даних ринку нерухомості.

Для досягнення цієї мети необхідно вирішити такі завдання:

– здійснити аналіз предметної області прогнозування цін на приватні житлові будинки;

– розглянути існуючі аналоги інформаційних технологій прогнозування цін на житлові будинки;

– сформулювати стадії інформаційної технології, розробити алгоритм роботи програмного забезпечення;

– виконати розвідувальний аналіз даних для виявлення патернів та аномалій, а також здійснити обробку вхідних даних для забезпечення їхньої якості;

– розробити 3 методи машинного навчання;

– провести тестування програмного засобу та проаналізувати отримані результати.

Об'єкт дослідження – процес прогнозування цін на приватні житлові будинки за допомогою інтелектуальних комп’ютерних програм.

Предмет дослідження – інформаційна технологія та програмні засоби, які застосовано для аналізу прогнозування цін на приватні житлові будинки та точність їх прогнозування..

Методи дослідження: У роботі використані наступні методи наукових досліджень: системного аналізу, методи машинного навчання, регресійні моделі для прогнозування ціни продажу будинків. У процесі виконання роботи проведено аналіз даних у системі Kaggle мовою програмування Python.

Наукова новизна одержаних результатів. Наукова новизна полягає у тому, що дістала подальший розвиток інформаційна технологія прогнозування ціни продажу будинків, яка відрізняється від відомих детальним налаштуванням гіперпараметрів моделей та їхнім стекувнням з використанням 7 ітерацій укладнення, що дозволяє підвищити точність такого прогнозування у порівнянні з аналогами.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено інформаційну технологію прогнозування цін на приватні житлові будинки.

Запропонована інформаційна технологія сприяє підвищенню точності прогнозування, зокрема:

– розроблено алгоритм роботи програмного забезпечення прогнозування цін на приватні житлові будинки;

– розроблено програмний засіб для прогнозування цін на приватні житлові будинки методами машинного навчання.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю виконання дослідження, проведеним аналізом та обробкою даних, тестуванням інформаційної технології прогнозування цін на приватні будинки. Адекватність розробленої інформаційної технології підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз та обробка вхідних даних, розробка інформаційної технології прогнозування цін на приватні житлові будинки.

Апробація результатів роботи. Результати досліджень апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи-2024», Вінниця, жовтень 2024 – червень 2025 року [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГНОЗУВАННЯ ЦІН НА ПРИВАТНІ ЖИТЛОВІ БУДИНКИ

1.1 Постановка задачі прогнозування цін на житлові будинки

Об’єктом дослідження даної роботи є прогнозування цін на житлову нерухомість. Це складна система, що враховує широкий спектр факторів, які впливають на вартість житла. Дослідження цього питання допомагає зрозуміти, які елементи впливають на коливання цін на ринку нерухомості і як їх прогнозувати.

Одне з найпоширеніших питань серед покупців і продавців нерухомості – «Що саме визначає ціну будинку?» Відповідь на це питання містить безліч аспектів. Наприклад, часто можна почути фразу «все залежить від місця», і це справедливо, хоча місце – це лише один із багатьох факторів.

Основні чинники, що впливають на ціни на житло, включають такі аспекти:

Попит і пропозиція. Кількість пропозицій на ринку і число покупців, що шукають житло, значною мірою визначають рівень цін. При великому попиті ціни зростають, тоді як при його зниженні вартість житла зменшується.

Стан нерухомості. Стан, у якому знаходиться будинок, значно впливає на його вартість. Житло, що не потребує додаткових вкладень, може мати значно вищу ціну, ніж об’єкт, що потребує ремонту.

Наявність шкіл. Для сімей з дітьми близькість до якісних навчальних закладів може мати вирішальне значення і збільшувати привабливість житла в очах потенційних покупців [2].

Інфраструктура. Зміни в інфраструктурі, наприклад, будівництво нових доріг чи залізничних ліній, можуть мати як позитивний, так і негативний вплив на вартість нерухомості залежно від рівня доступності і зручності для мешканців.

Розмір та планування. Загальна площа, кількість кімнат і, зокрема, наявність додаткових зручностей, як-от власні ванні кімнати, можуть суттєво підвищити ціну будинку.

Наявність місць для паркування. Наявність гаража, під'їзної дороги чи іншого місця для паркування може збільшити цінність нерухомості. Кількість паркомісць також є важливим фактором, особливо для родин з кількома автомобілями.

Потенціал для покращень. Деякі покупці надають перевагу будинкам із можливістю для покращень, що дозволяє їм адаптувати простір відповідно до своїх уподобань. Доречність додаткових споруд, як-от альтанки чи зимові сади, також може впливати на ціну, але потребує уважного планування.

Ці та інші фактори, такі як екологічність району чи соціально-економічний статус місцевості, визначають кінцеву вартість нерухомості та її привабливість для різних категорій покупців[3].

Отже, потрібно розробити інформаційну технологію, де будуть враховуватись всі ключові фактори, які впливають на ціну будинків, щоб покращити точність прогнозування.

Для вирішення проблеми, описаної в даному розділі, необхідно вирішити наступні завдання:

1. Провести розвідувальний аналіз;
2. Спроектувати структуру інформаційної технології;
3. Програмно реалізувати інформаційну технологію;
4. Провести тестування інформаційної технології.

1.2 Огляд методів машинного навчання і класифікації

Машинне навчання (Machine Learning, ML) — це важливий напрямок штучного інтелекту (AI), який дозволяє комп'ютерам автоматично навчатися на основі даних і попереднього досвіду, визначати закономірності та робити

прогнози з мінімальним втручанням людини. Ця технологія стає все більш популярною завдяки своїй здатності обробляти великі обсяги даних, ефективно працювати зі складними залежностями та адаптуватися до змінних умов.

Методи машинного навчання забезпечують автономну роботу комп’ютерних систем, що дозволяє їм навчатися, вдосконалюватися та адаптуватися без необхідності прямого втручання. Алгоритми ML використовують обчислювальні методи для навчання на основі реальних даних, і цей процес ітеративно покращує їх продуктивність із додаванням нових зразків. Наприклад, підхід глибокого навчання (deep learning), який є піддисципліною ML, дозволяє комп’ютерам імітувати складні людські риси, такі як навчання на прикладах, забезпечуючи при цьому вищу точність у порівнянні зі звичайними алгоритмами [4].

Сучасний розвиток технологій, зокрема великих даних (Big Data), Інтернету речей (IoT) та обчислювальних платформ, підкреслює необхідність використання ML для вирішення завдань у різних галузях, таких як фінанси, медицина, транспорт, промисловість, розпізнавання зображень та голосу. Для прогнозування цін на приватні житлові будинки ML пропонує особливо корисні можливості завдяки своїм ключовим перевагам:

- обробка великих обсягів даних - ML здатне ефективно працювати з великою кількістю інформації, включаючи історичні дані про ціни, характеристики об'єктів нерухомості, економічні показники тощо.
- виявлення складних взаємозв’язків - алгоритми машинного навчання можуть знаходити нелінійні та багатовимірні залежності між факторами, що впливають на ціни.
- гнучкість та адаптивність - ML-моделі легко адаптуються до змін на ринку нерухомості, наприклад, коливань попиту чи нових економічних умов.
- покращення точності прогнозів - у порівнянні з традиційними статистичними методами, машинне навчання демонструє високу точність та ефективність у прогнозуванні.

- автоматизація процесів - використання ML дозволяє зменшити потребу у ручній роботі, що значно прискорює процес оцінки.

Як працює машинне навчання? Алгоритми машинного навчання формуються на навчальному наборі даних для створення моделі. Коли нові вхідні дані вводяться в навчений алгоритм, він використовує розроблену модель для прогнозування (рис. 1.1).

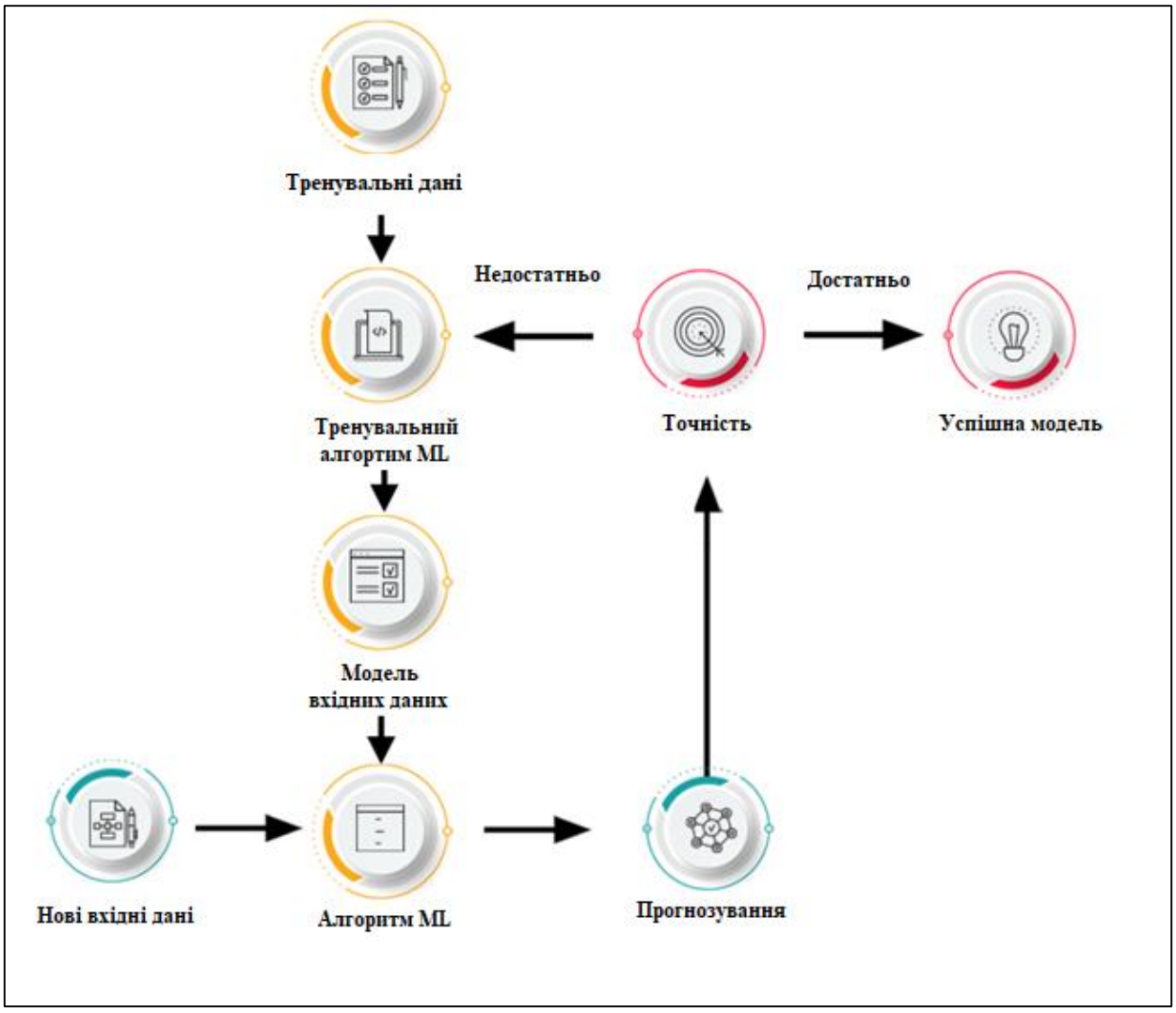

Рисунок 1.1 – Схема створення та застосування моделі машинного навчання

Далі прогноз перевіряється на точність. На основі його точності алгоритм ML або розгортається, або тренується повторно на розширеному навчальному наборі даних, поки не буде досягнута бажана точність [4].

Типи машинного навчання. Алгоритми машинного навчання можна навчати різними способами, кожен з яких має свої переваги та недоліки. Виходячи з цих методів і способів навчання, машинне навчання можна розділити на чотири основні типи (рис. 1.2).

Рисунок 1.2 – Типи машинного навчання

Кероване машинне навчання (навчання з учителем). Цей тип ML передбачає контроль, коли машини навчаються на маркованих наборах даних і можуть передбачати результати на основі отриманих знань. Маркований набір даних вказує на те, що деякі вхідні та вихідні параметри вже відображені. Таким чином, машина навчається на вхідних даних і відповідному виході. На наступних етапах створюється пристрій для прогнозування результату з використанням тестового набору даних.

Некероване машинне навчання (навчання без учителя). Безконтрольне навчання – це метод навчання, який позбавлений контролю. Тут машина навчається на немаркованому наборі даних і може передбачати результати без

будь-якого контролю. Алгоритм навчання без нагляду має на меті згрупувати несортований набір даних на основі схожості, відмінностей та закономірностей вхідних даних [5].

Напівавтоматичне або напівконтрольоване навчання включає в себе характеристики як керованого, так і некерованого машинного навчання. Воно використовує комбінацію мічених і немічних наборів даних для навчання своїх алгоритмів. Використовуючи обидва типи наборів даних, напівкероване навчання долає недоліки згаданих вище варіантів.

Навчання з підкріпленням – це процес, заснований на зворотному зв'язку. Тут AI-компонент автоматично оцінює навколишнє середовище за методом проб і помилок, діє, вчиться на власному досвіді та покращує продуктивність. Компонент винагороджується за кожну правильну дію і карається за кожен неправильний крок. Таким чином, компонент навчання з підкріпленням спрямований на максимізацію винагороди за правильні дії.

На відміну від керованого навчання, у навчанні з підкріпленням немає маркованих даних, і агенти навчаються лише на власному досвіді. Розглянемо відеоігри. Тут гра задає середовище, а кожен рух агента підкріплення визначає його стан. Агент має право отримувати зворотній зв'язок через покарання та винагороди, впливаючи таким чином на загальний результат гри. Кінцевою метою агента є досягнення високого результату.

Навчання з підкріпленням застосовується в різних галузях, таких як теорія ігор, теорія інформації та мультиагентні системи. Навчання з підкріпленням поділяється на два типи методів або алгоритмів:

- з позитивним підкріпленням – це стосується додавання підкріплювального стимулу після певної поведінки агента, що підвищує ймовірність того, що поведінка може повторитися в майбутньому, наприклад, додавання винагороди після певної поведінки;
- з негативним підкріпленням. Навчання з негативним підкріпленням стосується посилення певної поведінки, яка дозволяє уникнути негативного результату [5].

Класифікація – це завдання, яке вимагає використання алгоритмів машинного навчання, що вчиться присвоювати мітки класів прикладам з проблемної області. Простий і зрозумілий приклад – класифікація електронних листів як "спам" або "не спам".

Основні види класифікації в машинному навчанні включають:

- бінарна класифікація – це процес класифікації об'єктів у дві категорії, наприклад, так / ні, позитивний / негативний, правда / брехня і т.д.;
- багатокласова класифікація – це процес класифікації об'єктів на більше, ніж дві категорії. Цей тип класифікації може використовуватися, наприклад, для класифікації зображень різних видів тварин, визначення типу листя на деревах, розпізнавання кольорів і т.д.;
- мультикласова класифікація – це розширена версія багатокласової класифікації, яка дозволяє класифікувати об'єкти в більш ніж один клас одночасно. Цей тип класифікації може використовуватися для класифікації зображень з декількома об'єктами на них, наприклад, зображень людей з кількома предметами одягу;
- ієрархічна класифікація – це процес класифікації об'єктів, що відбувається на основі їх відношення до інших об'єктів. Об'єкти групуються у більші категорії на основі спільних характеристик, що дозволяє побудувати ієрархію класів;
- ансамбль класифікаторів – це підхід до класифікації, при якому використовуються кілька моделей машинного навчання для отримання більш точних результатів [5].

З огляду на вищезазначену інформацію можна сказати, що машинне навчання є доцільним та перспективним інструментом для прогнозування цін на приватні житлові будинки. Його застосування дозволяє вирішувати складні аналітичні задачі, підвищуючи ефективність і точність прогнозів, що є критично важливим для ринкової аналітики та прийняття управлінських рішень.

1.3 Огляд відомих аналогів прогнозування цін на приватні житлові будинки

Існують сучасні автоматизовані системи для пошуку та оцінки вартості житлової нерухомості, такі як будинки або квартири, на основі вхідних параметрів. Ці платформи використовують різноманітні фактори, включаючи географічне розташування, площу, кількість кімнат, рік побудови, стан нерухомості та інші критерії для точнішого розрахунку ринкової вартості об'єкта.

Окрім базової оцінки, такі системи також надають можливість відстежувати динаміку зміни цін за певний період часу. Це дозволяє користувачам побачити, як змінювалась вартість нерухомості в різні моменти, враховуючи коливання на ринку, сезонність або інші економічні чинники. Аналізуючи такі дані, можна отримати важливу інформацію про те, чи є поточна ціна завищеною або заниженою відносно середніх ринкових показників.

Такі системи часто використовуються покупцями, продавцями та інвесторами для прийняття обґрунтованих рішень щодо купівлі або продажу нерухомості. Завдяки наявності історичних даних про ціни, вони можуть допомогти визначити найкращий момент для здійснення операції, мінімізуючи ризики переплати або продажу за заниженою вартістю.

Крім того, деякі з цих платформ надають прогнози щодо майбутніх змін цін, базуючись на поточних тенденціях ринку та економічних умовах. Це дозволяє не тільки оцінювати актуальну вартість, але й планувати довгострокові інвестиції, орієнтуючись на прогнозовані зміни вартості нерухомості.

Rightmove.co.uk — це один з найбільших і найвідоміших вебсайтів у Великій Британії для пошуку нерухомості. Заснований у 2000 році, Rightmove став лідером на ринку нерухомості завдяки своїй масштабній базі даних, де розміщуються оголошення як приватних осіб, так і агентств нерухомості.

Основні характеристики Rightmove:

1. Платформа для пошуку нерухомості: Rightmove пропонує можливість шукати житлову та комерційну нерухомість для купівлі або оренди. Вебсайт охоплює увесь ринок нерухомості у Великій Британії — від квартир і будинків до комерційних об’єктів.

2. Широкий вибір оголошень. На сайті розміщуються пропозиції від агентств нерухомості, забудовників і власників. Rightmove співпрацює з тисячами агентств у країні, що робить його важливим інструментом для пошуку нерухомості.

3. Зручний інтерфейс. Сайт дозволяє користувачам налаштовувати фільтри пошуку за різними параметрами — місцезнаходження, ціна, кількість кімнат, тип нерухомості тощо. Це допомагає легко знайти пропозиції, які відповідають конкретним потребам.

4. Аналітика ринку. Rightmove також надає інструменти аналітики та звіти про тенденції на ринку нерухомості. Це включає дані про середні ціни, коливання цін за певний період і прогнози.

5. Мобільний додаток. Компанія пропонує мобільний додаток для зручного пошуку нерухомості в будь-який час і з будь-якого місця.

6. Міжнародний аспект. Хоча сайт переважно сфокусований на ринку Великої Британії, Rightmove також пропонує міжнародні пропозиції для тих, хто шукає нерухомість за межами країни [6].

Zillow — це провідна онлайн-платформа для пошуку та оцінки нерухомості, головним чином у Сполучених Штатах. Заснована у 2006 році, компанія швидко стала одним із найпопулярніших ресурсів для покупців, продавців, орендодавців та агентів з нерухомості. Zillow надає широкий спектр інструментів для оцінки вартості нерухомості, аналізу ринку та пошуку будинків і квартир.

Основні характеристики Zillow:

1. Оцінка нерухомості (Zestimate). Zestimate — це один із найвідоміших інструментів платформи, який надає оцінку ринкової вартості

житла на основі алгоритму, що враховує різні фактори. Алгоритм аналізує такі параметри, як історія продажів, місцезнаходження, площа будинку, кількість кімнат та загальний стан нерухомості. Хоча Zestimate не є офіційною оцінкою для юридичних операцій, його часто використовують як орієнтир при купівлі або продажу нерухомості.

2. Широка база даних оголошень. Zillow містить мільйони активних оголошень про продаж та оренду житла. Користувачі можуть шукати нерухомість за різними критеріями, такими як ціна, площа, кількість кімнат, місцезнаходження, рік побудови тощо. Також платформа відображає будинки, які не виставлені на продаж, але оцінює їх вартість та надає можливість зберігати їх для відстеження змін ціни.

3. Аналітика ринку. Zillow надає інструменти для аналізу ринку нерухомості. Користувачі можуть переглядати дані про середні ціни в певних районах, тенденції змін цін, а також прогнозувати можливі зміни вартості нерухомості на основі наявних даних. Це особливо корисно для інвесторів і рієлторів, які хочуть приймати зважені рішення, базуючись на аналізі ринку.

4. Мобільні додатки. Zillow пропонує зручні мобільні додатки для смартфонів, що дозволяє користувачам шукати нерухомість, оцінювати її вартість і відстежувати зміни цін у будь-який час і в будь-якому місці.

5. Інтерактивні карти та додаткова інформація. Платформа пропонує інтерактивні карти, які відображають нерухомість з деталями про транспорт, інфраструктуру, школи, парки та інші важливі об'єкти, що можуть вплинути на рішення про купівлю або оренду. Zillow також надає огляди від місцевих жителів, що дозволяє дізнатися більше про якість життя в певному районі.

6. Інструменти для продавців та орендодавців. Власники нерухомості можуть легко створювати оголошення про продаж чи оренду на платформі Zillow, використовуючи зручний інтерфейс. Zillow також допомагає з організацією відкритих переглядів, оцінкою попиту та взаємодією з потенційними покупцями або орендарями.

7. Fintech-послуги. Zillow пропонує фінансові інструменти, такі як іпотечні калькулятори, які допомагають користувачам оцінити їх фінансові можливості та отримати попереднє схвалення на кредит [7].

Address.ua — це український онлайн-ресурс, що спеціалізується на пошуку та оцінці нерухомості. Платформа стала популярною серед покупців, орендодавців і ріелторів, пропонуючи різноманітні послуги для роботи з ринком нерухомості в Україні.

Основні характеристики Address.ua:

1. Широкий вибір оголошень. Address.ua містить великий каталог оголошень про продаж і оренду житла, включаючи квартири, будинки, комерційну нерухомість та земельні ділянки. Користувачі можуть шукати нерухомість за різними критеріями, такими як ціна, розташування, площа, тип нерухомості тощо.

2. Функція оцінки вартості. Платформа пропонує інструменти для оцінки ринкової вартості нерухомості, що дозволяє користувачам отримувати приблизну оцінку ціни об'єктів на основі аналізу аналогічних пропозицій у даному регіоні.

3. Аналіз ринку. Address.ua надає користувачам інформацію про тенденції на ринку нерухомості, включаючи дані про середні ціни на житло в різних містах і районах, що дозволяє споживачам приймати обґрунтовані рішення.

4. Професійні послуги. Платформа співпрацює з рієлторами та агентствами нерухомості, що дає можливість користувачам отримувати професійні консультації та допомогу в купівлі, продажу або оренді нерухомості.

5. Зручний інтерфейс. Address.ua має зрозумілий та інтуїтивно зрозумілий інтерфейс, що робить пошук нерухомості легким та зручним. Користувачі можуть фільтрувати результати пошуку та зберігати вподобані оголошення для подальшого перегляду.

6. Мобільні додатки. Платформа може пропонувати мобільні додатки для зручності користувачів, які шукають нерухомість на ходу.

7. Інформаційні ресурси. Address.ua також публікує статті та рекомендації про купівлю, продаж та оренду нерухомості, що може бути корисним для користувачів, які не мають досвіду в цих питаннях [8].

Отже, аналіз розглянутих платформ показує, що кожна з них має певні можливості для оцінки вартості житла. Проте вони включають лише основні фактори які впливають на ціну. Тому, для досягнення більш точної оцінки необхідно вдосконалювати методи прогнозування, залучаючи ширший набір факторів, які впливають на ціну будинків, це дозволить підвищити точність прогнозування та ефективність моделей машинного навчання.

Як відомий аналог розглянемо та опишемо ноутбук у системі Kaggle під назвою «House Pricing Predictions», автором якого є Pratik Barua [9].

Для початку автор проводить розвідувальний аналіз даних та перевіряє дані на нормальний розподіл (рис. 1.3).

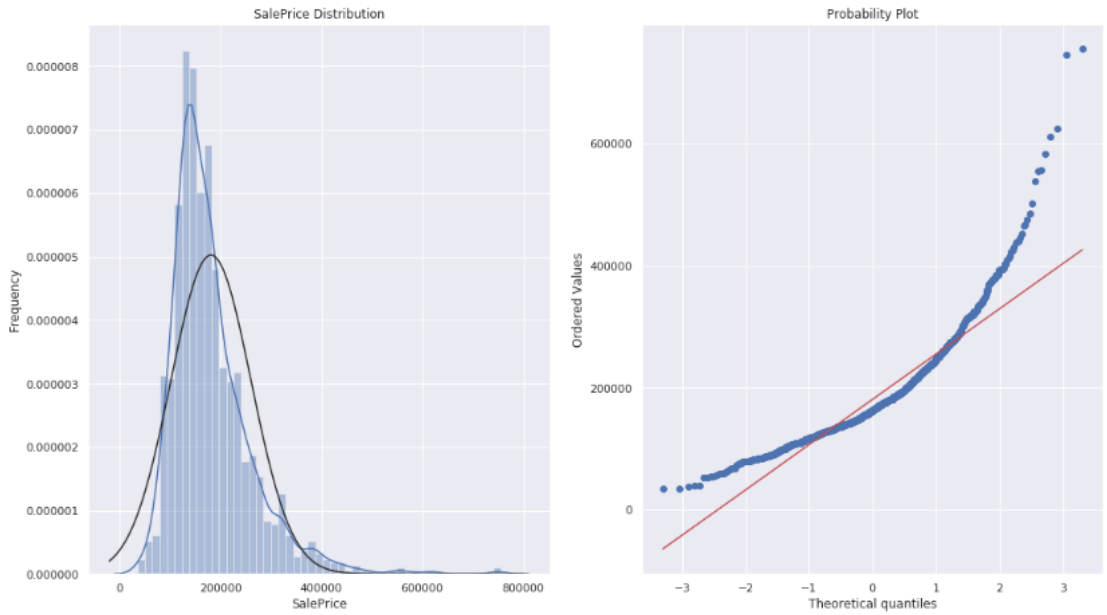

Рисунок 1.3 – Розподіл цін та цільова змінна

Автор застосовує логарифмічне перетворення для того щоб дані були нормально розподіленими. Нижче на рисунку 1.4 наведені графіки після логарифмічного перетворення.

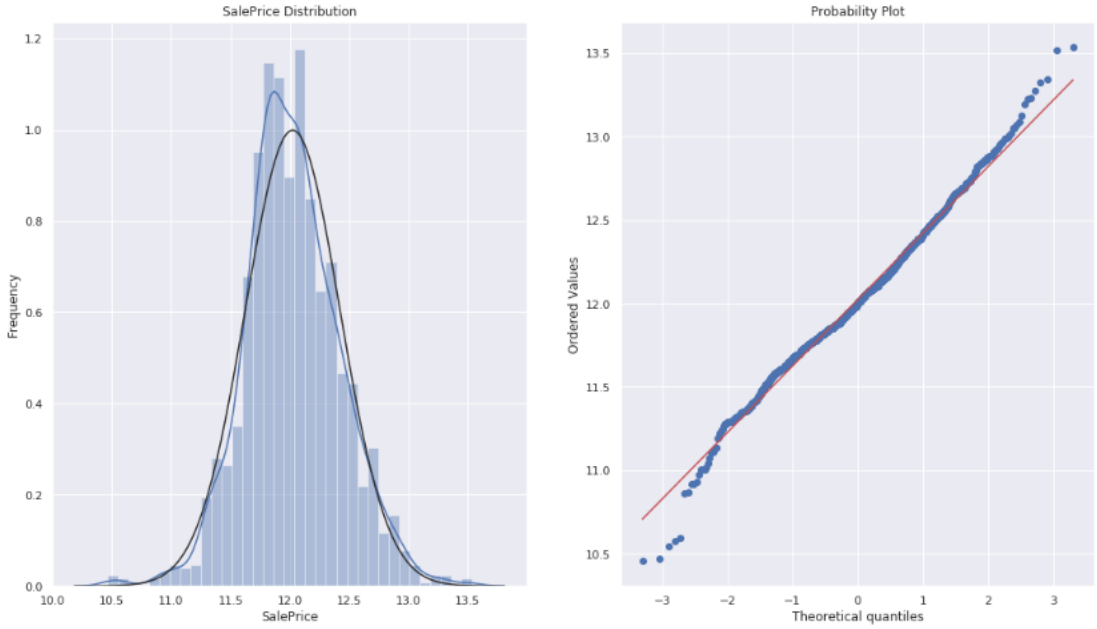

Рисунок 1.4 – Розподіл цін та цільова змінна після перетворення

Далі виконується двох факторний аналіз для перевірки відхилень значень (рис. 1.5). Двохфакторний аналіз у контексті цього графіка використовується для того, щоб зрозуміти взаємозв'язок між двома змінними – ціною продажу будинку (SalePrice) і різними факторами (наприклад, LotFrontage, LotArea, OverallQual тощо).

Основна мета аналізу:

1. Виявлення кореляцій: За допомогою двохфакторних графіків можна побачити, які змінні найбільше впливають на ціну будинку. Наприклад, на графіку видно, що змінні OverallQual і TotalBsmtSF мають очевидний позитивний зв'язок із ціною.

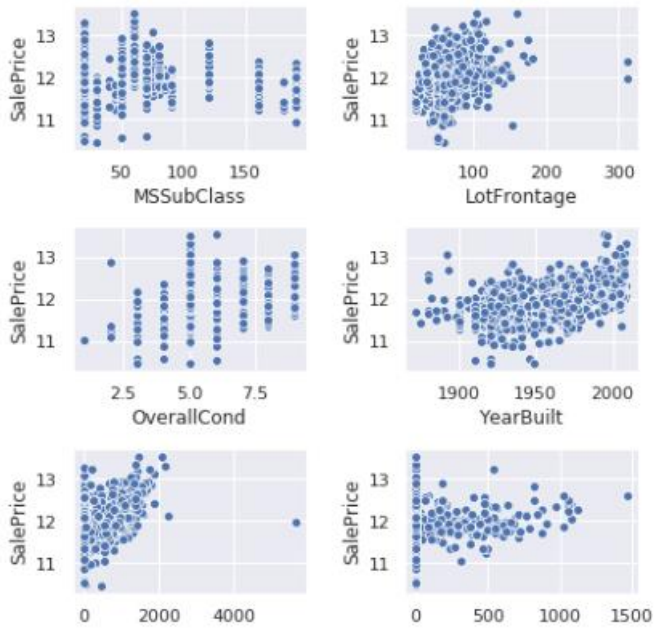

Рисунок 1.5 – Графіки двохфакторного аналізу

2. Виявлення відхилень: Графіки допомагають знайти аномальні значення або відхилення, які можуть вказувати на помилки в даних або незвичні будинки з дуже високою або низькою ціною. Це корисно для подальшого очищення та обробки даних.

3. Аналіз структури даних: Візуалізація допомагає зрозуміти структуру даних, наприклад, наявність лінійних або нелінійних залежностей, що дозволяє вибрати кращий метод моделювання (лінійні чи нелінійні моделі).

4. Вибір важливих ознак: Двохфакторний аналіз допомагає обрати найбільш значущі ознаки для моделювання, залишаючи ті, які найбільше впливають на ціну.

Після виявлення пропусків, автор застосував такі методи заповнення відсутніх даних:

- для числових значень він застосував середнє чи медіану;
- для категоріальних значень використовував найбільш поширене значення або нові категорії (наприклад, "None")(рис. 1.6).

```
In [23]: for col in ('FireplaceQu', 'Fence', 'Alley', 'MiscFeature', 'PoolQC', 'MSSubClass'):
             dataset[col] = dataset[col].fillna('None')
```

Рисунок 1.6 – Заповнення пропущених значень значенням "None".

Далі автор поділив дані для тренування і тестування, використовуючи метод train_test_split з бібліотеки Sklearn, щоб уникнути перенавчання. Після цього автор почав навчання моделей для застосування подальшого прогнозування.

Результат точності прогнозування моделей за метрикою RMSE занесено до таблиці 1.1.

Таблиця 1.1 – Результат точності прогнозування моделей

Назва моделі	Похибка за метрикою RMSE
LightGBM	0.119
XGBoost	0.114
Lasso	0.112
ElasticNet	0.112

1.4 Висновок до розділу 1

Отже, в першому розділі було визначено основні аспекти дослідження в сфері прогнозування цін на житлову нерухомість. Проаналізовано фактори, які суттєво впливають на цінність об’єктів житлової нерухомості, зокрема попит і пропозицію, місце розташування, стан об'єкта, інфраструктуру, а також потенціал для покращень.

Розглянуто існуючі автоматизовані системи для оцінки вартості житла, які використовують ці параметри для підвищення точності прогнозів, зокрема Rightmove, Zillow та Address.ua, що є популярними платформами у

Великобританії, США та Україні відповідно. Водночас, через обмежений перелік врахованих параметрів, ці платформи не завжди забезпечують достатню точність прогнозів, що відкриває перспективу для вдосконалення методів та залучення більшої кількості релевантних ознак.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ЦІН НА ПРИВАТНІ ЖИТЛОВІ БУДИНКИ

2.1 Обґрунтування вибору методів машинного навчання для прогнозування цін на приватні житлові будинки

У першому розділі було показано, що для задачі прогнозування цін на приватні житлові будинки варто використовувати методи машинного навчання. Проте машинне навчання пропонує безліч моделей для вирішення завдань прогнозування, кожна з яких має свої переваги та підходить для різних типів даних і завдань.

Kernel Ridge Regression (KRR) є методом машинного навчання для прогнозування, який об'єднує принципи лінійної регресії з використанням ядерної функції для моделювання нелінійних взаємозв'язків. Ядерна функція допомагає перетворити вхідні дані у простір, де вони стають лінійно розділеними. Завдяки регуляризації KRR контролює розмір коефіцієнтів регресії, зменшуючи ризик перенавчання. Це робить KRR корисним для широкого кола задач, зокрема прогнозування цін на житло, фінансових показників та розпізнавання образів. Його гнучкість та здатність до моделювання складних залежностей робить його ефективним для багатьох завдань прогнозування [11].

Lasso (Least Absolute Shrinkage and Selection Operator) також є методом для задач регресії, який поєднує прогнозування з регуляризацією. Він застосовує L1-регуляризацію, завдяки якій певні коефіцієнти стають нульовими, що спрощує модель і робить її більш інтерпретованою. Під час навчання Lasso мінімізує функцію втрату разом із регуляризаційним компонентом, що допомагає контролювати перенавчання та обирати важливі змінні [12].

Градієнтний бустинг є методом машинного навчання, що використовує ансамблювання слабких моделей (часто дерев рішень) для створення потужної прогностичної моделі. У цьому методі кожна наступна модель виправляє помилки попередньої, застосовуючи градієнтний спуск для покращення результатів. Градієнтний бустинг можна використовувати як для регресії, так і для класифікації, що робить його універсальним інструментом у машинному навчанні [13].

ElasticNet — це метод, який поєднує L1- та L2-регуляризації, дозволяючи контролювати перенавчання та одночасно відбирати значущі змінні. Він допомагає розподілити ваги змінних між регуляризаційними компонентами, оптимізуючи модель для завдань прогнозування на числових даних. ElasticNet є корисним для задач, де обидві регуляризаційні складові мають значення, а також для створення моделей, стійких до перенавчання.

Дерева рішень(Random Forest) — один з основних алгоритмів для регресії та класифікації, який працює шляхом створення послідовних рішень на основі характеристик вхідних даних. Дерева рішень забезпечують простоту інтерпретації, здатність враховувати як числові, так і категоріальні змінні, та можуть бути розширені до ансамблів для покращення точності прогнозування.

Метод K-Nearest Neighbors (KNN) прогнозує нові значення, базуючись на найближчих сусідах у багатовимірному просторі характеристик. Цей метод простий, інтерпретований і дозволяє враховувати вагу сусідів залежно від їхньої відстані до нового зразка, що робить його ефективним для невеликих наборів даних [14].

LightGBM (Light Gradient Boosting Machine) - це вдосконалена версія градієнтного бустингу, оптимізована для роботи з великими наборами даних. LightGBM підтримує категоріальні змінні, є швидшим за традиційні методи та здатен обробляти складні залежності в даних. Завдяки цьому, LightGBM є ефективним інструментом для прогнозування цін на житло, особливо у випадках великих обсягів даних.

CatBoost – це алгоритм градієнтного бустингу, розроблений для обробки даних, які включають багато категоріальних змінних. CatBoost автоматично кодує такі змінні без необхідності їх попередньої трансформації. Ця модель забезпечує високу точність і стійкість до переобучення, що робить її корисною для задач прогнозування нерухомості, де категоріальні змінні (наприклад, тип району, клас будинку) відіграють важливу роль.

Artificial Neural Networks (ANN) - штучні нейронні мережі є потужним інструментом для прогнозування, особливо коли дані є складними і мають багато параметрів. ANN здатні моделювати складні нелінійні залежності, використовуючи кілька шарів нейронів. Для прогнозування цін на житло ANN можуть обробляти великі обсяги даних, включаючи текстову, числову та категоріальну інформацію. Залежно від складності даних, глибина мережі може бути налаштована для досягнення максимальної точності [15].

Bayesian Regression:- байєсівські методи використовують імовірнісний підхід до моделювання. Bayesian Regression враховує невизначеність в даних і прогнозах, що дозволяє створити модель, яка не тільки прогнозує ціни, але й дає оцінку ймовірності кожного прогнозу.

Recurrent Neural Networks (RNN) та Long Short-Term Memory (LSTM). Ці моделі використовуються для задач, де дані мають часові залежності, наприклад, для аналізу цін на житло в часі. RNN і LSTM здатні враховувати тренди, сезонність і коливання цін, що робить їх корисними для довгострокового прогнозування [16].

Завдяки здатності до адаптації та точності, ці моделі є доцільними інструментами для розробки інформаційної технології прогнозування цін на приватні житлові будинки.

Для вирішення задачі прогнозування цін на приватні житлові будинки було обрано моделі LASSO, Elastic Net Regression, Kernel Ridge Regression, та Gradient Boosting Regression (XGBoost). Вибір цих методів зумовлений їхніми унікальними перевагами, які відповідають вимогам даного дослідження. Модель LASSO дозволяє здійснювати автоматичний відбір змінних завдяки

L1-регуляризації, що важливо для спрощення моделі та виділення найбільш значущих факторів, які впливають на ціни житла. Elastic Net Regression поєднує переваги L1- та L2-регуляризації, що робить його ефективним для роботи з даними, які мають як сильну кореляцію між ознаками, так і високу варіативність. Kernel Ridge Regression є потужним інструментом для моделювання складних нелінійних взаємозв'язків між змінними, що є типовим для задач прогнозування цін у нерухомості. Водночас Gradient Boosting Regression (XGBoost) забезпечує високу точність завдяки своїй здатності виправляти помилки попередніх моделей у процесі ансамблювання, що дозволяє враховувати складні залежності між характеристиками житлових об'єктів. Застосування цих моделей у сукупності дозволяє досягти високої точності прогнозування, забезпечуючи як інтерпретованість результатів, так і стійкість до перенавчання, що є ключовими факторами для створення ефективної інформаційної технології прогнозування цін на житлову нерухомість.

2.2 Розробка алгоритму інформаційної технології прогнозування цін на будинки

Для розв'язання даного завдання створено алгоритм інформаційної технології з опрацювання вхідних даних та налаштування й застосування моделі, яка містить такі кроки:

1. Аналіз вхідних даних;
2. Візуалізація даних за допомогою графіків щоб отримати краще розуміння залежності між ознаками та їх цінами;
3. Обробка вхідних даних та перевірка на наявність пропущених значень і їх заповнення відповідно до ознак, наприклад значеннями "Nan" або "0";
4. Розбиття даних на тренувальні й тестові, використовуючи метод перехресної перевірки;

5. Вибір базових моделей машинного навчання, які будуть використані для прогнозування цін на будинки;

6. Оптимізація моделей;

7. Оцінка точності моделей на тренувальних даних використовуючи метрику (RMSE);

8. Стекування моделей GBoost, KRR, ENet та використання Lasso як метамоделі для покращення точності;

9. Ансамблювання стекової моделі, LightGBM та XGBoost;

10. Формування прогнозів та графіків.

Граф-схема алгоритму інформаційної технології (рис. 2.1)

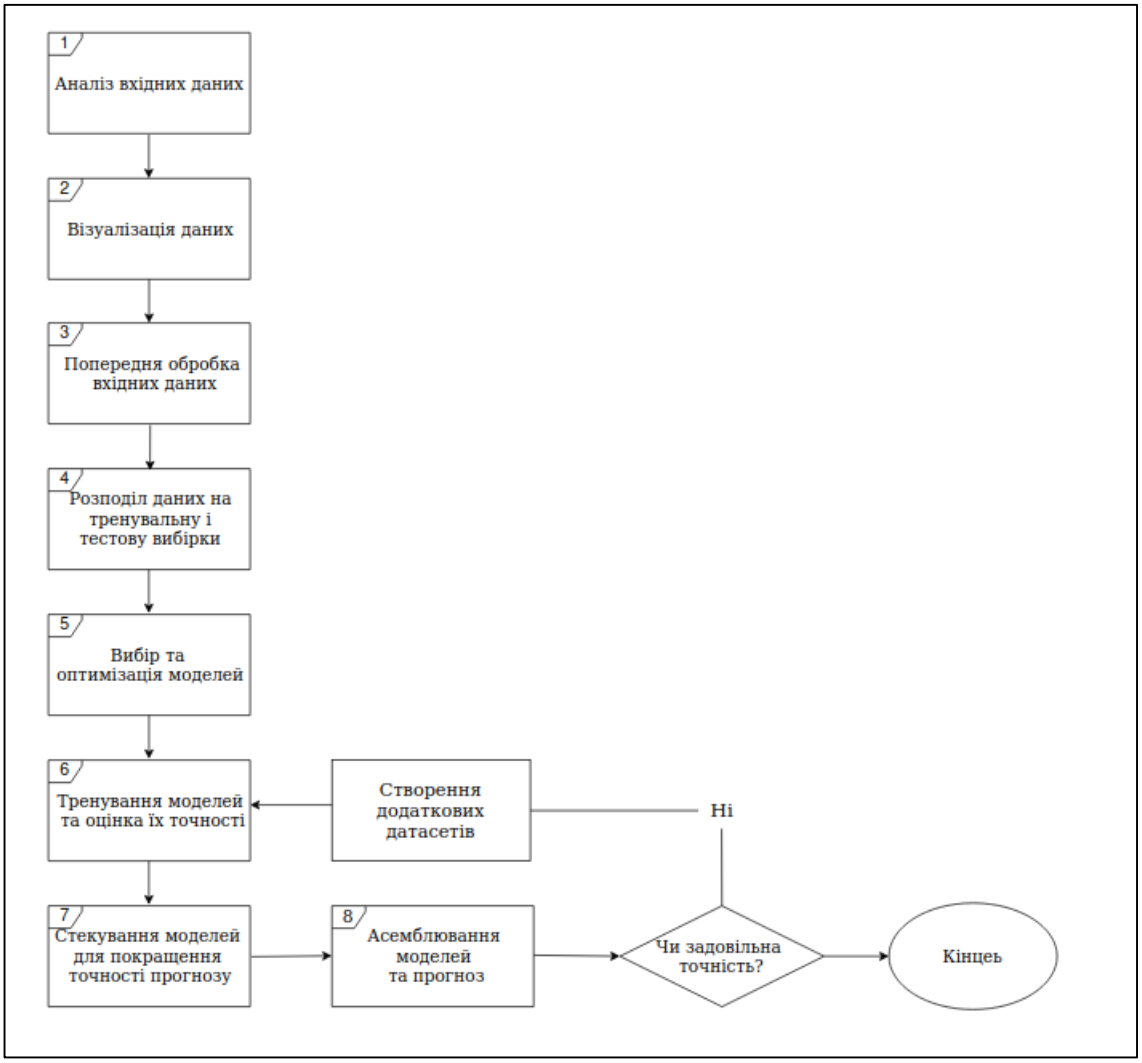

Рисунок 2.1 – Граф-схема алгоритму інформаційної технології прогнозування цін на приватні житлові будинки методами машинного навчання

2.3 Опис датасету

При попередньому огляді датасету було помічено, що багато даних мають пропущені значення. Проведемо розвідувальний аналіз даних. Дані взято із датасету «House Prices - Advanced Regression Techniques» [17].

Є 1460 екземплярів навчальних даних і 1460 тестових даних. Загальна кількість атрибутів дорівнює 81, з них 36 кількісних, 43 якісних + Id та змінна SalePrice – ціна продажу нерухомості в доларах. Наведемо декілька ознак:

– MSSubClass – клас будівлі;

– MSZoning – класифікація району;

– LotFrontage – лінійні метри вулиці, що з'єднана з об'єктом нерухомості;

– LotArea – розмір ділянки в квадратних футах;

– Street – вулиця, тип дорожнього доступу;

– Alley – тип алеї;

– LotShape – загальна форма ділянки;

– LandContour – рівність ділянки;

– Utilities – тип наявних інженерних комунікацій;

– LotConfig – конфігурація ділянки;

– LandSlope – нахил ділянки;

– Neighborhood – райони в межах міста Еймс;

– Condition1 – близькість до головної дороги або залізниці;

– Condition2 – близькість до головної дороги або залізниці (за наявності);

– BldgType – тип житла;

– HouseStyle – кількість поверхів;

– OverallQual – загальна якість матеріалів та оздоблення;

– OverallCond – загальна оцінка стану будинку;

– YearBuilt – рік побудови;

– YearRemodAdd – дата реконструкції;

– RoofStyle – тип даху;

– Exterior1st – зовнішнє покриття будинку;

– Exterior2nd – зовнішнє покриття на будинку (якщо більше одного матеріалу).

Приклад тренувального набору даних (train) показано на рисунку 2.3.

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	Po
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	Na
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	Na
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	Na
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	Na
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	Na
5	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	Na
6	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	Na
7	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	Na
8	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	Na
9	190	RL	50.0	7420	Pave	NaN	Reg	Lvl	AllPub	Corner	...	0	Na

Рисунок 2.2 – Тренувальний набір даних

Для початку розраховується кількість пропущених значень для тестового набору даних (рис. 2.3).

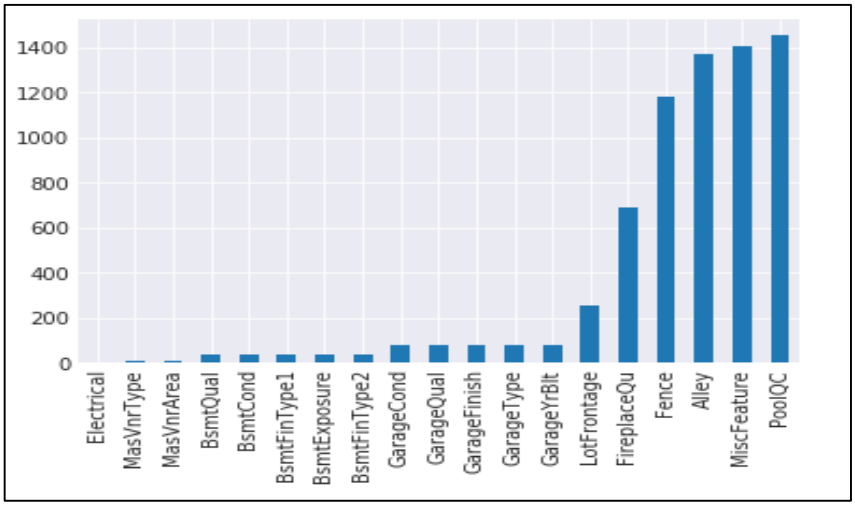

Рисунок 2.3 – Графік пропущених значень

19 атрибутів мають пропущені значення, 5 з яких становлять понад 50% усіх даних. У більшості випадків “NAN” означає відсутність об'єкта, описаного атрибутом, наприклад, відсутній басейн, паркан, немає гаража та підвалу. Візуалізація розподілу Джонсона зображена на рисунку 2.4.

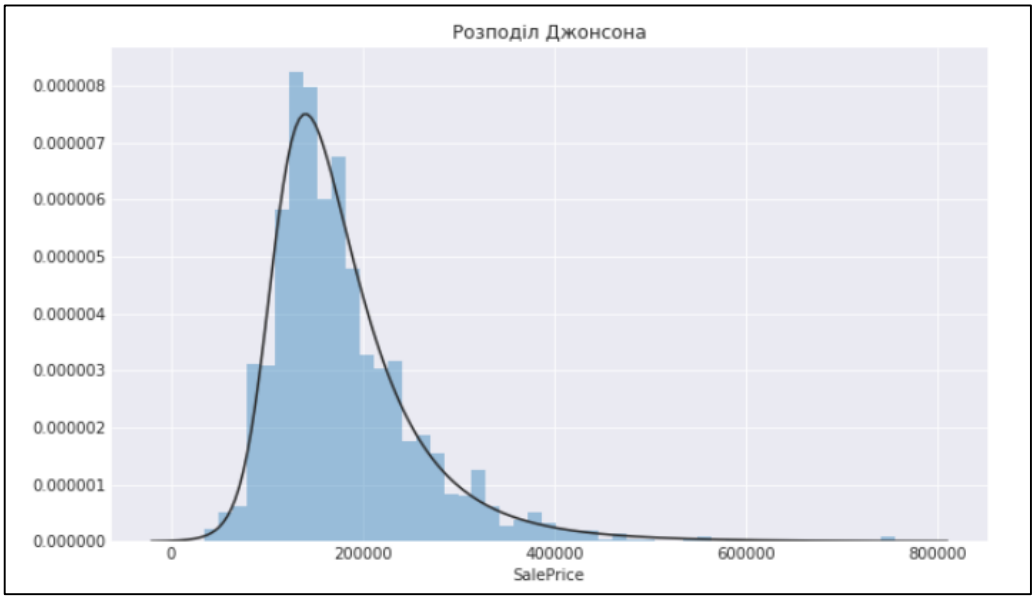

Рисунок 2.4 – Візуалізація за розподілом Джонсона

Візуалізація логарифмічного розподілу (рис. 2.5).

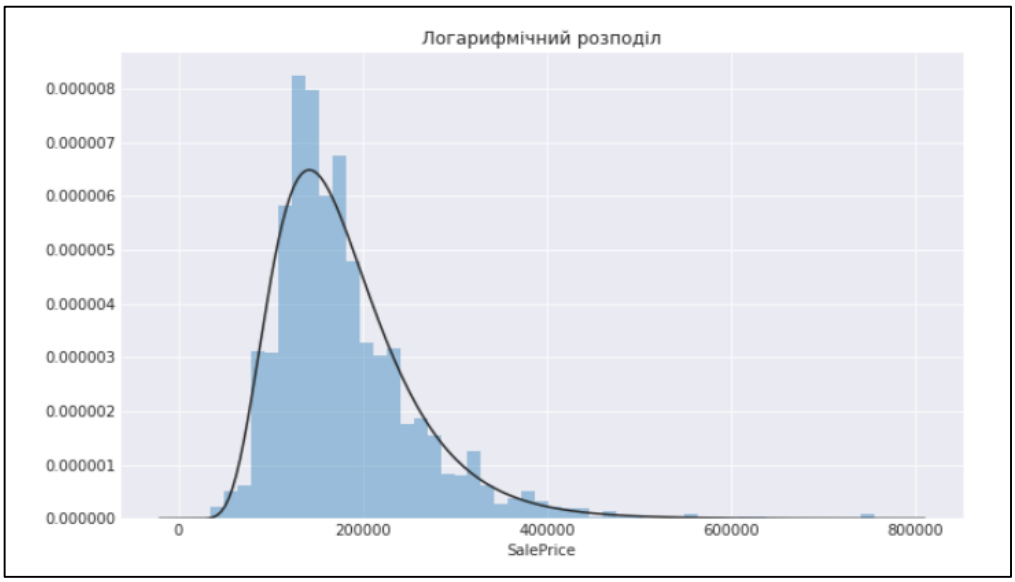

Рисунок 2.5 – Логарифмічний розподіл

Очевидно, що SalePrice не підпорядковується нормальному розподілу, тому перед виконанням регресії її потрібно трансформувати. Хоча логарифмічне перетворення дає досить хороші результати, найкраще підходить розподіл Джонсона.

Також перевіряються кількісні змінні на нормальний розподіл (рис. 2.6).

```
test_normality = lambda x: stats.shapiro(x.fillna(0))[1] < 0.01
normal = pd.DataFrame(train[quantitative])
normal = normal.apply(test_normality)
print(not normal.any())
```

```
False
```

Рисунок 2.6 – Перевірка на нормальний розподіл

Результат – “False” отже кількісні змінні не мають нормального розподілу, тому їх потрібно трансформувати.

Проведемо аналіз графіків кількісних атрибутів.

На першому графіку змінна MSSubClass (Клас житла за типом забудови). Значення скошені вправо, декілька чітких піків відповідають різним типам забудови. Отже, краще залишити змінну в категоріальному форматі або розглянути її як окремі класи (one-hot encoding), адже вона, ймовірно, відповідає різним типам житла.

На другому графіку змінна LotFrontage (Фасад земельної ділянки). Розподіл помітно скошений вправо. Більшість значень сконцентровані в межах 50-100. Присутні викиди (значення понад 150). Можливо, буде корисним трансформувати цю змінну (наприклад, логарифмування) для зменшення впливу викидів.

Графіки розподілу кількісних атрибутів MSSubClass, LotFrontage, LotArea, OverallQual показано на рисунку 2.7.

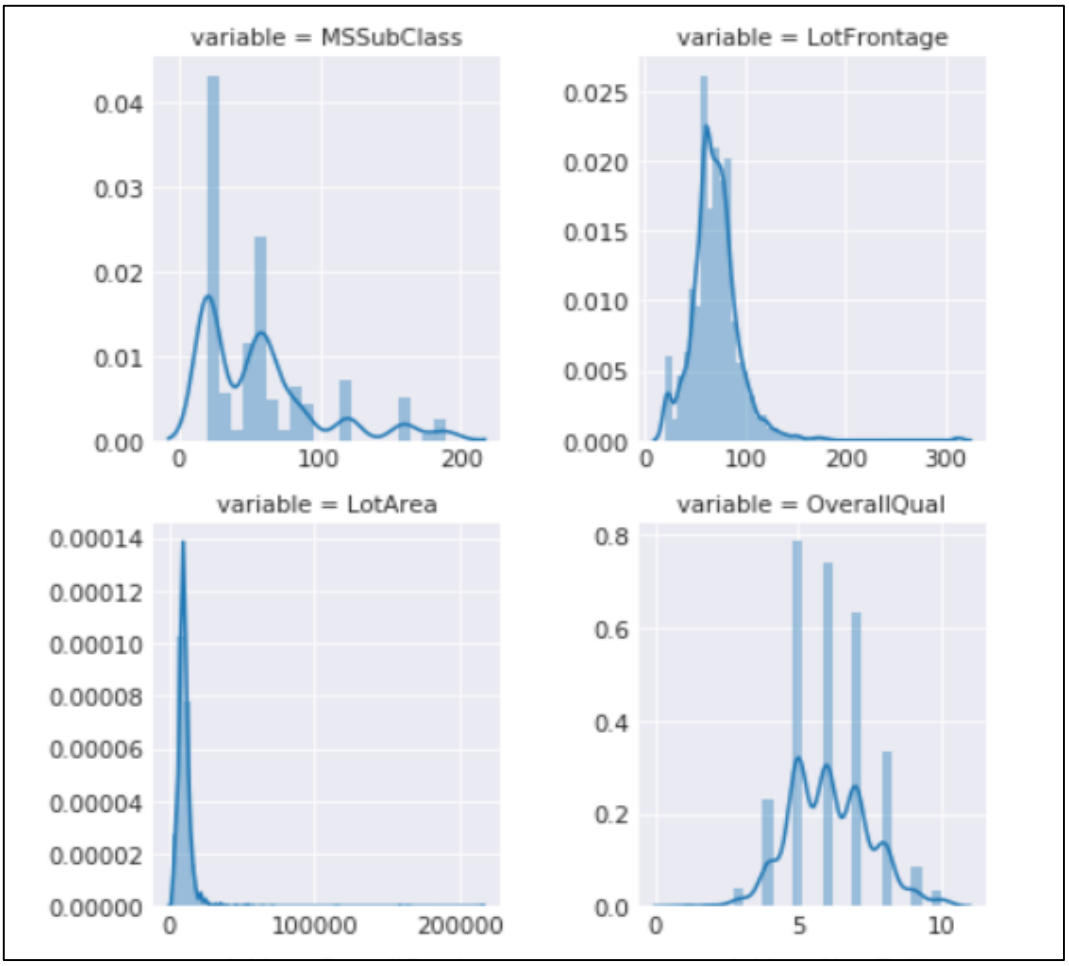

Рисунок 2.7 – Розподіл кількісних атрибутів OverallCond, YearBuilt, YearRemodAdd, MasVnrArea

На третьому графіку змінна LotArea (Площа земельної ділянки). Розподіл має сильний правий нахил, з великою кількістю значень на нижньому кінці. Існують значні викиди (великі площі понад 50,000 футів). Трансформація (логарифмування) може допомогти нормалізувати розподіл.

На четвертому гарфіку змінна OverallQual (Загальна якість будинку). Розподіл даних близький до нормального, з кількома піками. Основна маса значень знаходиться в діапазоні від 5 до 8. Отже, підходить для моделювання без трансформацій. Якість є важливою ознакою і має чітку кореляцію з цінами.

Графіки розподілу кількісних атрибутів OverallCond, YearBuilt, YearRemodAdd, MasVnrArea. показано на рисунку 2.8.

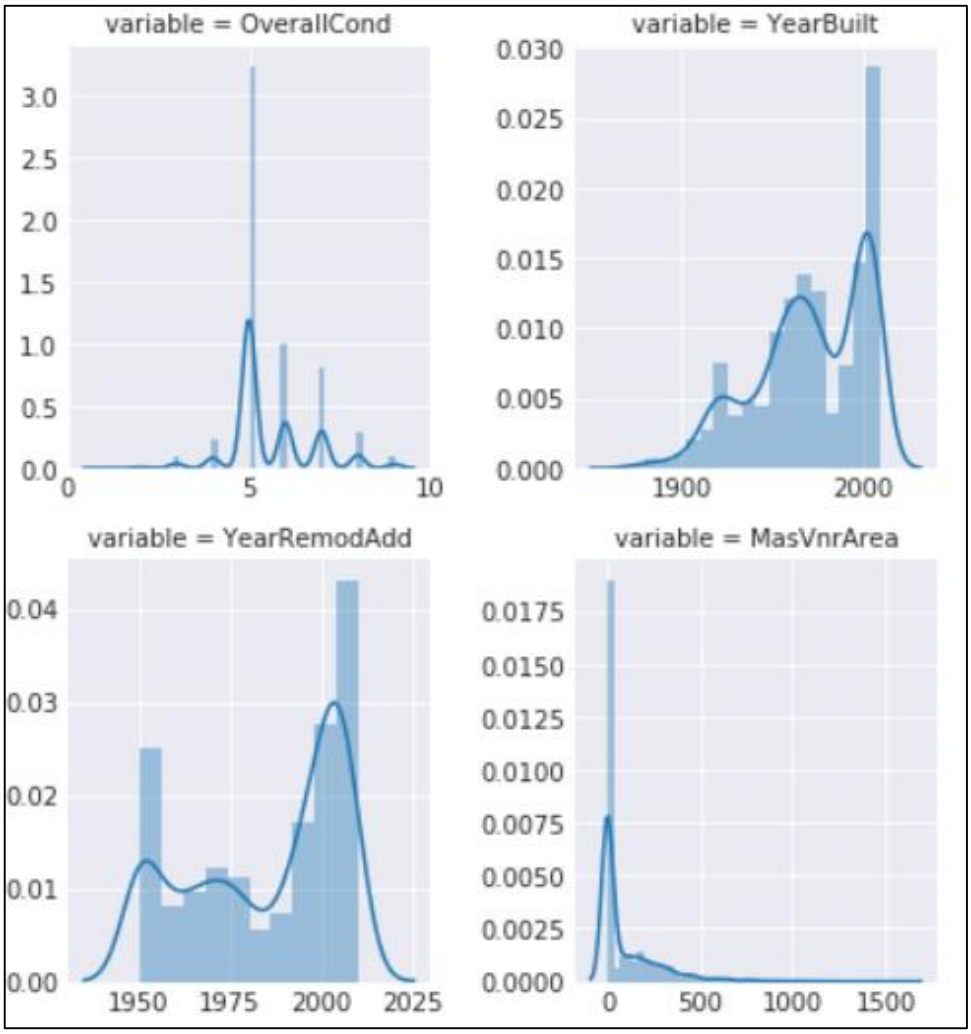

Рисунок 2.8 – Розподіл кількісних атрибутів OverallCond, YearBuilt, YearRemodAdd, MasVnrArea

На першому графіку змінна OverallCond (Загальний стан будинку) Більшість значень знаходяться в межах 5–6, що означає середній стан будинків. Є піки для стану "5" (найпоширеніший клас), а також для "3" і "6". Дуже мало даних для станів нижче "3" і вище "8".

На другому графіку змінна YearBuilt (Рік побудови). Спостерігається зростання кількості будинків, побудованих після 1950-х років, із піком у 2000-х. Старі будинки (до 1900 року) зустрічаються рідко, що створює довгий "хвіст" розподілу. Дані сильно зміщені вправо (основна маса знаходиться ближче до сучасних років).

На третьому графіку змінна YearRemodAdd (Рік останнього ремонту/модифікації). Розподіл даних подібний до YearBuilt, але з сильним акцентом на останні роки. З графіку видно, що найбільше модифікацій проводилось у 2000–2010 роках. Також спостерігається відносно рівномірний розподіл до 1950-х.

На четвертому графіку змінна MasVnrArea (Площа облицювання будинку, кв. футів). Дуже багато нульових значень (відсутність облицювання), із деякими значеннями, що сягають до 1500 кв.футів. Розподіл дуже скошений: більшість значень ближче до 0, є кілька великих аномальних значень. Ця змінна потребує логарифмічного перетворення для коригування її впливу на модель.

Проведемо аналіз наступних графіків.

На першому графіку зображений розподіл змінної BsmtFinSF1 (Площа першої завершеної зони підвалу). По графіку видно, що розподіл сильно зміщений вліво. Помітна значна кількість нульових значень, що означає відсутність завершеного підвалу у багатьох будинках. Також є невелика кількість будинків із дуже великою площею завершеного підвалу (понад 4000 кв. футів), які можуть бути аномаліями.

На другому графіку зображений розподіл змінної BsmtFinSF2 (Площа другої завершеної зони підвалу). Розподіл сильно скошений; більшість значень дорівнює нулю. У переважної більшості будинків немає другої завершеної зони підвалу. Деякі будинки мають невеликі площі цієї зони, і лише кілька — більше 1000 кв. футів. Отже, ця змінна може бути менш значущою для моделі через велику кількість нульів. Також додатково варто перевірити її кореляцію із SalePrice.

Графіки розподілу кількісних атрибутів BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF показано на рисунку 2.9.

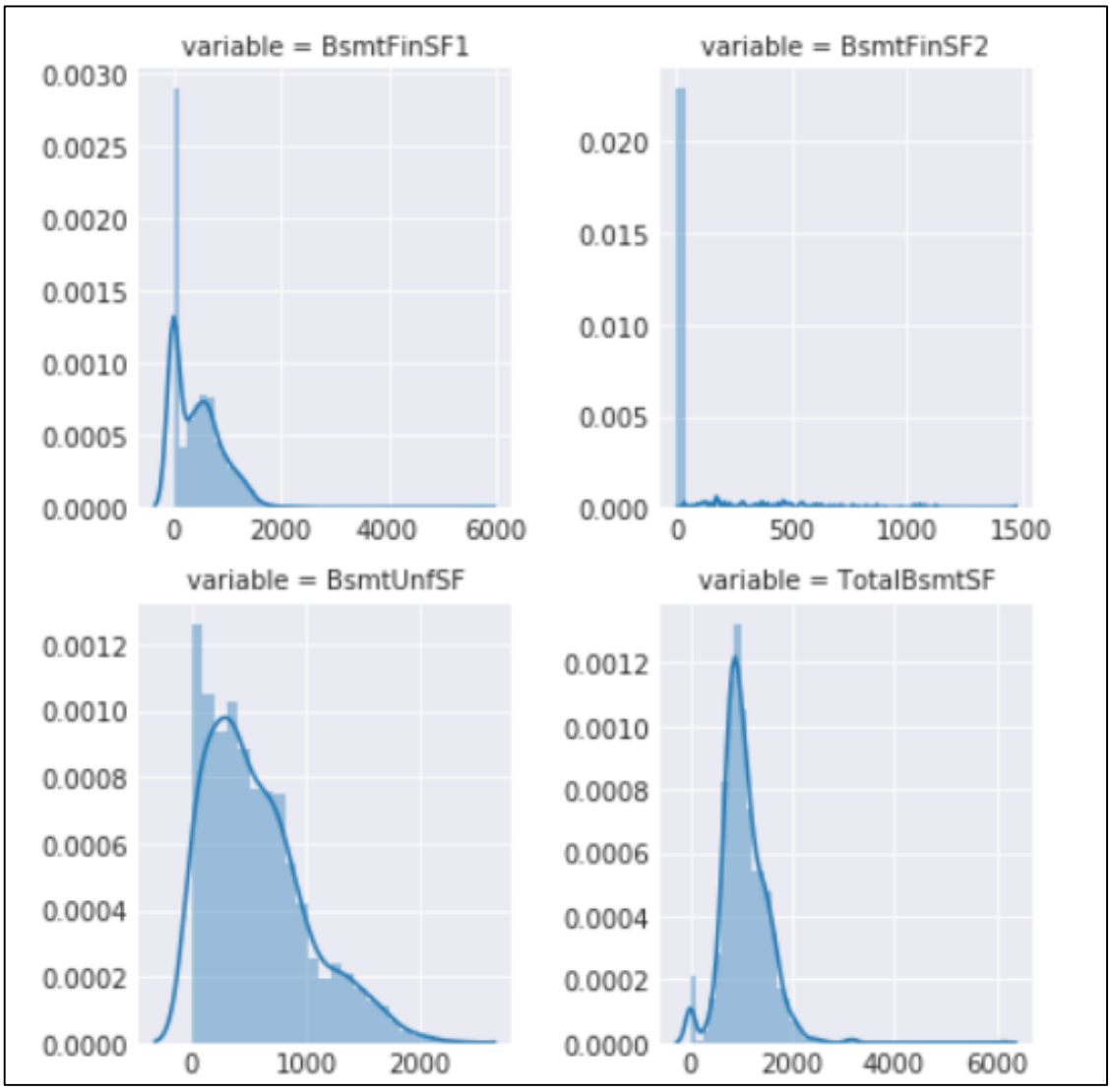

Рисунок 2.9 – Розподіл кількісних атрибутів BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF

На третьому графіку зображений розподіл змінної BsmtUnfSF (Площа незавершеної частини підвалу). Розподіл скошений вправо, але стає рівномірним у середньому діапазоні значень. Є невелика кількість нульових значень, що означає відсутність незавершених частин підвалу. Значення варіюються від 0 до понад 2000 кв. футів.

На четвертому графіку зображений розподіл змінної TotalBsmtSF (Загальна площа підвалу). Розподіл схожий на BsmtFinSF1, але більш концентрований у діапазоні до 2000 кв. футів. Висока кореляція із

завершеними і незавершеними площами підвалу. Наявність аномальних значень із площею понад 4000 кв. футів.

Також важливо перевірити категоріальні дані. Для якісних змінних можна застосувати два методи. Перший – перевірити розподіл SalePrice відносно значень змінної та перерахувати їх. Другий – створити фіктивну змінну для кожної можливої категорії.

Проведемо аналіз графіків категоріальних ознак.

На першому графіку змінна MSZoning, яка описує зону землекористування.

- RL (Residential Low Density) — житлова зона низької щільності.
- RM (Residential Medium Density) — житлова зона середньої щільності.
- C (all) (Commercial) — комерційна зона.
- FV (Floating Village Residential) — житлова зона на воді (елітна).
- RH (Residential High Density) — житлова зона високої щільності.

Найвищі ціни спостерігаються для зони FV, тоді як C та RH мають нижчі медіанні ціни. Розподіл у зоні RL досить широкий, вказуючи на різні цінові сегменти.

Street. Тут розглядаються типи доріг: Pave (заасфальтовані) та Grvl (гравійні). Медіанні ціни будинків на асфальтованих дорогах значно вищі в той час, як Grvl демонструє менший розмах цін, що може вказувати на менш різнорідні властивості будинків.

Змінна Alley містить категорії MISSING (відсутні дані), Grvl(гравійні), та Pave(заасфальтовані). MISSING може вказувати на те, що більшість будинків не мають алеї (цілком ймовірно для багатьох житлових зон). Їх ціни знаходяться у середньому діапазоні. Pave має тенденцію до вищих цін у порівнянні з Grvl.

Графіки залежності категоріальних змінних MSZoning, Street, Alley, LotShape відносно SalePrice (рис. 2.10).

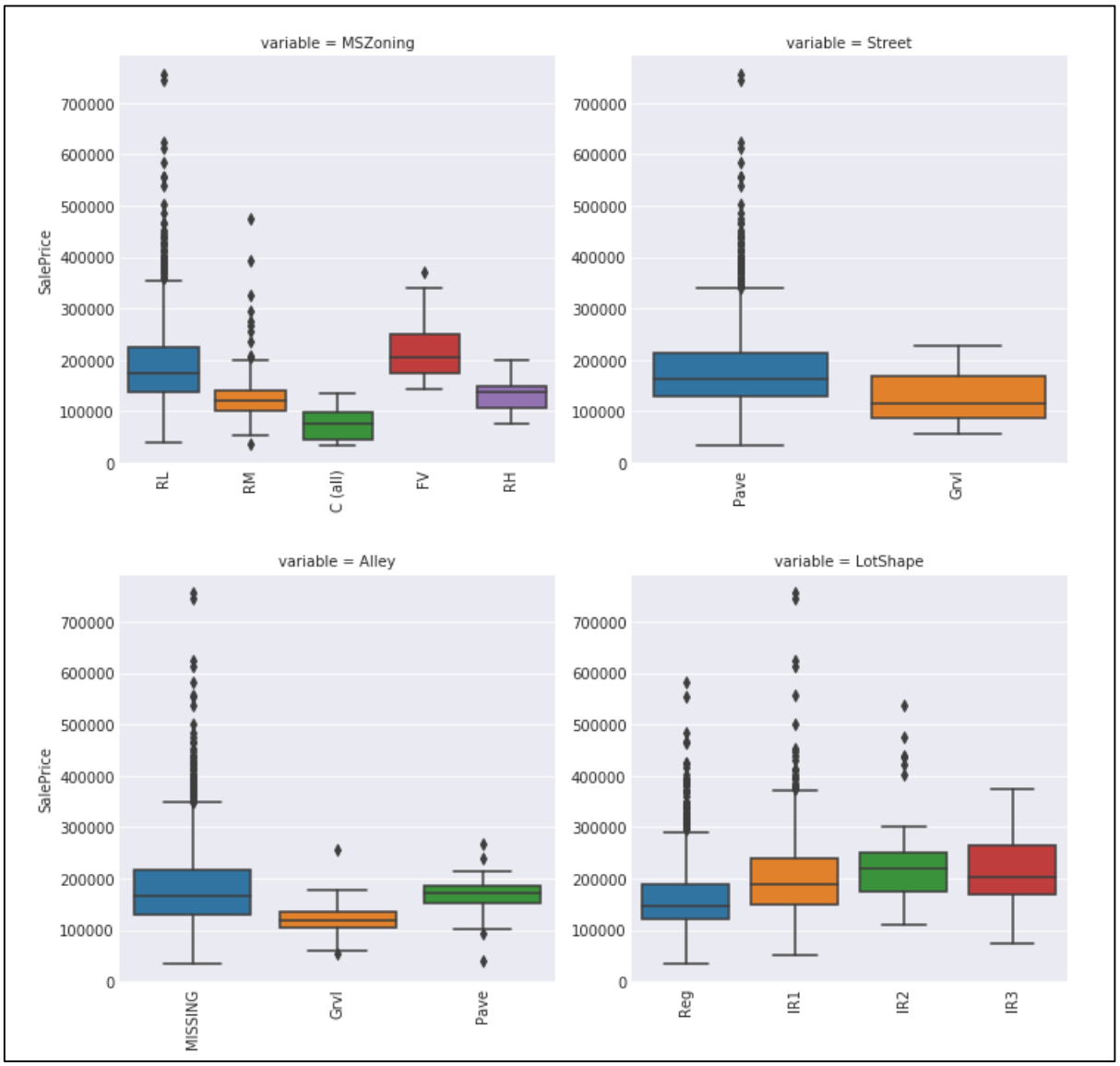

Рисунок 2.10 – Графіки залежності категоріальних змінних

LotShape (Форма земельної ділянки).

Reg (Regular) — Регулярна (правильна) форма.

IR1 (Slightly Irregular) — Трохи нерегулярна форма.

IR2 (Moderately Irregular) — Помірно нерегулярна форма.

IR3 (Irregular) — Нерегулярна форма.

Найвищі ціни спостерігаються для IR3, можливо, через ексклюзивність або більші площі.

Загальні спостереження:

Аномалії (викиди). У кожному графіку є помітні викиди — будинки з ціною понад $500,000. Це може бути елітна нерухомість, яку варто окремо проаналізувати чи фільтрувати під час моделювання.

Вплив відсутніх даних. У змінній Alley, категорія "MISSING" займає значну частину вибірки. Врахування цього може бути корисним у моделі, адже відсутність алеї також може бути характеристикою нерухомості.

Залежність від категорій. Деякі категорії (наприклад, FV у MSZoning або Pave у Street) явно пов’язані з вищими цінами, що свідчить про їх значущість для моделі.

Проведемо аналіз наступних графіків:

На першому графіку зображена залежність ціни SalePrice від GarageCond (Стан гаража).

- TA (Typical/Average) — Типовий/середній стан.
- Fa (Fair) — Задовільний стан.
- Gd (Good) — Хороший стан.
- Po (Poor) — Поганий стан.
- Ex (Excellent) — Відмінний стан.
- MISSING — Відсутність гаража.

Найбільший розмах цін спостерігається для TA, що, ймовірно, є найпоширенішою категорією. Гаражі в стані Ex пов’язані з вищими цінами, проте їх кількість у вибірці мала.

Найбільший розмах цін спостерігається для TA, що, ймовірно, є найпоширенішою категорією. Гаражі в стані Ex пов’язані з вищими цінами, проте їх кількість у вибірці мала.

Графіки залежності категоріальних змінних GarageCond, PavedDrive, PoolQC, Fence відносно SalePrice (рис. 2.11).

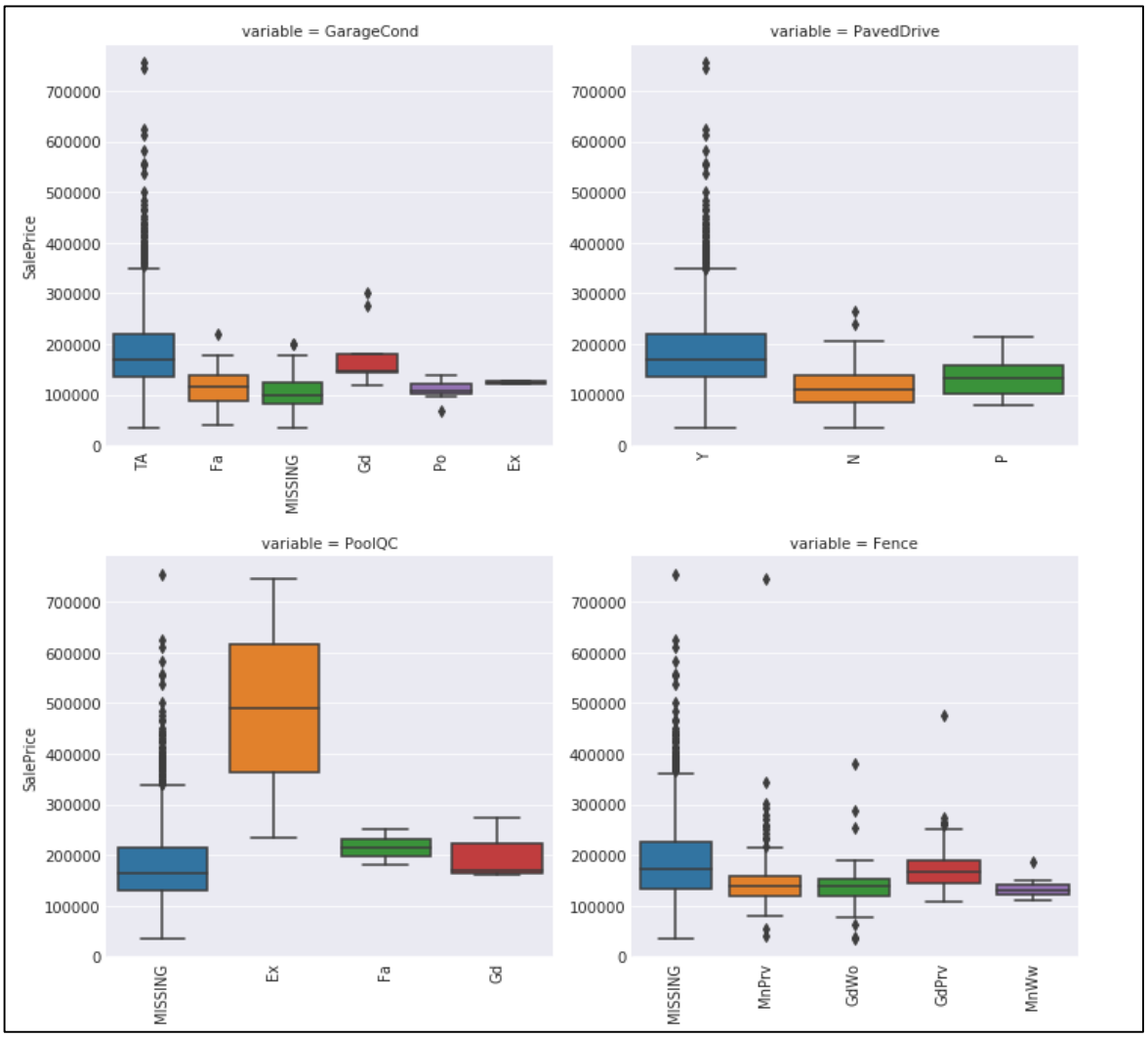

Рисунок 2.11 – Графіки залежності категоріальних змінних

На другому графіку зображена залежність ціни SalePrice від PavedDrive (Асфальтована дорога до будинку):

- Y (Yes) — Є асфальтована дорога.
- N (No) — Відсутня асфальтована дорога.
- P (Partial) — Частково асфальтована дорога.

Будинки з повністю асфальтованими дорогами (Y) мають найвищі медіанні ціни. Відсутність асфальтованої дороги (N) пов’язана зі значно нижчими цінами, тоді як P має розподіл, схожий на N.

На третьому графіку зображена залежність ціни SalePrice від PoolQC (Якість басейну):

- Ex (Excellent) — Відмінна якість.

- Gd (Good) — Хороша якість.
- Fa (Fair) — Задовільна якість.
- MISSING — Відсутність басейну.

Найвищі ціни пов’язані з басейнами Ex, що свідчить про ексклюзивність будинків з якісними басейнами. Значення MISSING займає найбільшу частину вибірки, що підтверджує, що басейни є рідкісною характеристикою.

На четвертому графіку зображена залежність ціни SalePrice від Fence (Якість паркану):

- MnPrv (Minimum Privacy) — Мінімальна конфіденційність.
- GdPrv (Good Privacy) — Хороша конфіденційність.
- GdWo (Good Wood) — Хороша деревина.
- MnWw (Minimum Wood/Wire) — Мінімальна деревина/метал.
- MISSING — Відсутність паркану.

Відсутність паркану (MISSING) не впливає негативно на ціни, що вказує, що паркан не є критичною характеристикою. Категорія GdPrv пов’язана з вищими цінами, ймовірно, через додаткову приватність.

Загальні спостереження:

Аномалії (викиди). У всіх графіках видно будинки з високими цінами (> $500,000), що є винятками та вказують на преміум-сегмент.

Вплив відсутніх даних. PoolQC та Fence мають значну кількість значень "MISSING". Це вказує на те, що басейни та паркани є нетиповими для більшості будинків, і це слід враховувати під час моделювання.

Значущість змінних. GarageCond, PavedDrive, та PoolQC мають чітку залежність категорій від ціни, що робить їх значущими для моделі.

Нижче наведено швидку оцінку впливу категоріальної змінної на SalePrice. Для кожної змінної SalePrice розбиваються на окремі множини на основі значень категорій. Потім перевіряються за допомогою ANOVA тесту, чи мають набори схожий розподіл. Якщо змінна має незначний вплив, то середні значення наборів повинні бути однаковими. Зменшення «pval» є ознакою збільшення різноманітності в розбиттях (рис. 2.12 – 2.13).

```
def anova(frame):
    anv = pd.DataFrame()
    anv['feature'] = qualitative
    pvals = []
    for c in qualitative:
        samples = []
        for cls in frame[c].unique():
            s = frame[frame[c] == cls]['SalePrice'].values
            samples.append(s)
        pval = stats.f_oneway(*samples)[1]
        pvals.append(pval)
    anv['pval'] = pvals
    return anv.sort_values('pval')

a = anova(train)
a['disparity'] = np.log(1./a['pval'].values)
sns.barplot(data=a, x='feature', y='disparity')
x=plt.xticks(rotation=90)
```

Рисунок 2.12 – Аналіз дисперсії для якісних змінних

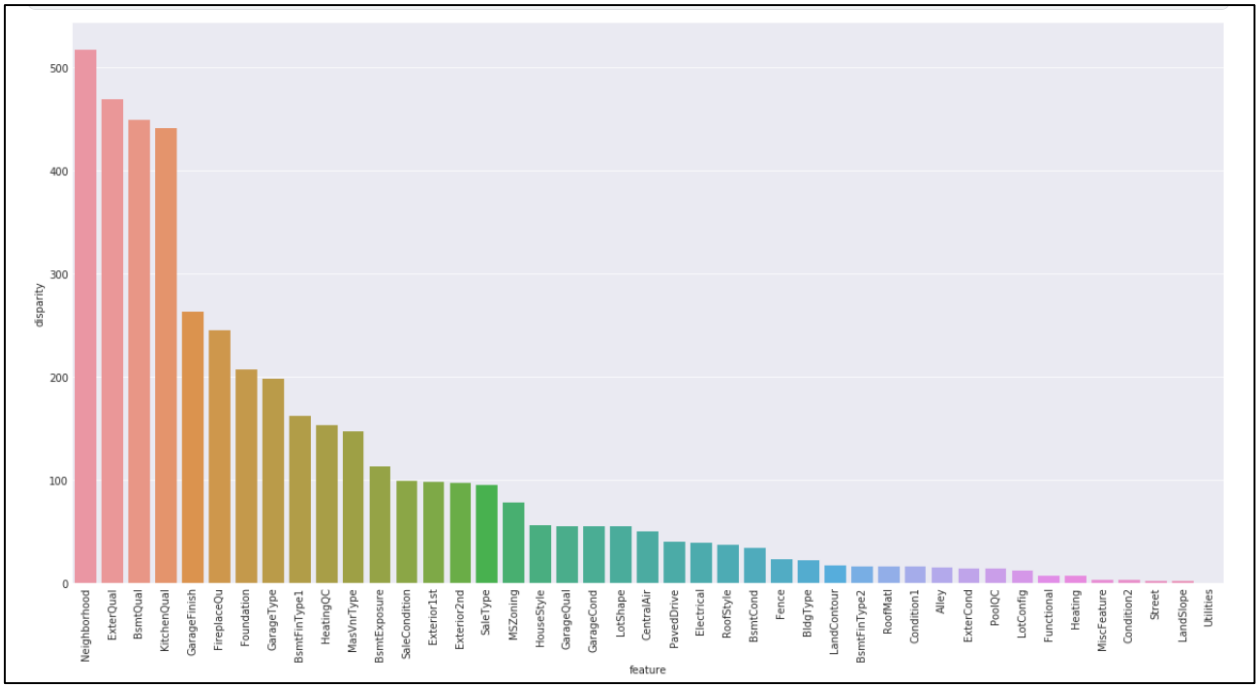

Рисунок 2.13 – Результат аналізу

Кодування категоріальних ознак в числові значення зображено на рисунку 2.14.

```
def encode(frame, feature):
    ordering = pd.DataFrame()
    ordering['val'] = frame[feature].unique()
    ordering.index = ordering.val
    ordering['spmean'] = frame[[feature, 'SalePrice']].groupby(feature).mean()['SalePrice']
    ordering = ordering.sort_values('spmean')
    ordering['ordering'] = range(1, ordering.shape[0]+1)
    ordering = ordering['ordering'].to_dict()

    for cat, o in ordering.items():
        frame.loc[frame[feature] == cat, feature+'_E'] = o
qual_encoded = []
for q in qualitative:
    encode(train, q)
    qual_encoded.append(q+'_E')
print(qual_encoded)
```

Рисунок 2.14 – Кодування категоріальних ознак

Тепер якісні змінні кодуються відповідно до впорядкування на основі середнього значення SalePrice.

Кореляція Спірмана краще підходить для цього випадку, оскільки вона вловлює зв’язки між змінними, навіть якщо вони нелінійні. OverallQual є основним критерієм у визначенні ціни на житло. Сусідство має великий вплив, частково воно має певну внутрішню цінність саме по собі, але також будинки в певних регіонах мають тенденцію мати однакові характеристики (змішування), що призводить до подібних оцінок.

Загалом, щоб зменшити плутанину, до регресійних моделей слід додавати лише некорельовані між собою змінні (які корелюють з SalePrice) (рис. 2.15).

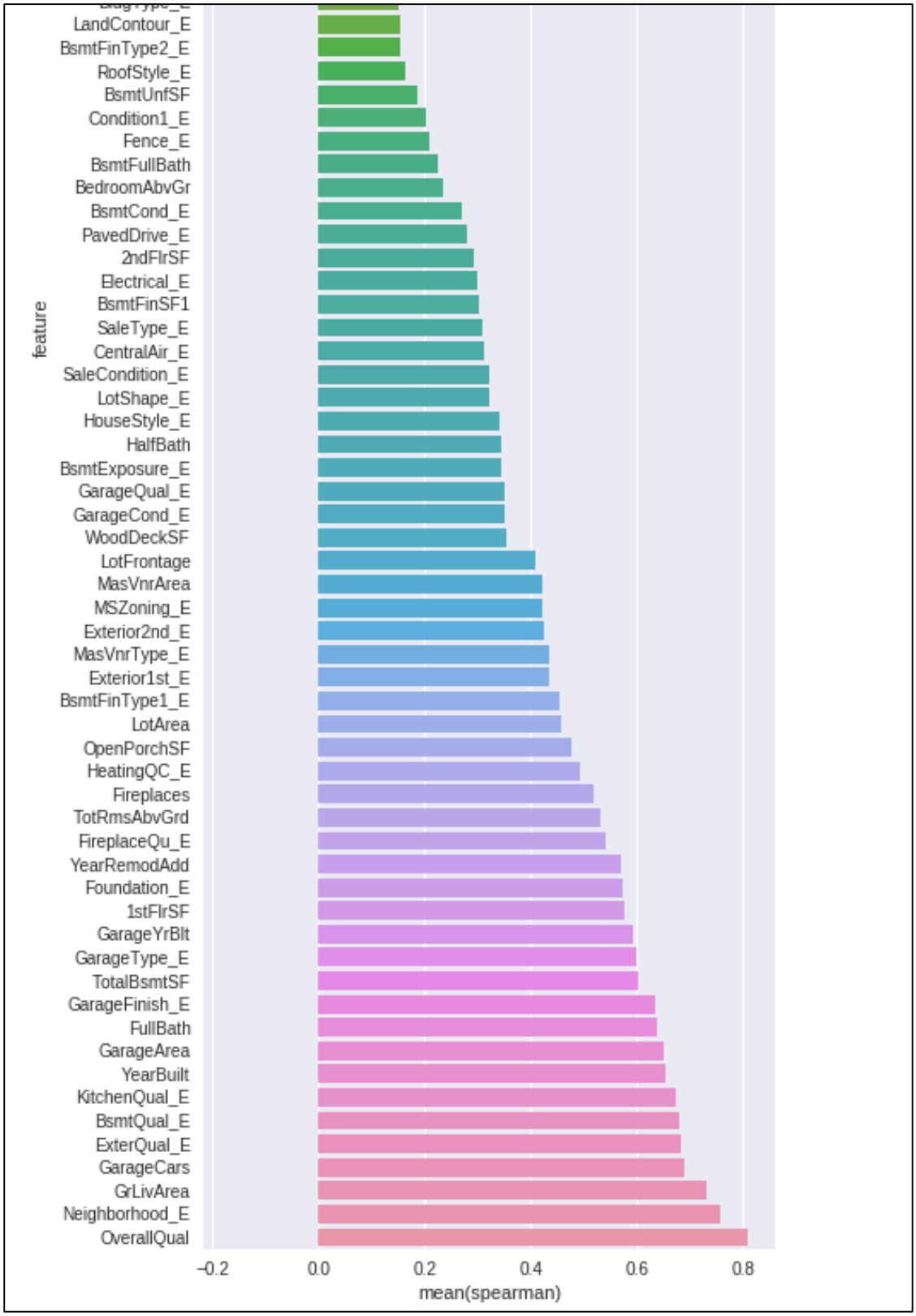

Рисунок 2.15 – Кореляція Спірмана

Кореляційна матриця для дослідження зв’язків між числовими та закодованими категоріальними ознаками (рис. 2.16).

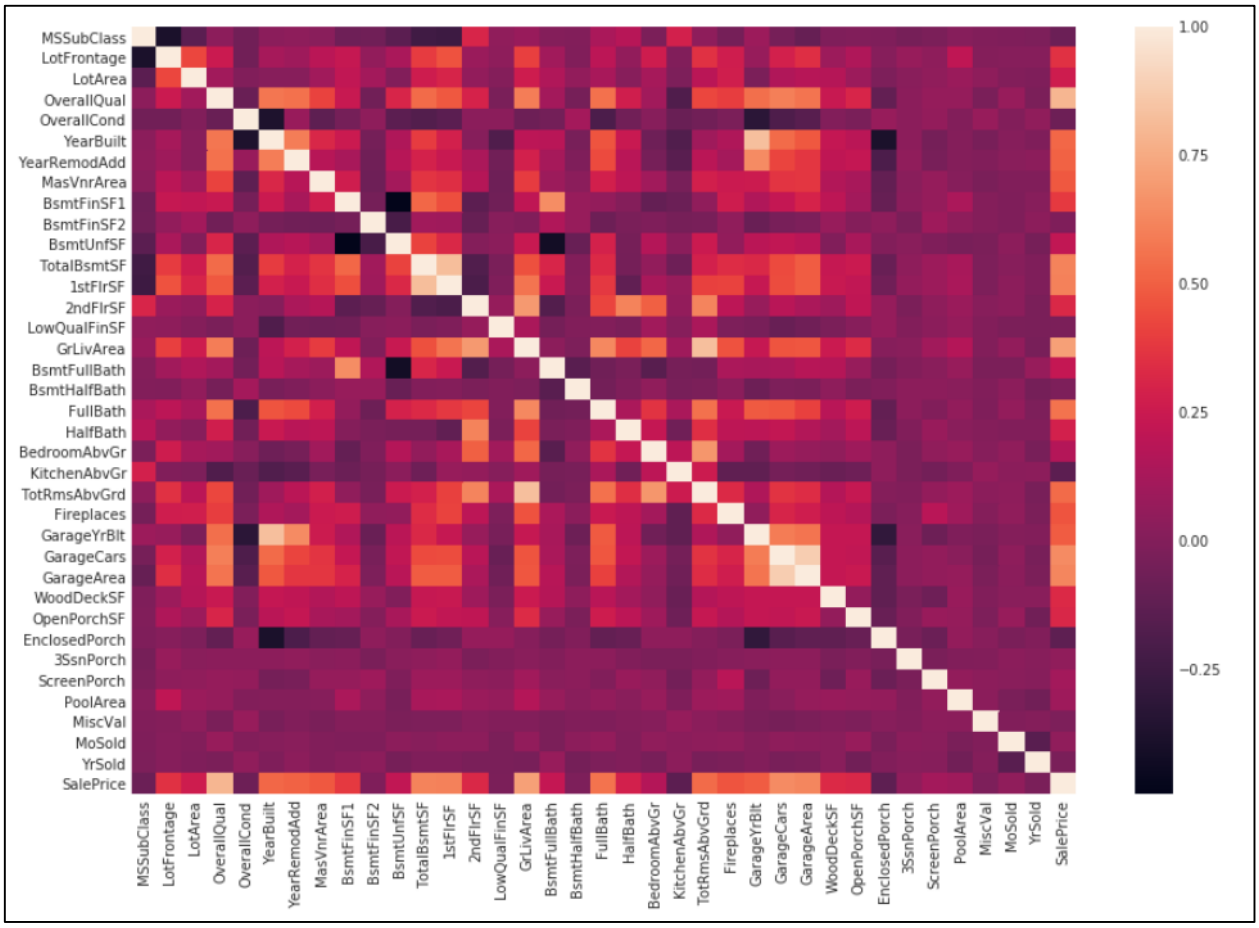

Рисунок 2.16 – Кореляційна матриця

Існує багато сильних кореляцій між змінними. Гаражі, схоже, побудовані в тому ж році, що і будинки, підвали мають, як правило, таку ж площу, що і перші поверхи, що є досить очевидним. Площа гаража сильно корелює з кількістю автомобілів. Сусідство корелює з багатьма іншими змінними, і це підтверджує ідею, що будинки в одному регіоні мають однакові характеристики. Тип житла негативно корелює з площею кухні над рівнем підлоги.

Також було б корисно побачити, як ціна продажу співвідноситься з кожною незалежною змінною.

Співвідношення ознак MSSubClass, LotFrontage, LotArea, OverallQual до SalePrice (рис. 2.17).

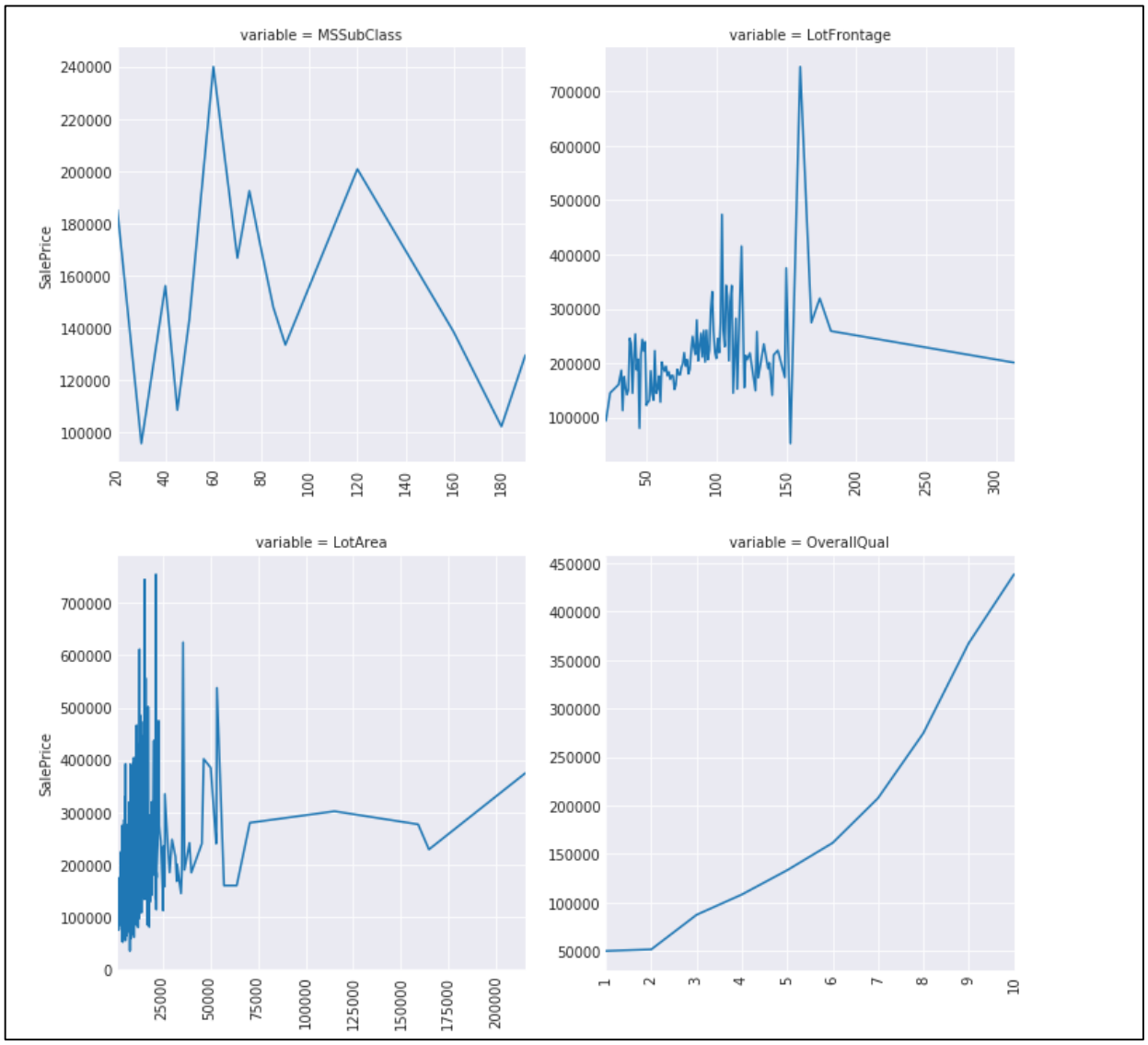

Рисунок 2.17 – Співвідношення ознак

Проведемо аналіз даних графіків.

MSSubClass (Тип житлового будинку) — змінна відображає клас забудови, наприклад:

20 — Одноповерховий будинок 1946 року чи пізніше.

60 — Дворівневий будинок 1946 року чи пізніше.

120 — Одноповерховий будинок з добудовами після 1946 року.

Проаналізувавши перший графік можна помітити, що ціни сильно варіюються між класами. Категорія 20 демонструє середні ціни, а категорія 120 має вищу медіану. Велика кількість піків та спадів може свідчити про вплив додаткових факторів, як рік побудови чи якість.

LotFrontage (Фасад земельної ділянки, у футах) - змінна представляє довжину вуличного фасаду ділянки.

На другому графіку спостерігається нерівномірний розподіл цін. Від 50 до 150 футів ціна сильно варіюється, що може свідчити про вплив інших факторів (якість будинку, район тощо). Після 150 футів ціна знижується, але це, ймовірно, через рідкість таких об'єктів у вибірці.

LotArea (Загальна площа ділянки, у квадратних футах) — змінна описує загальну площу земельної ділянки.

Третій графік показує зростання ціни зі збільшенням площі, але після 50,000 кв. футів ця залежність стає слабшою. Високі ціни для невеликих ділянок можуть свідчити про вплив інших факторів (наприклад, розташування в елітному районі).

OverallQual (Загальна якість будинку) — змінна оцінює загальну якість матеріалів і оздоблення, від 1 (низька якість) до 10 (висока якість).

На четвертому графіку помітна сильна позитивна кореляція: чим вища якість, тим вища ціна. Починаючи з оцінки 6, ціни суттєво зростають, а при 10 спостерігаються найвищі ціни.

Загальні спостереження:

Залежність від площі та фасаду. Хоча більші ділянки часто коштують дорожче, ціна також залежить від інших характеристик, як якість будинку (OverallQual).

Нерівномірність вибірки. Для деяких змінних (наприклад, LotFrontage > 150) кількість спостережень мала, що може впливати на моделі.

Сильні кореляції. OverallQual має найчіткіший зв’язок із цінами, що робить цю змінну важливою для побудови моделі.

Нижче показано співвідношення ознак OverallCond, YearBuilt, YearRemodAdd, MasVnrArea до SalePrice (рис. 2.18).

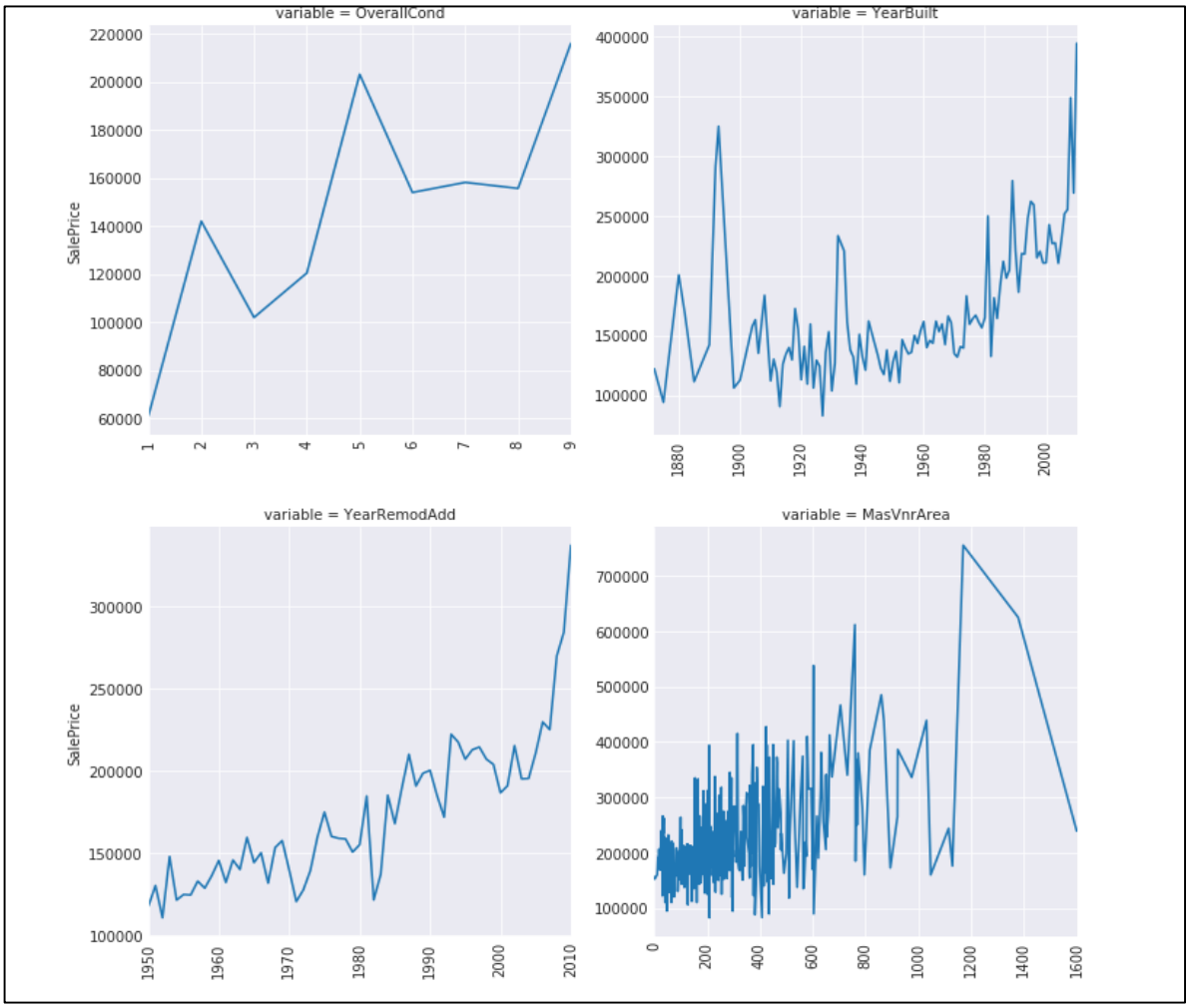

Рисунок 2.18 – Співвідношення ознак OverallCond, YearBuilt, YearRemodAdd, MasVnrArea до SalePrice

На першому графіку зображено співвідношення OverallCond (Загальний стан будинку) до SalePrice.

Ця змінна оцінює загальний стан будинку, від 1 (дуже поганий) до 9 (відмінний). Ціна загалом зростає зі збільшенням оцінки стану. Найпомітніше зростання спостерігається між оцінками 3 і 7.Для оцінок 8 і 9 ціни стають найвищими, що вказує на значення гарного стану для покупців.

На другому графіку зображено співвідношення YearBuilt (Рік побудови) до SalePrice. Після 1950 року чітко помітний тренд зростання цін, особливо для будинків, побудованих після 2000 року. Старі будинки (до 1900 року) мають як низькі, так і високі ціни, можливо, через історичну цінність.

На третьому графіку зображено співвідношення YearRemodAdd (Рік останньої реконструкції) до SalePrice. Ця змінна вказує на рік, коли будинок востаннє ремонтувався чи реконструювався. По графіку видно, що ціни мають сильну залежність від року останньої реконструкції: чим ближче до сучасності, тим вища ціна. Після 2000 року спостерігається стрімке зростання цін, можливо, через використання сучасних матеріалів чи дизайну.

На четвертому графіку зображено співвідношення MasVnrArea до SalePrice. Ця змінна описує площу зовнішнього облицювання з каменю чи цегли. Відсутність облицювання (значення близько до 0) асоціюється з широким розкидом цін. Зі збільшенням площі облицювання ціни спершу зростають, досягаючи піку на рівні близько 1,200 кв. футів. Після цього ціни зменшуються, ймовірно, через рідкість об'єктів із дуже великою площею облицювання.

Загальні спостереження:

1. Часові тренди. Рік побудови та реконструкції чітко вказують на перевагу нових і оновлених будинків.

2. Облицювання як фактор. Наявність зовнішнього облицювання значно впливає на ціну, але важливий не лише розмір, а й якість.

3. Загальний стан. OverallCond впливає на ціну, але менш виражено, ніж рік реконструкції чи облицювання.

4. Рідкісні категорії. Старі будинки або великі площі облицювання мають нестандартний вплив на ціни, що може свідчити про аномалії чи ексклюзивні об'єкти.

2.4 Попередня обробка вхідних даних

Обробка вхідних даних в завданнях прогнозування виконується, перш за все, щоб підготувати дані для подальшої роботи, наприклад – моделювання. Вхідні дані можуть містити шум або непотрібні відхилення, які можуть негативно вплинути на якість прогнозів моделі. Також дані можуть містити

багато різних характеристик або ознак, і не всі з них можуть бути корисними для прогнозування. Обробка даних може включати вибір та відбір ознак, що мають найбільшу кореляцію або значимість для цільової змінної, що дозволяє зменшити розмірність даних та поліпшити продуктивність моделі.

Отже, обробка вхідних даних є важливим етапом для досягнення хорошого прогнозу.

Спочатку виконується перевірка відхилень значень. Для цього застосовується код показаний на рисунках 2.19.

```
fig, ax1 = plt.subplots()
ax1.scatter(x = train['GrLivArea'], y = train['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()
```

Рисунок 2.19 – Перевірка відхилень

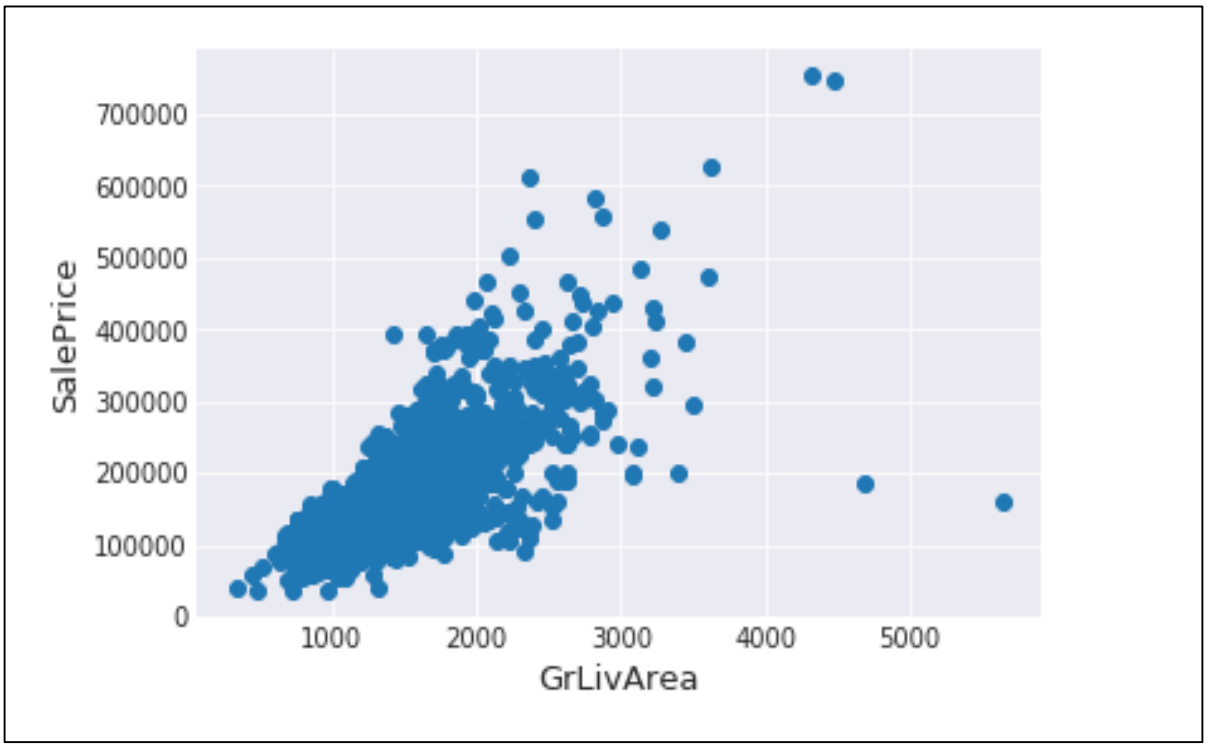

Рисунок 2.20 – Перевірка відхилень значень

Внизу праворуч є два відхилення з надзвичайно великими значеннями GrLivArea, які мають низьку ціну. Ці значення є величезними перебільшеннями. Тому їх можна видалити (рис. 2.21).

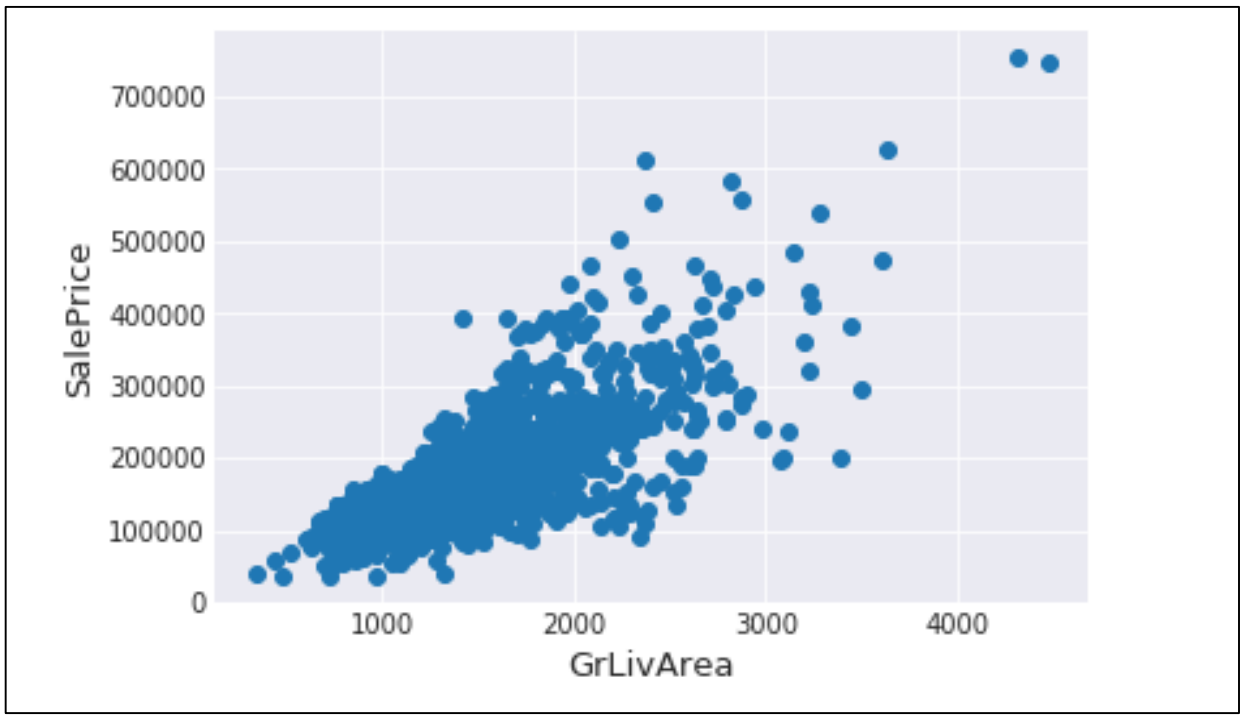

Рисунок 2.21 – Графік після видалення відхилень

Видалення викидів не завжди є безпечним. Ці два значення були видалені, оскільки вони дуже великі і дуже погані (надзвичайно великі площі за дуже низькими цінами).

Ймовірно, у навчальних даних є й інші викиди. Однак, їх видалення може погано вплинути на моделі, якщо в тестових даних також були викиди. Тому замість того, щоб видаляти їх усі, слід зробити деякі з моделей стійкими до них.

Отже, SalePrice – змінна, яку слід спрогнозувати. Для початку проводиться аналіз цієї змінної. Нижче на рисунку 2.22 зображено розподіл цін та цільову змінну.

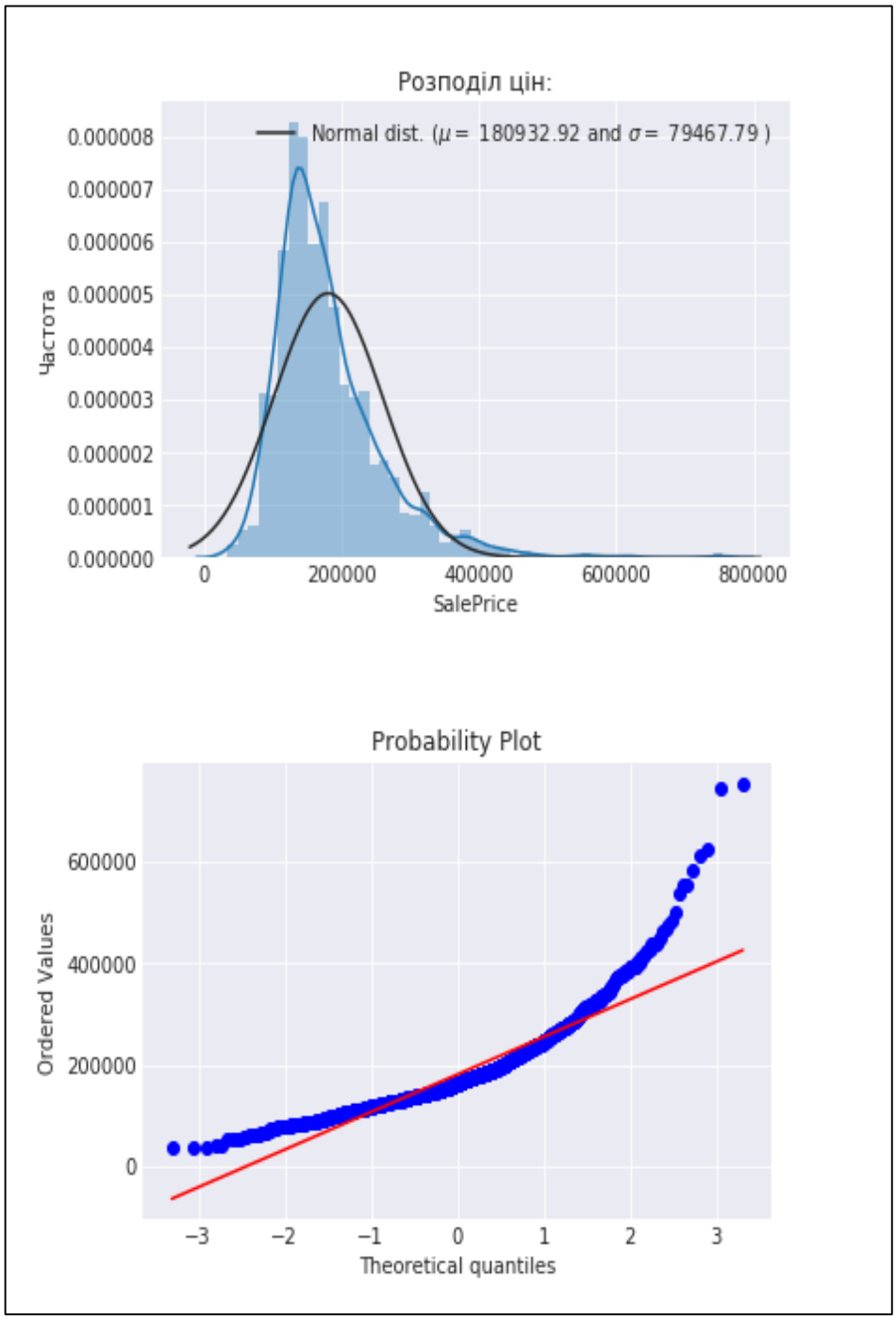

Рисунок 2.22 – Розподіл цін та цільова змінна

Цільова змінна є правосторонньою. Оскільки (лінійні) моделі люблять нормально розподілені дані, потрібно перетворити цю змінну і зробити її більш нормально розподіленою. Для цього застосовується логарифмічне перетворення (рис. 2.23).

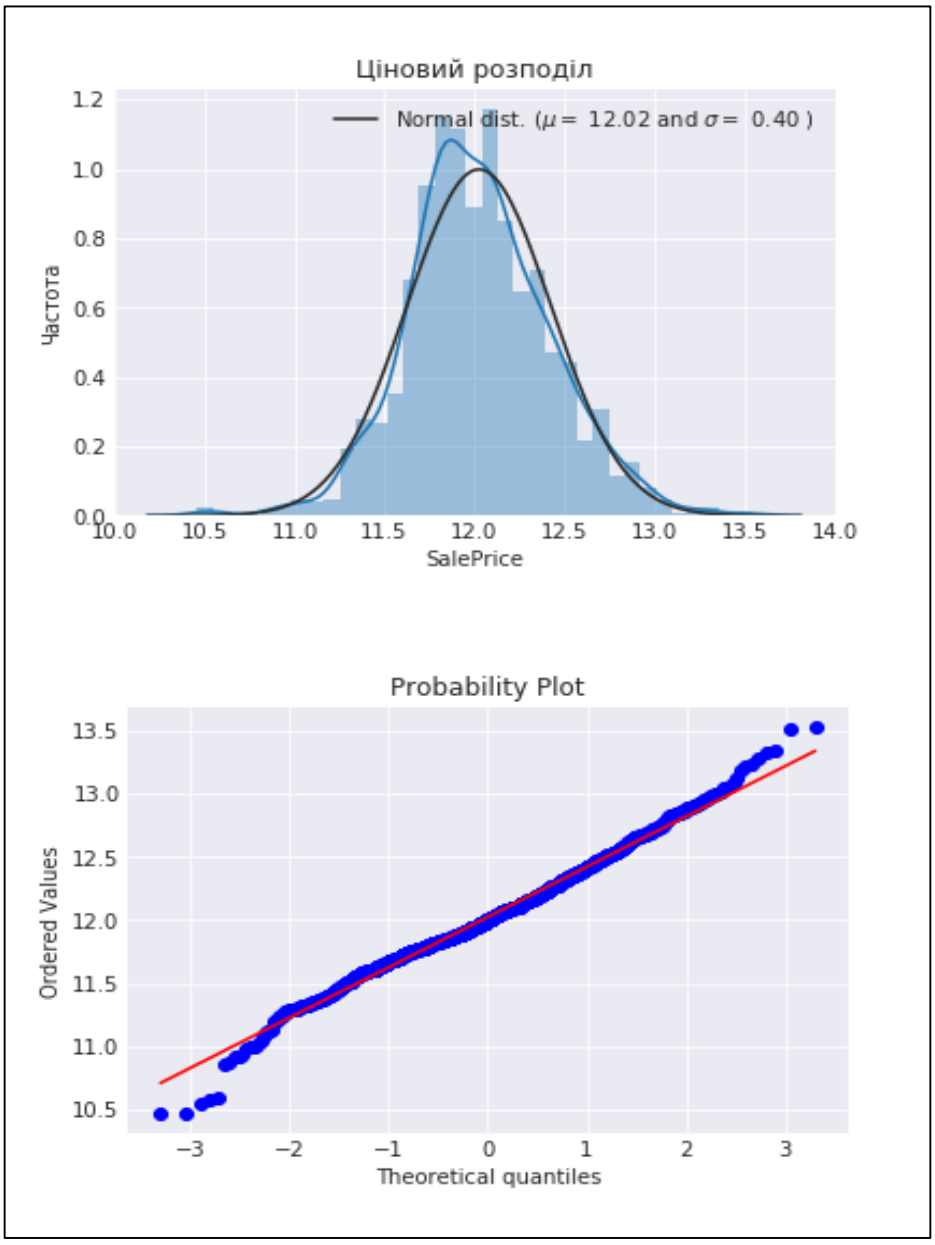

Рисунок 2.23 – Розподіл цін та цільова змінна після перетворення

Після логарифмічного перетворення перекіс виправлено, і дані виглядають більш рівномірно розподіленими.

Далі виконується об'єднання навчальних та тестових даних в один фрейм даних (рис. 2.24).

```
ntrain = train.shape[0]
ntest = test.shape[0]
y_train = train.SalePrice.values
all_data = pd.concat((train, test)).reset_index(drop=True)
all_data.drop(['SalePrice'], axis=1, inplace=True)
print("Весь розмір даних : {}".format(all_data.shape))
```

```
Весь розмір даних : (2917, 79)
```

Рисунок 2.24 – Об’єднані дані

Нижче вказані пропущені або відсутні дані (рис. 2.25)

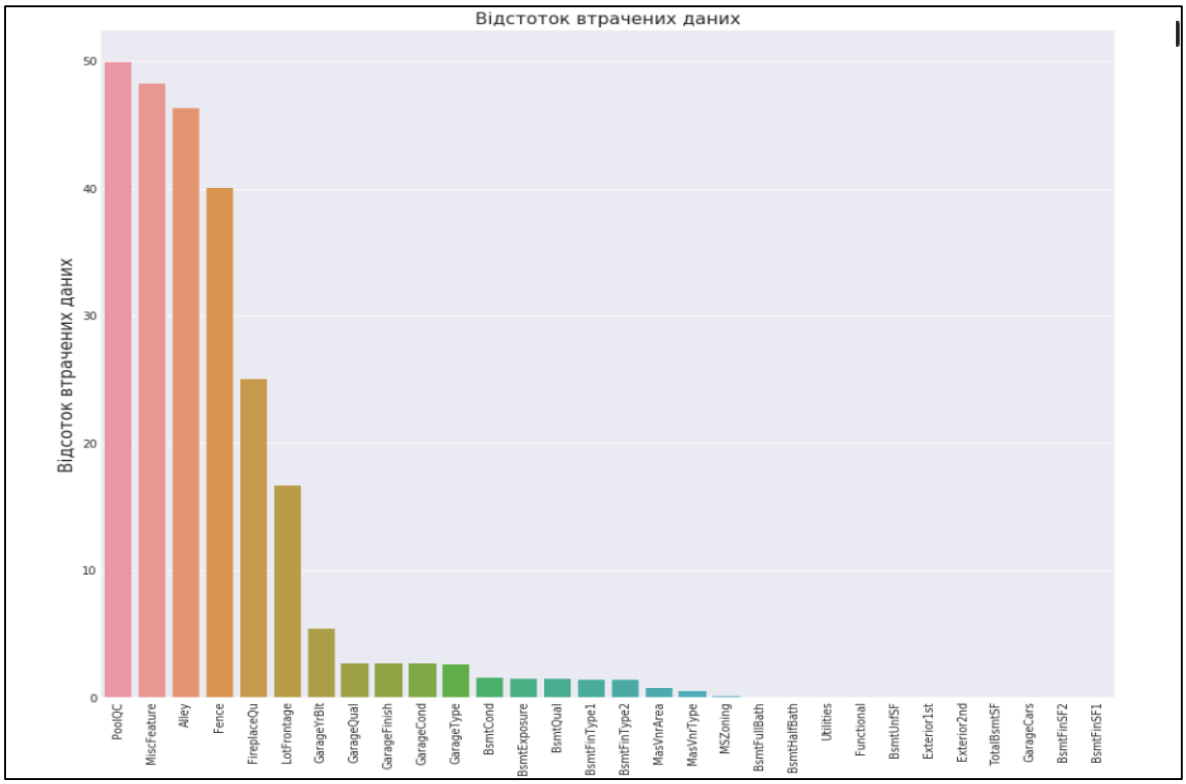

Рисунок 2.25 – Пропущені дані

Для того, щоб уникнути проблем з обробкою пропущених значень у подальшому аналізі або моделюванні даних, відбувається заміна пропущених даних значенням “None” де це можливо. Приклад заміни пропущених значень показано на рисунку 2.26.

```
all_data["MiscFeature"] = all_data["MiscFeature"].fillna("None")
```

Рисунок 2.26 – Заміна пропущених значень.

Також відсутні значення замінюються на 0. Наприклад: “GarageYrBlt”, “GarageArea” та “GarageCars” – заміна відсутніх даних на 0 (оскільки відсутність гаража означає відсутність автомобілів у такому гаражі) (рис. 2.27).

```
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
    all_data[col] = all_data[col].fillna(0)
```

Рисунок 2.27 – Заміна відсутніх значень на “0”

Перевірка пропущених значень (рис. 2.28):

```
#Перевіряємо чи залишились втрачені дані
all_data_na = (all_data.isnull().sum() / len(all_data)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Втрачені дані' :all_data_na})
missing_data.head()
```

Втрачені дані

Рисунок 2.28 – Перевірка пропущених значень

Також для покращення набору даних застосовується Перетворення Бокса-Кокса. Перетворення Бокса-Кокса є статистичним методом, який використовується для зменшення або усунення викривленості (асиметрії) в розподілі даних. Цей метод дозволяє нормалізувати дані та покращити їх властивості.

Основна мета перетворення Бокса-Кокса – зробити розподіл даних більш нормальним, тобто зрівняти його та зробити його ближчим до

симетричного розподілу. Код застосування перетворення Бокса-Кокса показано на рисунку 2.29.

```
skewness = skewness[abs(skewness) > 0.75]
print("Тут було {} відхилень(ня) числових характеристик до перетворення Кокса-Бокса".format(skewness.shape[0]))

from scipy.special import boxcox1p
skewed_features = skewness.index
lam = 0.15
for feat in skewed_features:
    all_data[feat] = boxcox1p(all_data[feat], lam)
```

```
Тут було 59 відхилень(ня) числових характеристик до перетворення Кокса-Бокса
```

Рисунок 2.29 – Перетворення Бокса-Кокса

До перетворення було 59 відхилень значень які мали викривлення більше 0.75.

Далі створюємо новий датафрейм з перетвореними категоріальними змінними (рис. 2.30).

```
train = all_data[:ntrain]
test = all_data[ntrain:]
```

Рисунок 2.30 – створення нового датафрейму

Поділ оброблених даних на тренувальні та тестові набори для використання в задачах машинного навчання (рис. 2.31).

```
train = all_data[:ntrain]
test = all_data[ntrain:]
```

Рисунок 2.31 – Поділ даних на вибірки

Після обробки вхідних даних можна переходити до моделювання та прогнозування ціни на будинки.

2.5 Висновок до розділу 2

В другому розділі розглянуто та описано методи машинного навчання та класифікації. Проведено аналіз методів прогнозування, в результаті чого було обрано декілька методів, які найкраще підходять для прогнозування цін на будинки, а саме ENet, KRR, XGBoost, GBoost, Lasso та LightGBM.

Розроблено алгоритм інформаційної технології з опрацювання вхідних даних та налаштування й застосування моделей.

Проведено розвідувальний аналіз даних, в результаті якого було виявлено основні ознаки, які найбільше впливають на ціну будинку.

Виконана попередня обробка вхідних даних, видалені аномальні значення, які могли негативно вплинути на прогнозування моделі. Вхідні дані підготовлені для застосування методів прогнозування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ЦІН НА ПРИВАТНІ ЖИТЛОВІ БУДИНКИ

3.1 Обґрунтування вибору мови та середовища розробки інформаційної технології

Оскільки, основна задача даної роботи – прогнозування цін методами машинного навчання, доцільно використати мову програмування Python.

Python – це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Вперше вона була випущена в 1991 році. Назва "Python", створена для того, щоб бути легкою та веселою, є відсиланням до британської комедійної групи Monty Python. Python має репутацію мови, зручної для початківців, замінивши Java як найпоширенішу вступну мову, оскільки вона вирішує більшу частину складних завдань для користувача, дозволяючи новачкам зосередитися на повному розумінні концепцій програмування, а не на дрібних деталях .

Python використовується для серверної веб-розробки, розробки програмного забезпечення, математики та системних сценаріїв, а також популярна для швидкої розробки додатків і як мова сценаріїв або клейка мова для зв'язування існуючих компонентів завдяки своїм високорівневим вбудованим структурам даних, динамічній типізації та динамічному зв'язуванню. Завдяки синтаксису, що легко засвоюється, та акценту на читабельності, витрати на підтримку програм зменшуються завдяки Python. Крім того, підтримка модулів і пакетів у Python полегшує створення модульних програм і повторне використання коду. Python – це мова з відкритим вихідним кодом, тому численні незалежні програмісти постійно створюють для неї бібліотеки та функціональні можливості [18].

Приклади використання Python:

- створення веб-додатків на сервері;

− побудова робочих процесів, які можна використовувати в поєднанні з програмним забезпеченням;

− підключення до систем баз даних;

− читання та модифікація файлів;

− виконання складних математичних обчислень;

− обробка великих даних;

− швидке створення прототипів;

− розробка готового до виробництва програмного забезпечення.

На професійному рівні Python чудово підходить для внутрішньої веб-розробки, аналізу даних, штучного інтелекту та наукових обчислень. Розробники також використовують Python для створення інструментів для підвищення продуктивності, ігор та десктопних додатків.

Можливості та переваги Python:

− сумісність з різними платформами, включаючи Windows, Mac, Linux, Raspberry Pi та інші;

− використовує простий синтаксис, в порівнянні з англійською мовою, що дозволяє розробникам використовувати менше рядків, ніж інші мови програмування;

− працює на системі інтерпретаторів, що дозволяє негайно виконувати код, прискорюючи створення прототипів;

− може бути процедурною, об'єктно-орієнтованою або функціональною.

Гнучкість Python. Python – мова з динамічною типізацією, є особливо гнучкою, усуває жорсткі правила створення функцій і пропонує більшу гнучкість у вирішенні проблем за допомогою різноманітних методів. Він також дозволяє компілювати та запускати програми аж до проблемної області, оскільки використовує перевірку типу під час виконання, а не під час компіляції.

Python і AI(штучний інтелект) Дослідники є фанатами Python. Google TensorFlow, а також інші бібліотеки (scikit-learn, Keras) створюють основу для

розробки AI завдяки зручності та гнучкості, яку він пропонує користувачам Python. Ці бібліотеки та їх доступність є критично важливими, оскільки вони дозволяють розробникам зосередитися на зростанні та розбудові.

Плюсом буде вміти працювати з індексними пакетами Python (PyPI) – це сховище програмного забезпечення для мови програмування Python. PyPI допомагає користувачам знаходити та встановлювати програмне забезпечення, розроблене спільнотою Python [18].

У мові Python існує безліч методів прогнозування, які можна використовувати для різних типів даних та завдань прогнозування. Оскільки основною задачею роботи є прогнозування цін на будинки, потрібно обрати методи прогнозування, які покажуть найбільш точний результат. Отже, наведемо деякі методи прогнозування.

На сьогоднішній день існує велика кількість середовищ для роботи з мовою програмування Python, а саме:

– PyCharm;

– Anaconda Jupiter Notebook;

– Kite;

– Spyder;

– IDLE;

– Visual Studio Code;

– Atom;

– PyDev, та багато інших

Також існують онлайн середовища для написання коду мовою програмування Python, до таких середовищ можна віднести:

– Pythonanywhere;

– інші онлайн інтерпретатори Python;

– Kaggle Notebook.

У задачах машинного навчання доречно використати онлайн середовище розроблення від системи Kaggle, яке називається «Notebooks». Система Kaggle дозволяє досліджувати та запускати код машинного навчання

за допомогою Kaggle Notebooks, хмарного обчислювального середовища, яке забезпечує відтворюваний та спільний аналіз.

Варто зазначити, що використовуючи систему Kaggle Notebooks, можливо запустити код на хмарному обчислювальному середовищі із виділеними ресурсами оперативної пам'яті та до 20 ГБ вихідних даних із ноутбука можна зберегти на диск у /kaggle/working. Ці дані зберігаються автоматично і ви можете повторно використовувати їх.

Notebooks – це більше, ніж просто редактор коду. Це обчислювальне середовище, створене для полегшення відтворення наукової роботи з даними.

У IDE Notebooks у вас є доступ до інтерактивного сеансу, що працює в контейнері Docker із попередньо встановленими пакетами, можливість монтувати версійні джерела даних, настроюванні обчислювальні ресурси, такі як графічні процесори тощо [19].

Також є можливість спільного розроблення коду чи проекту, варто просто долучити учасника і надати статус співавтора та надати дозвіл на редагування коду у ноутбуці.

Перевагою Kaggle над іншими онлайн редакторами є те, що у системі є велика кількість наборів даних (датасетів) на різні тематики у вільному доступі, кожний, який бажає може працювати з цими даними, обговорювати, коментувати вже існуючі попередньо розроблені проекти (ноутбуки) та на основі вже наявних вдосконалити його, або за необхідності розробити власний ноутбук самотужки.

Отже, для виконання задачі по прогнозуванні ціни продажу будинків обрано мову розроблення та програмування Python, оскільки найбільше підходить для роботи з методами машинного навчання, у якості середовища розроблення програмного коду на мові Python обрано систему Kaggle IDE Notebooks, хмарних ресурсів для обчислення у безкоштовному режимі цілком достатньо, щоб виконати дану роботу.

У мові програмування Python існує неймовірно велика кількість бібліотек.

У даній роботі для її реалізації будуть використовуватись наступні бібліотеки:

NumPy – це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій з цими масивами [20].

Pandas – це зручний і швидкий інструмент для роботи з даними, що володіє великим функціоналом [21].

Matplotlib – це бібліотека двовимірної графіки для мови програмування Python, за допомогою якої можна створювати високоякісні малюнки різних форматів[22] .

Seaborn – бібліотека візуалізації даних Python, заснована на Matplotlib. Забезпечує інтерфейс високого рівня для малювання привабливої та інформативної статистичної графіки [23].

Plotly – бібліотека, яка дозволяє швидко візуалізувати дані та завдяки інтерактивності допомагає краще в них розібратися, тобто є можливість заглибитися в деталі при необхідності.

Mpl_toolkits – надає основні інструменти тривимірної побудови графіків (розсіювання, пошуку, лінії, сітки) .

SciPy – це пакет прикладних математичних процедур, заснований на розширенні Numpy Python. З SciPy інтерактивний сеанс Python перетворюється в таке ж повноцінне середовище обробки даних і прототип складних систем, як MATLAB, IDL, Octave, R-Lab і SciLab .

WordCloud – хмара, наповнена безліччю слів різного розміру, які показують частоту чи важливість кожного слова . Folium – бібліотека для візуалізації географічних даних і інформації, яка містить координати та місця розташування .

Scikit-Learn – бібліотека для машинного навчання і прогнозної аналітики. Вона містить ряд методів, що охоплюють: алгоритми класифікації та регресії, кластеризації, валідацію і вибір моделей. Також її можна застосовувати для зменшення розмірності даних і виділення ознак [23].

Машинне навчання в Scikit-Learn полягає в тому, щоб імпортувати правильні модулі та запустити метод підбору моделі. Складніше очистити, відформатувати та підготувати дані, а також підібрати оптимальні вхідні значення і моделі. Тому перш ніж взятися за Scikit-Learn, потрібно, по-перше, відпрацювати навички роботи з Python і pandas, щоб навчитися якісно готувати дані, а по-друге, освоїти теорію і математичну основу різних моделей прогнозування та класифікації, щоб розуміти, що відбувається з даними при їх застосуванні .

Pandas_profiling – бібліотека для швидкого розвідувального аналізу даних. Результати її роботи виражаються не у вигляді якихось окремих показників, а в формі досить докладного HTML-звіту, що містить велику частину тих відомостей проаналізованих даних, які можливо буде потрібн знати перед тим, як приступати до більш щільної роботи з ними .

XGBoost (скорочення від EXtreme Gradient Boosting) – популярна бібліотека машинного навчання, що реалізує модель градієнтного бустингу, що представляє альтернативу регресійним методам і нейронних мереж. Метод полягає в створенні ансамблю послідовно уточнюючих один одного дерев рішень [24].

LightGBM – швидка, розподілена високопродуктивна платформа з градієнтним прискоренням (GBDT, GBRT, GBM) на основі алгоритмів дерева прийняття рішень. Бібліотека використовується для ранжирування, класифікації та багатьох інших завдань машинного навчання .

Warnings – бібліотека, яка видає попереджувальні повідомлення, як правило, видаються в ситуаціях, коли корисно попередити користувача про певний стан програми, коли ця умова (як правило) не вимагає збільшення винятку та припинення програми. Наприклад, можна створити попередження, коли програма використовує застарілий модуль .

3.2 Удосконалення базових моделей

Для визначення стратегії перехресної перевірки використовується функція cross_val_score з бібліотеки Sklearn. Однак ця функція не має атрибуту shuffle, тому додається рядок коду, щоб перемішати набір даних перед перехресною перевіркою (рис. 3.1).

```
n_folds = 5

def rmse_cv(model):
    kf = KFold(n_folds, shuffle=True, random_state=42).get_n_splits(train.values)
    rmse= np.sqrt(-cross_val_score(model, train.values, y_train, scoring="neg_mean_squared_error", cv = kf))
    return(rmse)
```

Рисунок 3.1 – Змішування набору даних

LASSO Regression. Ця модель може бути дуже чутливою до викидів. Тому потрібно зробити її більш стійкою до них. Для цього використовується метод Robustscaler (рис. 3.2).

```
lasso = make_pipeline(RobustScaler(), Lasso(alpha =0.0005, random_state=1))
```

Рисунок 3.2 – Використання методу Robustscaler для моделі LASSO

Модель Elastic Net Regression зроблено стійкою до нестандартних ситуацій за допомогою методу Robustscaler (рис. 3.3).

```
ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005, l1_ratio=.9, random_state=3))
```

Рисунок 3.3 – Використання методу Robustscaler для моделі ENet

Оптимізація моделі Kernel Ridge Regression (рис. 3.4).

```
KRR = KernelRidge(alpha=0.6, kernel='polynomial', degree=2, coef0=2.5)
```

Рисунок 3.4 – оптимізація Kernel Ridge Regression

Для точного прогнозу було налаштовано такі параметри:

- alpha=0.6 — це регуляризаційний параметр, який контролює баланс між точністю моделі та її здатністю узагальнювати;
- kernel='polynomial' — ядро, яке визначає, як перетворюються вхідні дані. У цьому випадку використовується поліноміальне ядро;
- degree=2 — ступінь полінома. Поліном другого ступеня означає, що модель враховує квадратичні взаємозв'язки між вхідними ознаками;
- coef0=2.5 — константа, яка додається до поліноміального ядра перед піднесенням до степеня. Впливає на гнучкість ядра.

Gradient Boosting Regression з втратою huber, що робить її стійкою до викидів показано на рисунку 3.5.

```
GBoost = GradientBoostingRegressor(n_estimators=3000, learning_rate=0.05,
                                   max_depth=4, max_features='sqrt',
                                   min_samples_leaf=15, min_samples_split=10,
                                   loss='huber', random_state =5)
```

Рисунок 3.5 – Gradient Boosting Regression

Давайте розберемо налаштування моделі GradientBoostingRegressor:

- n_estimators=3000, кількість дерев. Це кількість ітерацій (або базових моделей), які використовуються в бустингу. Значення 3000 означає, що модель створюватиме 3000 дерев. Це багато і може покращити точність, але збільшить час навчання і ризик перенавчання.
- learning_rate=0.05, це швидкість навчання. Кожне дерево вносить невелику поправку до прогнозу. Менше значення (0.05) означає, що модель

навчається повільніше, що може зменшити ризик перенавчання. Як правило, якщо зменшується learning_rate, потрібно збільшити n_estimators.

- max_depth=4, максимальна глибина дерев. Обмежує глибину кожного дерева. Значення 4 означає, що дерева матимуть до 4 рівнів. Це допомагає контролювати складність моделі і запобігає перенавчанню.

- max_features='sqrt', кількість ознак для розгляду при розщепленні: 'sqrt' означає, що при кожному розщепленні буде використовуватися квадратний корінь від кількості загальних ознак. Це стандартна практика для зменшення кореляції між деревами та покращення узагальнення.

- min_samples_leaf=15, мінімальна кількість зразків у листку. Обмежує мінімальну кількість зразків, які повинні бути у вузлі, щоб він вважався листком. Значення 15 допомагає уникати занадто маленьких вузлів, що сприяє стабільності і зменшує ризик перенавчання.

- min_samples_split=10, мінімальна кількість зразків для розділення вузла. Вузол розщеплюється тільки якщо в ньому є не менше 10 зразків. Це також допомагає зменшити перенавчання.

- Loss='huber', функція втрат: huber є стійкою до викидів. Вона комбінує переваги квадратичної функції втрат (для малих відхилень) і абсолютної похибки (для великих відхилень).

- random_state=5, випадковий стан. Контролює випадковість в алгоритмі, забезпечуючи відтворюваність результатів.

Оптимізація моделі XGBoost (рис. 3.6).

```
model_xgb = xgb.XGBRegressor(colsample_bytree=0.5, gamma=0.05,
                             learning_rate=0.05, max_depth=3,
                             min_child_weight=1.8, n_estimators=2200,
                             reg_alpha=0.5, reg_lambda=0.8,
                             subsample=0.5, silent=1,
                             random_state =7, nthread = -1)
```

Рисунок 3.6 – Оптимізація моделі XGBoost

Розглянемо налаштування моделі XGBRegressor:

- colsample_bytree=0.5, відсоток ознак для кожного дерева. Значення 0.5 означає, що кожне дерево вибиратиме 50% ознак випадковим чином. Це допомагає зменшити кореляцію між деревами та зменшити перенавчання.

- gamma=0.05, мінімальне зменшення втрат для розщеплення вузла. Розщеплення відбувається тільки якщо це зменшує втрати більше ніж на 0.05. Це допомагає уникати несуттєвих розщеплень.

- learning_rate=0.05, швидкість навчання. Кожне дерево вносить невелику поправку до прогнозу. Значення 0.05 означає, що навчання проходить повільніше, що зменшує ризик перенавчання.

- max_depth=3, максимальна глибина дерев. Значення 3 обмежує дерева до трьох рівнів. Це робить модель менш схильною до перенавчання.

- min_child_weight=1.8, мінімальна вага для розщеплення вузла. Вузол розщеплюється тільки якщо сукупна вага (або кількість зразків) в кожному з дочірніх вузлів перевищує 1.8. Це параметр регуляризації, що запобігає утворенню вузлів із малою кількістю даних.

- n_estimators=2200, кількість дерев. Це кількість базових моделей (дерев), які будуть побудовані. Значення 2200 забезпечує більш точні результати, але збільшує час навчання.

- reg_alpha=0.5, L1-регуляризація. Цей параметр додає штраф за абсолютні значення ваг моделі, допомагаючи зробити модель простішою та зменшити перенавчання.

- reg_lambda=0.8, L2-регуляризація. Цей параметр додає штраф за квадрат ваг, також зменшуючи ризик перенавчання. Значення 0.8 є помірним і дозволяє досягти балансу між точністю і регуляризацією.

- subsample=0.5, відсоток вибірки для кожного дерева. Значення 0.5 означає, що кожне дерево тренується на 50% випадково обраних зразків з повного датасету. Це допомагає знизити ризик перенавчання.

- silent=1, приглушення виводу. Забезпечує, що під час навчання модель не виводитиме повідомлення. (У нових версіях параметр може бути замінений на verbosity).

- random_state=7, випадковий стан. Забезпечує відтворюваність результатів, фіксуючи порядок випадкових вибірок.

- nthread=-1, кількість потоків. Значення -1 використовує всі доступні ядра процесора для паралельного обчислення, що пришвидшує навчання.

Оптимізація моделі LightGBM (рис. 3.7).

```
model_lgb = lgb.LGBMRegressor(objective='regression',num_leaves=5,
                              learning_rate=0.05, n_estimators=720,
                              max_bin = 55, bagging_fraction = 0.8,
                              bagging_freq = 5, feature_fraction = 0.2,
                              feature_fraction_seed=9, bagging_seed=9,
                              min_data_in_leaf =6, min_sum_hessian_in_leaf = 11)
```

Рисунок 3.7 – Оптимізація моделі LightGBM

Розберемо параметри налаштувань моделі LGBMRegressor:

objective='regression', ціль функці - регресія, тобто модель прогнозує числове значення. Це стандартний вибір для задач прогнозування, таких як ціна житла.

num_leaves=5, кількість листків у кожному дереві. Значення 5 обмежує кількість листків у дереві, роблячи його простішим і менш схильним до перенавчання.

learning_rate=0.05, швидкість навчання. Низька швидкість навчання (0.05) дозволяє поступово оновлювати ваги та знижує ризик перенавчання.

n_estimators=720, кількість дерев. Модель використовує 720 дерев, що забезпечує достатню кількість ітерацій для точного прогнозу.

max_bin=55, максимальна кількість бінів для дискретизації. Число 55 означає, що числові ознаки будуть дискретизовані в 55 груп. Це впливає на швидкість і точність обчислень.

bagging_fraction=0.8, частка вибірки для кожного дерева. Модель тренується на 80% випадково обраних даних, що зменшує кореляцію між деревами та запобігає перенавчанню.

bagging_freq=5, частота виконання bagging, Кожні 5 ітерацій модель оновлює підмножину вибірки для створення дерев. Це додає варіативності в навчанні.

feature_fraction=0.2, частка ознак для кожного дерева. Кожне дерево вибирає 20% доступних ознак. Це сприяє зменшенню перенавчання.

feature_fraction_seed=9 і bagging_seed=9, випадкові стани для feature_fraction та bagging. Ці параметри задають початкове значення для генератора випадкових чисел, забезпечуючи відтворюваність результатів.

min_data_in_leaf=6, мінімальна кількість даних у листі. Листок буде створений тільки якщо в ньому буде щонайменше 6 зразків. Це обмеження запобігає створенню листків з невеликою кількістю даних.

min_sum_hessian_in_leaf=11, мінімальна сума гессіанів у листі. Значення 11 означає, що листок створюється лише якщо сума другого похідного (гессіану) по всіх зразках у листі перевищує цей поріг. Це параметр регуляризації, який додатково захищає від перенавчання.

Обчислення перехресної перевірки середньоквадратичної помилки (RMSE) для моделей. Значення середнього (mean) та стандартного відхилення (std) оцінки “score” для моделей показано на рисунку 3.8.

```
score = rmse_cv(ENet)
print("ElasticNet score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
ElasticNet score: 0.1116 (0.0072)
```

```
score = rmse_cv(KRR)
print("Kernel Ridge score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Kernel Ridge score: 0.1152 (0.0072)
```

```
score = rmse_cv(GBoost)
print("Gradient Boosting score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Gradient Boosting score: 0.1173 (0.0076)
```

```
score = rmse_cv(model_xgb)
print("Xgboost score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Xgboost score: 0.1167 (0.0062)
```

```
score = rmse_cv(model_lgb)
print("LGBM score: {:.4f} ({:.4f})\n" .format(score.mean(), score.std()))
```

```
LGBM score: 0.1144 (0.0056)
```

Рисунок 3.8 – Середньоквадратична похибка перехресної перевірки базових моделей

Отже, серед обраних моделей найкращий результат перехресної перевірки середньоквадратичної похибки показала модель LightGBM = 0.114.

3.3 Застосування стекування моделей для покращення результатів прогнозу

Найпростіший підхід до стекування – усереднення базових моделей. Для цього підходу для базових моделей створюється новий клас для розширення scikit-learn моделлю, а також для інкапсуляції та повторного використання коду (успадкування) (рис. 3.9).

```
class AveragingModels(BaseEstimator, RegressorMixin, TransformerMixin):
    def __init__(self, models):
        self.models = models

    #ми визначаємо клони оригінальних моделей, в які вписуємо дані
    def fit(self, X, y):
        self.models_ = [clone(x) for x in self.models]

        # Навчання клонованої базової моделі
        for model in self.models_:
            model.fit(X, y)

        return self

    #Тепер ми робимо прогнози для клонованих моделей і усереднюємо їх
    def predict(self, X):
        predictions = np.column_stack([
            model.predict(X) for model in self.models_
        ])
        return np.mean(predictions, axis=1)
```

```
+ Code  + Markdown
```

```
averaged_models = AveragingModels(models = (ENet, GBoost, KRR, lasso))

score = rmse_cv(averaged_models)
print(" Усереднена оцінка базових моделей: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Усереднена оцінка базових моделей: 0.1088 (0.0072)
```

Рисунок 3.9 – Усереднення базових моделей

Найпростіший підхід до стекування дійсно покращує результат. Це спонукає піти далі і дослідити менш простий підхід до стекування.

В цьому підході додається метамодель до усереднених базових моделей і використовуються прогнози згинів цих базових моделей для навчання мета-моделі.

Процедуру навчання можна описати наступним чином:

– розділюється загальна навчальна множина на дві непересічні множини (тут train і holdout);

– навчається кілька базових моделей на першій частині (train);

– тестуються базові моделі на другій частині (holdout);

– використовується прогнозування з попереднього пункту) (так звані прогнозування поза межами) як вхідні дані, а правильні відповіді (цільова змінна) – як вихідні дані для навчання моделі вищого рівня, яка називається мета-моделлю.

Перші три кроки виконуються ітеративно. Якщо взяти для прикладу 5-кратне укладання, спочатку навчальні дані розподіляються на 5 частин. Потім виконується 5 ітерацій. На кожній ітерації базова модель навчається, а прогнозуємо на тому, що залишився (склад, що витримує).

Таким чином, після 5 ітерацій всі дані будуть використовуються для отримання прогнозів, які потім використовуються як нова функцію для навчання метамоделі на кроці 4.

У частині прогнозування прогнози всіх базових моделей зводяться до середнього значення на тестових даних і використовуються як мета-ознаки, на основі яких робиться остаточне прогнозування за допомогою метамоделі (рис. 3.10).

```
class StackingAveragedModels(BaseEstimator, RegressorMixin, TransformerMixin):
    def __init__(self, base_models, meta_model, n_folds=5):
        self.base_models = base_models
        self.meta_model = meta_model
        self.n_folds = n_folds

    # Знову підбираємо дані клонів оригінальних моделей
    def fit(self, X, y):
        self.base_models_ = [list() for x in self.base_models]
        self.meta_model_ = clone(self.meta_model)
        kfold = KFold(n_splits=self.n_folds, shuffle=True, random_state=156)

        # Навчаємо клоновані базові моделі, а потім створюємо нестандартні прогнози,
        # які необхідні для навчання клонованої мета-моделі
        out_of_fold_predictions = np.zeros((X.shape[0], len(self.base_models)))
        for i, model in enumerate(self.base_models):
            for train_index, holdout_index in kfold.split(X, y):
                instance = clone(model)
                self.base_models_[i].append(instance)
                instance.fit(X[train_index], y[train_index])
                y_pred = instance.predict(X[holdout_index])
                out_of_fold_predictions[holdout_index, i] = y_pred

        # Тепер навчаємо клоновану мета-модель, використовуючи передбачення out of fold як нову функцію
        self.meta_model_.fit(out_of_fold_predictions, y)
        return self

    #Зробили прогнози всіх базових моделей на тестових даних і використовуємо усереднені прогнози як
    #мета-ознаки для остаточного прогнозу, який виконується мета-моделлю
    def predict(self, X):
        meta_features = np.column_stack([
            np.column_stack([model.predict(X) for model in base_models]).mean(axis=1)
            for base_models in self.base_models_ ])
        return self.meta_model_.predict(meta_features)
```

Рисунок 3.10 – Навчання метамоделі

Усереднена оцінка моделей. Щоб зробити два підходи порівняними (використовуючи однакову кількість моделей), усереднюється Enet KRR та Gboost, а потім додається lasso як метамодель (рис. 3.11).

```
stacked_averaged_models = StackingAveragedModels(base_models = (ENet, GBoost, KRR),
                                                 meta_model = lasso)

score = rmse_cv(stacked_averaged_models)
print("Усереднений бал моделей: {:.4f} ({:.4f})".format(score.mean(), score.std()))
```

```
Усереднений бал моделей: 0.1083 (0.0071)
```

Рисунок 3.11 – Усереднений бал моделей

3.4 Тестування роботи програми

RMSE (Root Mean Square Error) – це метрика оцінки точності моделі, яка використовується для вимірювання відхилень між спостережуваними і прогнозованими значеннями. Вона широко використовується в задачах регресії для оцінки рівня помилки моделі. В даній метриці чим менше значення, тим більша точність моделі. Менше значення RMSE вказує на те, що середньоквадратична похибка між прогнозованими і спостережуваними значеннями є меншою, що в свою чергу означає, що модель має кращу прогностичну точність.

Результат точності прогнозу стекової моделі StackedRegressor за метрикою RMSE показано на рисунку 3.12.

```
stacked_averaged_models.fit(train.values, y_train)
stacked_train_pred = stacked_averaged_models.predict(train.values)
stacked_pred = np.expm1(stacked_averaged_models.predict(test.values))
print(rmse(y_train, stacked_train_pred))
```

```
0.074591171218
```

Рисунок 3.12 - Результат точності прогнозування моделі StackedRegressor

Результат точності прогнозування моделі XGBoost за метрикою RMSE (рис. 3.13):

```
model_xgb.fit(train, y_train)
xgb_train_pred = model_xgb.predict(train)
xgb_pred = np.expm1(model_xgb.predict(test))
print(rmse(y_train, xgb_train_pred))
```

```
0.080419148461
```

Рисунок 3.13 – Результат точності прогнозування моделі XGBoost

Результат точності прогнозування моделі LightGBM за метрикою RMSE (рис. 3.14):

```
model_lgb.fit(train, y_train)
lgb_train_pred = model_lgb.predict(train)
lgb_pred = np.expm1(model_lgb.predict(test.values))
print(rmse(y_train, lgb_train_pred))
```

```
0.0732082944168
```

Рисунок 3.14 – Результат точності прогнозування моделі LightGBM

Результати прогнозування занесено до таблиці 3.1.

Таблиця 3.1 – Результати точності прогнозування

Власні результати		Результати аналога	
Назва моделі	Похибка за метрикою RMSE	Назва моделі	Похибка за метрикою RMSE
LightGBM	0.073	LightGBM	0.119
XGBoost	0.080	XGBoost	0.114
StackedRegressor	0.074	Lasso	0.112

З таблиці видно що в порівнянні з кращим результатом аналога точність прогнозування покращились на 34%, тому основне завдання, а саме підвищення точності прогнозування цін на приватні житлові будинки методами машинного навчання виконано.

Графік прогнозу моделі LightGBM на 40 значенях (рис. 3.15).

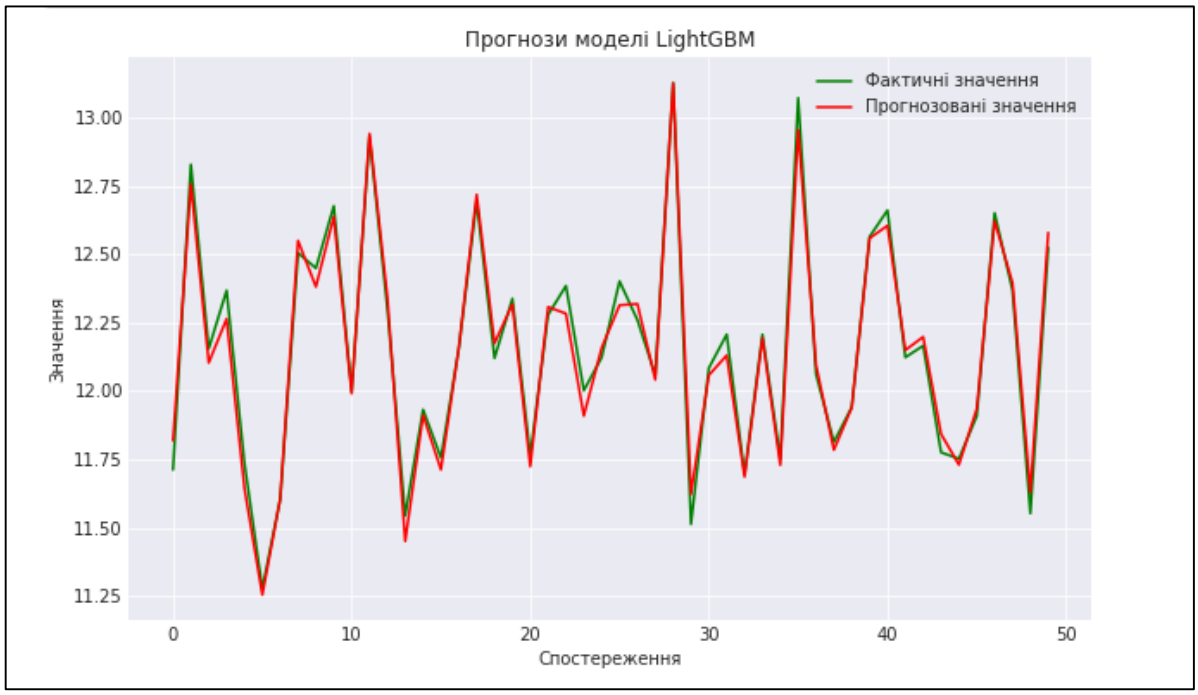

Рисунок 3.15 – Графік прогнозу моделі LightGBM

Графік прогнозу моделі XGBoost перших 40 значень (рис. 3.16).

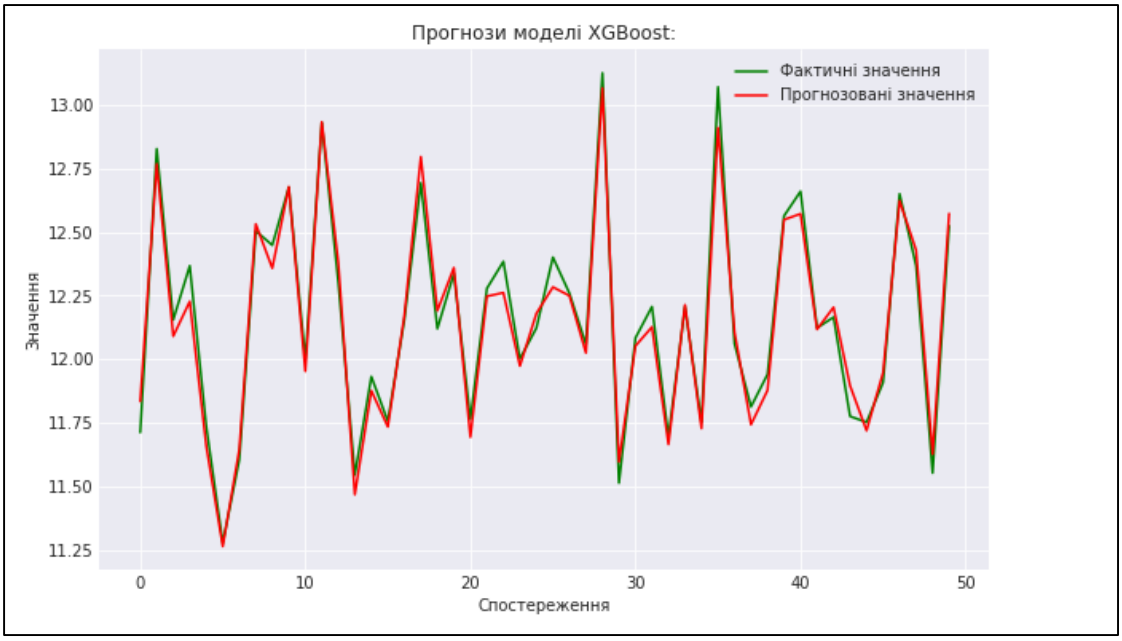

Рисунок 3.16 – Графік прогнозу моделі XGBoost

Графік прогнозу моделі StackedRegressor на перших 40 значень (рис. 3.17).

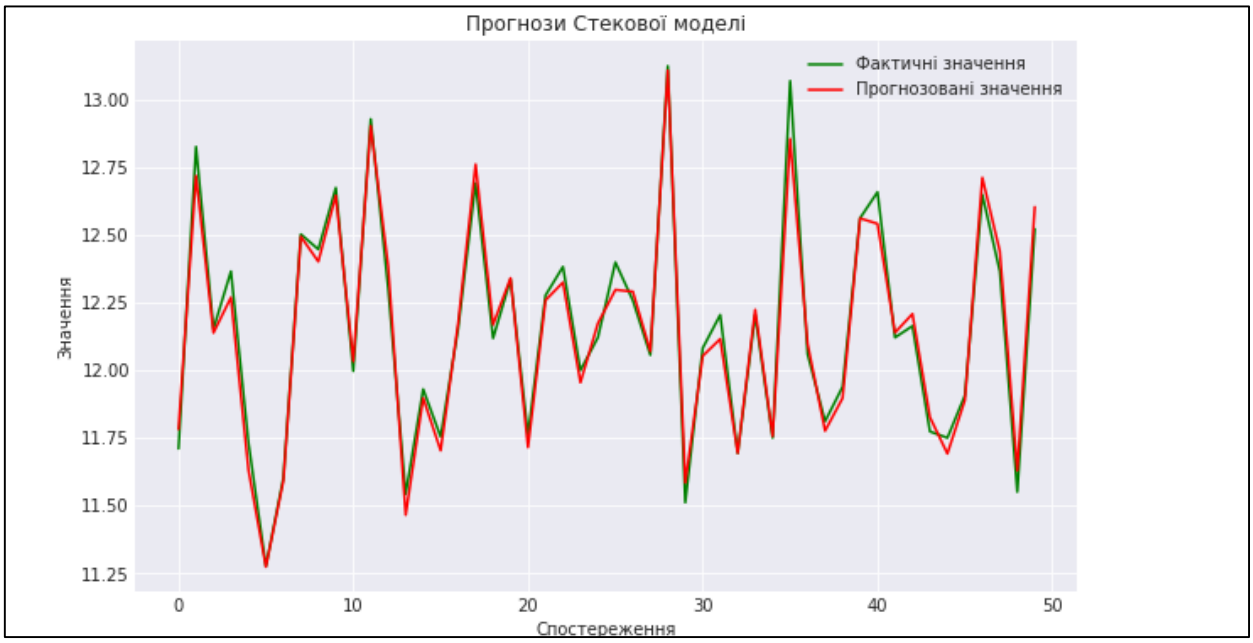

Рисунок 3.17 – Графік прогнозу моделі StackedRegressor

Проаналізувавши дані графіки можна сказати, що прогноз використаних моделей є досить точним. В цілому дані моделі доцільно використовувати для прогнозування цін на будинки.

Фінальне прогнозування ансамблю показано на рисунку 3.18.

```
ensemble = stacked_pred*0.5 + xgb_pred*0.25 + lgb_pred*0.25
```

Рисунок 3.18 – Прогнозування ансамблю

Завантаження результату прогнозування показано на рисунку 3.19.

```
sub = pd.DataFrame()
sub['Id'] = test_ID
sub['SalePrice'] = ensemble
sub.to_csv('submission.csv',index=False)
```

Рисунок 3.19 – Завантаження результату прогнозування

3.5 Висновок до розділу 3

У третьому розділі було проведено оптимізацію моделей для покращення точності прогнозування та показано їх середньоквадратичну похибку. За перевіркою середньоквадратичної похибки найбільш точною виявилась модель LightGBM із результатом 0.073. Додатково було розглянуто покращення прогнозування за допомогою стекування моделей. Стекування виявилося ефективним методом, який поєднує прогнозування базових моделей та створює злиту модель. Цей підхід дозволив отримати досить точний результат, що дорівнює 0.074. Також було розроблено інформаційну технологію прогнозування цін на будинки, показано її граф-схему та алгоритм роботи.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	2	2
Ринкові переваги (наявність аналогів)	3	2	2
Ринкові переваги (ціна продукту)	3	3	3
Ринкові переваги (технічні властивості)	2	2	2
Ринкові переваги (експлуатаційні витрати)	3	2	3

Ринкові перспективи (розмір ринку)	1	1	2
Ринкові перспективи (конкуренція)	2	2	3
Практична здійсненність (наявність фахівців)	3	3	3
Практична здійсненність (наявність фінансів)	1	2	2
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	2	2	3
Практична здійсненність (розробка документів)	2	2	2
Сума балів	25	24	28
Середньоарифметична сума балів, СБ	25,7		

За результатами розрахунків, наведених в таблиці 1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання – середній.

4.2 Розрахунок витрат на здійснення науково-дослідної розробки інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р} \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	30000	1363	40	54520
Розробник	25000	1136	40	45440
Консультанти	15000	682	15	10230
Всього:	110190			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмування	8	5	98,9	1,36	791,2
Налагодження	8	5	98,9	1,36	791,2
Тестування	8	4	92,4	1,27	739,2
Всього					2321,6

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (110190 + 2321{,}6) = 11251{,}2 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (110190 + 2321{,}6 + 11251{,}2) = 27227{,}9 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Розрахунок витрат на комплектуючі. Витрати на комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де $Н_і$ – кількість комплектуючих і-го виду, шт.; $Ц_і$ – ціна комплектуючих і-го виду, грн.; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів комплектуючих.

Таблиця 4.4 – Комплектуючі, використані на розробку

Найменування комплектуючих	Ціна за одиницю, грн.	Витрачено	Вартість витрачених комплектуючих, грн.
Флеш накопичувач	300	1	300
Всього, з врахуванням коефіцієнта транспортних витрат			330

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і}, \quad (4.5)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_{і}$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_{і}$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 4.7 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Visual Studio	450	1	450
Jupyter Notebook	0	1	0
Всього, з врахуванням коефіцієнта інсталяції та налагодження			495

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_{б}}{Т_{в}} \cdot \frac{t}{12}, \quad (4.6)$$

де $Ц_{б}$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_{в}$ – термін корисного використання основного фонду, роки.

Таблиця 4.8 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Комп'ютери Asus (2 шт)	40000	5	1.4	1867
Монітори Asus (2 шт)	6000	5	1.4	280
Маршрутизатор TP-Link	1000	5	1.4	23
Всього	2170			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Комп'ютери Asus	0.5	320
Монітори Asus	0.1	320
Маршрутизатор TP-Link	0.014	320

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0{,}5 \cdot 320 \cdot 7{,}5 \cdot 0{,}5}{0{,}98} + \frac{0{,}1 \cdot 320 \cdot 7{,}5 \cdot 0{,}5}{0{,}98} + \frac{0{,}014 \cdot 320 \cdot 7{,}5 \cdot 0{,}5}{0{,}98} = 752 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (110190 + 2321{,}6) \cdot \frac{70}{100} = 78758 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$Внзв = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (110190 + 2321{,}6) \cdot \frac{130}{100} = 146265 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} =$$
$$= 110190 + 2321{,}6 + 11251{,}2 + 27227{,}9 + 330 + 495 + 2170 + 752 + 78758 +$$
$$146265 = 379760{,}7 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{379760{,}7}{0{,}5} = 759521{,}4 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії розробки дослідного зразка, то η=0,5.

4.3 Розрахунок економічної ефективності науково-технічної розробки інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна

інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.6)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка

податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((35000 - 32000) \cdot 1100 - 35000 \cdot (1000 - 1100)) \cdot 0{,}8333 \cdot 0{,}5 \cdot \left(1 - \frac{18}{100}\right) = 1332980 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{1332980}{(1+0{,}1)^1} = 1211800 \text{ грн.},$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 1 \cdot 759521{,}4 = 759521{,}4 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=1...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{\text{абс}} = \text{ПП} - PV = 1211800 - 759521{,}4 \ = 452278{,}6 \text{ грн.,}$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар’єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[1]{1 + \frac{452278,6}{759521,4}} - 1 = 0,26,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Далі розраховуємо період окупності інвестицій $Т_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$Т_о = \frac{1}{E_в} = \frac{1}{0,26} = 3,8 \text{ року.}$$

Оскільки $Т_о$<1…5-х років, то це свідчить про комерційну привабливість науково-технічної розробки інформаційної технології передбачення цін на приватні житлові будинки методами машинного навчання і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновок до розділу 4

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 759521,40 гривень.

Було спрогнозовано орієнтовану величину витрат по кожній зі статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було розроблено інформаційну технологію прогнозування цін на будинки за допомогою методів машинного навчання.

Проаналізовано об’єкт дослідження та визначено актуальність прогнозування цін на будинки за допомогою методів машинного навчання. Розглянуто та описано методи машинного навчання та класифікації. Виявлено, що для ефективного прогнозування цін на будинки необхідно використовувати методи машинного навчання, які дозволяють враховувати різні фактори та залежності.

Виконано огляд відомих аналогів прогнозування цін на приватні житлові будинки. Виявлено що для більш точного прогнозування необхідно вдосконалювати методи прогнозування, залучаючи ширший набір факторів, які впливають на ціну будинків.

Проведено аналіз методів прогнозування, в результаті чого було обрано декілька методів, які найкраще підходять для прогнозування цін на будинки. Розроблено інформаційну технологію прогнозування цін на будинки, показано її граф-схему та алгоритм роботи.

Проведено розвідувальний аналіз даних, розглянута кореляція Спірмана, побудована кореляційна матриця для дослідження зв’язків між числовими та закодованими категоріальними ознаками, що дозволило отримати уявлення про характеристики набору даних та їх взаємозв'язки. В результаті аналізу було виявлено основні ознаки, які найбільше впливають на ціну будинку, наприклад: LotArea, Neighborhood, HouseStyle, OverallCond. Після розвідувального аналізу була проведена обробка вхідних даних для подальшого моделювання та прогнозування цін на будинки. Були видалені відхилення значень та виконані перетворення для покращення набору даних.

Обрано мову програмування Python, оскільки вона надає широкі можливості для реалізації методів машинного навчання.

Проведено моделювання, використовуючи різні методи машинного навчання. Було застосовано моделі ENet, KRR, XGBoost, GBoost, Lasso та LightGBM. За метрикою (RMSE) найбільш точною виявилась модель LightGBM з результатом 0.073. Отже її доцільно використовувати для прогнозування цін на нерухомість. Створена інформаційна технологія, має точність прогнозування на 34 % більшу ніж у подібного аналога.

Загалом, результати дослідження та прогнозування цін на будинки методами машинного навчання свідчать про ефективність такого підходу. Отримані моделі можуть бути використані для прогнозування цін на будинки в майбутньому, що дозволить зробити більш обґрунтовані рішення у сфері нерухомості.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дідик О.С., Паночишин Ю. М. «Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання», в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2024, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/author/submission/22713

2. Що визначає і впливає на ціну нерухомості. URL: https://novebti.ua/ua/blog/chto-opredelyaet-i-vliyaet-na-cenu-nedvizhimosti/

3. Що впливає на ціну будинку. URL: https://www.propertyroad.co.uk/what-affects-the-price-of-a-house/

4. Vijay Kanade, What Is Machine Learning? Definition, Types, Applications, and Trends for 2022. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/

5. What is machine learning? URL: https://www.ibm.com/topics/machine-learning

6. Платформа для пошуку нерухомості Rightmove. URL: https://www.rightmove.co.uk

7. Платформа для пошуку нерухомості Zillow. URL: https://www.zillow.com

8. Український онлайн ресурс для пошуку та оцінки нерухомості Address.ua. URL: https://address.ua

9. Pratik Barua. Kagle Notebook «Prediction», «House Prices - Advanced Regression Techniques». URL: https://www.kaggle.com/code/pratikbarua/house-pricing-predictions

10. Exploratory Data Analysis in Python. URL: https://towardsdatascience.com/exploratory-data-analysis-in-python-a-step-by-step-process-d0dfa6bf94ee

11. Kernel ridge regression. URL: https://scikit-learn.org/stable/modules/kernel_ridge.html

12. Least Absolute Shrinkage and Selection Operator. (LASSO). URL: https://www.publichealth.columbia.edu/research/population-health-methods/least-absolute-shrinkage-and-selection-operator-lasso

13. Gradient Boosting Algorithm: A Complete Guide for Beginners. URL: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

14. The k-Nearest Neighbors (kNN). Algorithm in Python. URL: https://realpython.com/knn-python/

15. Artificial Neural Networks (ANN). What Is an Artificial Neural Network. URL: https://www.coursera.org/articles/artificial-neural-network

16. Recurrent Neural Networks (RNN). What is a recurrent neural network (RNN). URL: https://www.ibm.com/topics/recurrent-neural-networks

17. Anna Montoya, DataCanary. (2016). House Prices - Advanced Regression Techniques. Kaggle. URL: https://kaggle.com/competitions/house-prices-advanced-regression-techniques

18. Cory R. Python Programming for Beginners (1st Edition). Independently published: September 22, 2022. pp. 122.

19. How to Use Kaggle. URL: https://www.kaggle.com/docs/notebooks

20. Бібліотека NumPy, 2020. URL: https://www.w3schools.com/python/numpy/

21. Бібліотека Pandas, 2023. URL: https://pandas.pydata.org/docs/user_guide/index.html

22. Бібліотека Matplotlib, 2023. URL: https://matplotlib.org/stable/tutorials/introductory/quick_start.html

23. Бібліотека Scitik-Learn, 2007-2023. URL: https://scikit-learn.org/stable/getting_started.html

24. XGBoost Documentation, 2022. URL: https://xgboost.readthedocs.io/en/stable/

25. Akyildirim, E., Goncu, A., Sensoy, A.*"Prediction of cryptocurrency returns using machine learning"*. Annals of Operations Research, Springer, vol. 297(1). February, 2021. pages 3-36.

26. Serigne, Kagle Notebook «Stacked Regressions : Top 4% on LeaderBoard». Updated 5 years ago. URL: https://www.kaggle.com/code/serigne/stacked-regressions-top-4-on-leaderboard

27. Dominik Gawlik, Kagle Notebook «House Prices EDA». Updated 6 years ago. URL: https://www.kaggle.com/code/dgawlik/house-prices-eda/notebook

28. Eric Matthes. Python Crash Course: A Hands-On, Project-Based Introduction to Programming: San Francisco: No Starch Press: January 2023, pp. 552.

29. Sun, Ning & Bai, Hongxi & Geng, Yuxia & Shi, Huizhu, "*Price evaluation model in second-hand car system based on BP neural network theory*," IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2017, pp. 431-436.

30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

31. Мокін О.Б., Мокін В.Б. і Мокін Б.І., Алгоритм методу ідентифікації моделі авторегресії – ковзного середнього, який узагальнює методику Юла–Уокера, та його програмна python-реалізація, *Вісник ВПІ*. 2022. № 4. С. 41–55.

32. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних / Я. В. Іванчук, В. І. Месюра, А. А. Яровий, О. Д. Манжілевський – Вінниця : ВНТУ, 2021. – 69 с.

33. Олександр Дідик. Kagle Notebook «Prediction», «House Prices - Advanced Regression Techniques». Last update 30.10.2024. URL: https://www.kaggle.com/code/olexandrdidyk/prediction

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 91,00% Схожість 9,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Дідик О.С.

Керівник роботи _____ Паночишин Ю. М.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток А (обов'язковий)
Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія прогнозування цін на приватні житлові будинки методами машинного навчання

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 91,00% Схожість 9,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Дідик О.С.

Керівник роботи _____ Паночишин Ю. М.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
#color = sns.color_palette()
sns.set_style('darkgrid')
import warnings
def ignore_warn(*args, **kwargs):
   pass
warnings.warn = ignore_warn
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import sklearn.linear_model as linear_model
import xgboost as xgb
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from IPython.display import HTML, display
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
from scipy import stats
from scipy.stats import norm, skew
#pd.set_option('display.float_format', lambda x: '{:.3f}'.format(x))
from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))
train = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')
train_ID = train['Id']
test_ID = test['Id']
train.drop("Id", axis = 1, inplace = True)
test.drop("Id", axis = 1, inplace = True)
quantitative = [f for f in train.columns if train.dtypes[f] != 'object']
quantitative.remove('SalePrice')
#quantitative.remove('Id')
qualitative = [f for f in train.columns if train.dtypes[f] == 'object']
missing = train.isnull().sum()
missing = missing[missing > 0]
missing.sort_values(inplace=True)
missing.plot.bar()
import scipy.stats as st
y = train['SalePrice']

plt.figure(1, figsize=(10, 6))
plt.title('Johnson SU')
sns.distplot(y, kde=False, fit=st.johnsonsu)
```

```
plt.figure(2, figsize=(10, 6))
plt.title('Нормальний розподіл')
sns.distplot(y, kde=False, fit=st.norm)
plt.figure(3, figsize=(10, 6))
plt.title('Логарифмічно нормальний розподіл')
sns.distplot(y, kde=False, fit=st.lognorm)
plt.show()
test_normality = lambda x: stats.shapiro(x.fillna(0))[1] < 0.01
normal = pd.DataFrame(train[quantitative])
normal = normal.apply(test_normality)
print(not normal.any())

f = pd.melt(train, value_vars=quantitative)
g = sns.FacetGrid(f, col="variable", col_wrap=2, sharex=False, sharey=False)
g = g.map(sns.distplot, "value")
for c in qualitative:
    train[c] = train[c].astype('category')
    if train[c].isnull().any():
        train[c] = train[c].cat.add_categories(['MISSING'])
        train[c] = train[c].fillna('MISSING')

def boxplot(x, y, **kwargs):
    sns.boxplot(x=x, y=y)
    x=plt.xticks(rotation=90)
f = pd.melt(train, id_vars=['SalePrice'], value_vars=qualitative)
g = sns.FacetGrid(f, col="variable",  col_wrap=2, sharex=False, sharey=False, size=5)
g = g.map(boxplot, "value", "SalePrice")
def anova(frame):
    anv = pd.DataFrame()
    anv['feature'] = qualitative
    pvals = []
    for c in qualitative:
        samples = []
        for cls in frame[c].unique():
            s = frame[frame[c] == cls]['SalePrice'].values
            samples.append(s)
        pval = stats.f_oneway(*samples)[1]
        pvals.append(pval)
    anv['pval'] = pvals
    return anv.sort_values('pval')

a = anova(train)
a['disparity'] = np.log(1./a['pval'].values)
sns.barplot(data=a, x='feature', y='disparity')
x=plt.xticks(rotation=90)
def encode(frame, feature):
    ordering = pd.DataFrame()
    ordering['val'] = frame[feature].unique()
    ordering.index = ordering.val
    ordering['spmean'] = frame[[feature, 'SalePrice']].groupby(feature).mean()['SalePrice']
    ordering = ordering.sort_values('spmean')
    ordering['ordering'] = range(1, ordering.shape[0]+1)
    ordering = ordering['ordering'].to_dict()
```

```
    for cat, o in ordering.items():
        frame.loc[frame[feature] == cat, feature+'_E'] = o
qual_encoded = []
for q in qualitative:
    encode(train, q)
    qual_encoded.append(q+'_E')
print(qual_encoded)
def spearman(frame, features):
    spr = pd.DataFrame()
    spr['feature'] = features
    spr['spearman'] = [frame[f].corr(frame['SalePrice'], 'spearman') for f in features]
    spr = spr.sort_values('spearman')
    plt.figure(figsize=(6, 0.25*len(features)))
    sns.barplot(data=spr, y='feature', x='spearman', orient='h')

features = quantitative + qual_encoded
spearman(train, features)
plt.figure(1)
corr = train[quantitative+['SalePrice']].corr()
sns.heatmap(corr)
plt.figure(2)
corr = train[qual_encoded+['SalePrice']].corr()
sns.heatmap(corr)
plt.figure(3)
corr = pd.DataFrame(np.zeros([len(quantitative)+1, len(qual_encoded)+1]),
index=quantitative+['SalePrice'], columns=qual_encoded+['SalePrice'])
for q1 in quantitative+['SalePrice']:
    for q2 in qual_encoded+['SalePrice']:
        corr.loc[q1, q2] = train[q1].corr(train[q2])
sns.heatmap(corr)
def pairplot(x, y, **kwargs):
    ax = plt.gca()
    ts = pd.DataFrame({'time': x, 'val': y})
    ts = ts.groupby('time').mean()
    ts.plot(ax=ax)
    plt.xticks(rotation=90)

f = pd.melt(train, id_vars=['SalePrice'], value_vars=quantitative+qual_encoded)
g = sns.FacetGrid(f, col="variable",  col_wrap=2, sharex=False, sharey=False, size=5)
g = g.map(pairplot, "value", "SalePrice")
def encode(frame, feature):
    ordering = pd.DataFrame()
    ordering['val'] = frame[feature].unique()
    ordering.index = ordering.val
    ordering['spmean'] = frame[[feature, 'SalePrice']].groupby(feature).mean()['SalePrice']
    ordering = ordering.sort_values('spmean')
    ordering['ordering'] = range(1, ordering.shape[0]+1)
    ordering = ordering['ordering'].to_dict()

    frame[feature+'_E'] = frame[feature].map(ordering)
    frame.drop([feature+'_E'], axis=1, inplace=True)
```

```
qual_encoded = []
for q in qualitative:
   encode(train, q)
   qual_encoded.append(q+'_E')
print(qual_encoded)

fig, ax1 = plt.subplots()
ax1.scatter(x = train['GrLivArea'], y = train['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()

train = train.drop(train[(train['GrLivArea']>4000) & (train['SalePrice']<300000)].index)

fig, ax1 = plt.subplots()
ax1.scatter(train['GrLivArea'], train['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()
sns.distplot(train['SalePrice'] , fit=norm);

#Отримайте встановлені параметри, що використовуються функцією
(mu, sigma) = norm.fit(train['SalePrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))

#Тепер будуємо розподіл
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
        loc='best')
plt.ylabel('Частота')
plt.title('Розподіл цін:')

#Отримаємо також QQ-графік
fig = plt.figure()
res = stats.probplot(train['SalePrice'], plot=plt)
plt.show()
train["SalePrice"] = np.log1p(train["SalePrice"])

#Перевіряємо новий розподіл
sns.distplot(train['SalePrice'] , fit=norm);

#Отримаємо встановлені параметри, що використовуються функцією
(mu, sigma) = norm.fit(train['SalePrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))

#Будуємо розподіл
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
        loc='best')
plt.ylabel('Частота')
plt.title('Ціновий розподіл')

#Також QQ графік
fig = plt.figure()
res = stats.probplot(train['SalePrice'], plot=plt)
```

```
plt.show()

ntrain = train.shape[0]
ntest = test.shape[0]
y_train = train.SalePrice.values
all_data = pd.concat((train, test)).reset_index(drop=True)
all_data.drop(['SalePrice'], axis=1, inplace=True)
print("Весь розмір даних : {}".format(all_data.shape))

all_data_na = (all_data.isnull().sum() / len(all_data)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)[:30]
missing_data = pd.DataFrame({'Втрачені дані' :all_data_na})
missing_data.head(20)
f, ax = plt.subplots(figsize=(15, 12))
plt.xticks(rotation='90')
sns.barplot(x=all_data_na.index, y=all_data_na)
plt.xlabel('Ознаки', fontsize=15)
plt.ylabel('Відсоток втрачених даних', fontsize=15)
plt.title('Відстоток втрачених даних', fontsize=15)
all_data["PoolQC"] = all_data["PoolQC"].fillna("None")
all_data["MiscFeature"] = all_data["MiscFeature"].fillna("None")
all_data["Alley"] = all_data["Alley"].fillna("None")
all_data["Fence"] = all_data["Fence"].fillna("None")
all_data["FireplaceQu"] = all_data["FireplaceQu"].fillna("None")
all_data["LotFrontage"] = all_data.groupby("Neighborhood")["LotFrontage"].transform(
   lambda x: x.fillna(x.median()))
for col in ('GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'):
   all_data[col] = all_data[col].fillna('None')
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
   all_data[col] = all_data[col].fillna(0)
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
   all_data[col] = all_data[col].fillna(0)
for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2'):
   all_data[col] = all_data[col].fillna('None')
all_data["MasVnrType"] = all_data["MasVnrType"].fillna("None")
all_data["MasVnrArea"] = all_data["MasVnrArea"].fillna(0)
all_data['MSZoning'] = all_data['MSZoning'].fillna(all_data['MSZoning'].mode()[0])
all_data = all_data.drop(['Utilities'], axis=1)
all_data["Functional"] = all_data["Functional"].fillna("Typ")
all_data['Electrical'] = all_data['Electrical'].fillna(all_data['Electrical'].mode()[0])
all_data['KitchenQual'] = all_data['KitchenQual'].fillna(all_data['KitchenQual'].mode()[0])
all_data['Exterior1st'] = all_data['Exterior1st'].fillna(all_data['Exterior1st'].mode()[0])
all_data['Exterior2nd'] = all_data['Exterior2nd'].fillna(all_data['Exterior2nd'].mode()[0])
all_data['SaleType'] = all_data['SaleType'].fillna(all_data['SaleType'].mode()[0])
all_data['MSSubClass'] = all_data['MSSubClass'].fillna("None")
all_data_na = (all_data.isnull().sum() / len(all_data)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Пропущені' :all_data_na})
missing_data.head()
all_data['MSSubClass'] = all_data['MSSubClass'].apply(str)

#Перетворення OverallCond на категоріональну змінну
all_data['OverallCond'] = all_data['OverallCond'].astype(str)
```

```
#Рік і місяць продажу перетворюються на категоріальні ознаки.
all_data['YrSold'] = all_data['YrSold'].astype(str)
all_data['MoSold'] = all_data['MoSold'].astype(str)
from sklearn.preprocessing import LabelEncoder
cols = ('FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual', 'GarageCond',
      'ExterQual', 'ExterCond','HeatingQC', 'PoolQC', 'KitchenQual', 'BsmtFinType1',
      'BsmtFinType2', 'Functional', 'Fence', 'BsmtExposure', 'GarageFinish', 'LandSlope',
      'LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir', 'MSSubClass', 'OverallCond',
      'YrSold', 'MoSold')
# оброблення стовпців, застосовуємо LabelEncoder до категоріональних ознак
for c in cols:
   lbl = LabelEncoder()
   lbl.fit(list(all_data[c].values))
   all_data[c] = lbl.transform(list(all_data[c].values))

print('Сформовані дані: {}'.format(all_data.shape))
all_data['TotalSF'] = all_data['TotalBsmtSF'] + all_data['1stFlrSF'] + all_data['2ndFlrSF']
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index

# Перевірка відхидлень всіх числових ознак
skewed_feats = all_data[numeric_feats].apply(lambda x: skew(x.dropna())).sort_values(ascending=False)
print("\nВикривлення у числових характеристиках: \n")
skewness = pd.DataFrame({'Викривлення' :skewed_feats})
skewness.head(10)
skewness = skewness[abs(skewness) > 0.75]
print("Тут є {} відхилень(ня) числових характеристик до перетворення Кокса-
Бокса".format(skewness.shape[0]))

from scipy.special import boxcox1p
skewed_features = skewness.index
lam = 0.15
for feat in skewed_features:
   all_data[feat] = boxcox1p(all_data[feat], lam)
all_data = pd.get_dummies(all_data)
print(all_data.shape)
train = all_data[:ntrain]
test = all_data[ntrain:]
from sklearn.linear_model import ElasticNet, Lasso,  BayesianRidge, LassoLarsIC
from sklearn.ensemble import RandomForestRegressor,  GradientBoostingRegressor
from sklearn.kernel_ridge import KernelRidge
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.base import BaseEstimator, TransformerMixin, RegressorMixin, clone

from sklearn.metrics import mean_squared_error
import xgboost as xgb
import lightgbm as lgb
n_folds = 5

def rmsle_cv(model):
   kf = KFold(n_folds, shuffle=True, random_state=42).get_n_splits(train.values)
```

```
    rmse= np.sqrt(-cross_val_score(model, train.values, y_train, scoring="neg_mean_squared_error", cv =
kf))
    return(rmse)
lasso = make_pipeline(RobustScaler(), Lasso(alpha =0.0005, random_state=1))
ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005, l1_ratio=.9, random_state=3))
KRR = KernelRidge(alpha=0.6, kernel='polynomial', degree=2, coef0=2.5)
GBoost = GradientBoostingRegressor(n_estimators=3000, learning_rate=0.05,
                                   max_depth=4, max_features='sqrt',
                                   min_samples_leaf=15, min_samples_split=10,
                                   loss='huber', random_state =5)
model_xgb = xgb.XGBRegressor(colsample_bytree=0.4603, gamma=0.0468,
                             learning_rate=0.05, max_depth=3,
                             min_child_weight=1.7817, n_estimators=2200,
                             reg_alpha=0.4640, reg_lambda=0.8571,
                             subsample=0.5213, silent=1,
                             random_state =7, nthread = -1)
model_lgb = lgb.LGBMRegressor(objective='regression',num_leaves=5,
                              learning_rate=0.05, n_estimators=720,
                              max_bin = 55, bagging_fraction = 0.8,
                              bagging_freq = 5, feature_fraction = 0.2319,
                              feature_fraction_seed=9, bagging_seed=9,
                              min_data_in_leaf =6, min_sum_hessian_in_leaf = 11)
score = rmsle_cv(lasso)
print("\nLasso score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
score = rmsle_cv(ENet)
print("ElasticNet score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
score = rmsle_cv(KRR)
print("Kernel Ridge score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
score = rmsle_cv(GBoost)
print("Gradient Boosting score: {:.4f} ({:.4f})\n".format(score.mean(), score.std())
score = rmsle_cv(model_xgb)
print("Xgboost score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
score = rmsle_cv(model_lgb)
print("LGBM score: {:.4f} ({:.4f})\n" .format(score.mean(), score.std()))
class AveragingModels(BaseEstimator, RegressorMixin, TransformerMixin):
    def __init__(self, models):
        self.models = models
    #ми визначаємо клони оригінальних моделей, в які вписуємо дані
    def fit(self, X, y):
        self.models_ = [clone(x) for x in self.models]

        # Навчання клонованої базової моделі
        for model in self.models_:
            model.fit(X, y)
        return self
    #Тепер ми робимо прогнози для клонованих моделей і усереднюємо їх
    def predict(self, X):
        predictions = np.column_stack([
            model.predict(X) for model in self.models_
        ])
        return np.mean(predictions, axis=1)

averaged_models = AveragingModels(models = (ENet, GBoost, KRR, lasso))
```

```
score = rmsle_cv(averaged_models)
print(" Усереднена оцінка базових моделей: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
class StackingAveragedModels(BaseEstimator, RegressorMixin, TransformerMixin):
   def __init__(self, base_models, meta_model, n_folds=7):
      self.base_models = base_models
      self.meta_model = meta_model
      self.n_folds = n_folds
   # Знову підбираємо дані клонів оригінальних моделей
   def fit(self, X, y):
      self.base_models_ = [list() for x in self.base_models]
      self.meta_model_ = clone(self.meta_model)
      kfold = KFold(n_splits=self.n_folds, shuffle=True, random_state=156)
      # Навчаємо клоновані базові моделі, а потім створюємо нестандартні прогнози,
      # які необхідні для навчання клонованої мета-моделі
      out_of_fold_predictions = np.zeros((X.shape[0], len(self.base_models)))
      for i, model in enumerate(self.base_models):
         for train_index, holdout_index in kfold.split(X, y):
            instance = clone(model)
            self.base_models_[i].append(instance)
            instance.fit(X[train_index], y[train_index])
            y_pred = instance.predict(X[holdout_index])
            out_of_fold_predictions[holdout_index, i] = y_pred

      self.meta_model_.fit(out_of_fold_predictions, y)
      return self
   def predict(self, X):
      meta_features = np.column_stack([
         np.column_stack([model.predict(X) for model in base_models]).mean(axis=1)
         for base_models in self.base_models_ ])
      return self.meta_model_.predict(meta_features)
stacked_averaged_models = StackingAveragedModels(base_models = (ENet, GBoost, KRR),
                                                 meta_model = lasso)
score = rmsle_cv(stacked_averaged_models)
print("Усереднений бал моделей: {:.4f} ({:.4f})".format(score.mean(), score.std()))
def rmsle(y, y_pred):
   return np.sqrt(mean_squared_error(y, y_pred))

stacked_averaged_models.fit(train.values, y_train)
stacked_train_pred = stacked_averaged_models.predict(train.values)
stacked_pred = np.expm1(stacked_averaged_models.predict(test.values))
print(rmsle(y_train, stacked_train_pred))
model_xgb.fit(train, y_train)
xgb_train_pred = model_xgb.predict(train)
xgb_pred = np.expm1(model_xgb.predict(test))
print(rmsle(y_train, xgb_train_pred))
model_lgb.fit(train, y_train)
lgb_train_pred = model_lgb.predict(train)
lgb_pred = np.expm1(model_lgb.predict(test.values))
print(rmsle(y_train, lgb_train_pred))
sub = pd.DataFrame()
sub['Id'] = test_ID
sub['SalePrice'] = ensemble
sub.to_csv('submission.csv',index=False)
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ ЦІН НА ПРИВАТНІ ЖИТЛОВІ БУДИНКИ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Виконав: студент 2-го курсу,
групи 2КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Дідик О.С.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Паночишин Ю.М.
(прізвище та ініціали)
«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

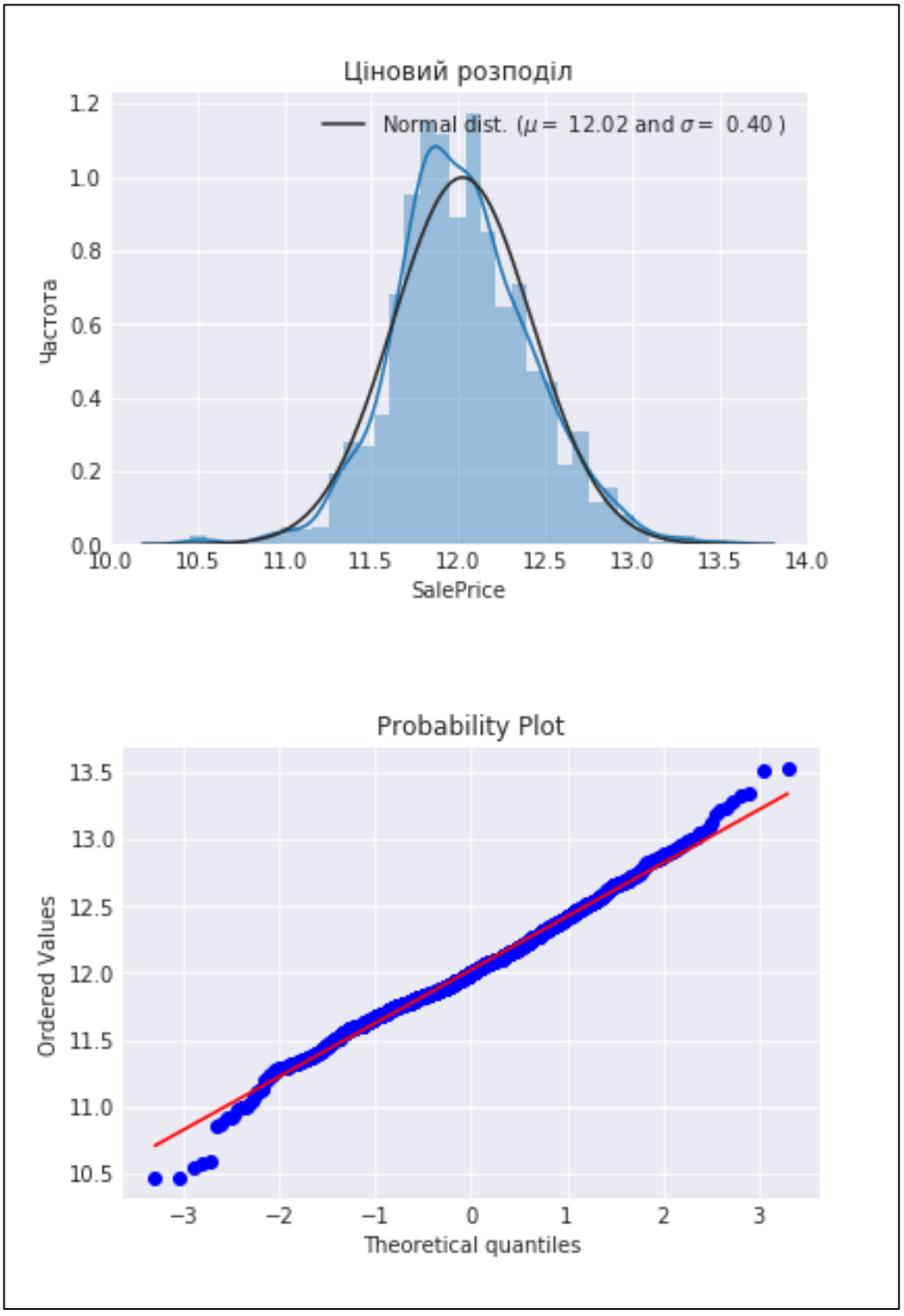

Рисунок В.1 – Розподіл цін та цільова змінна

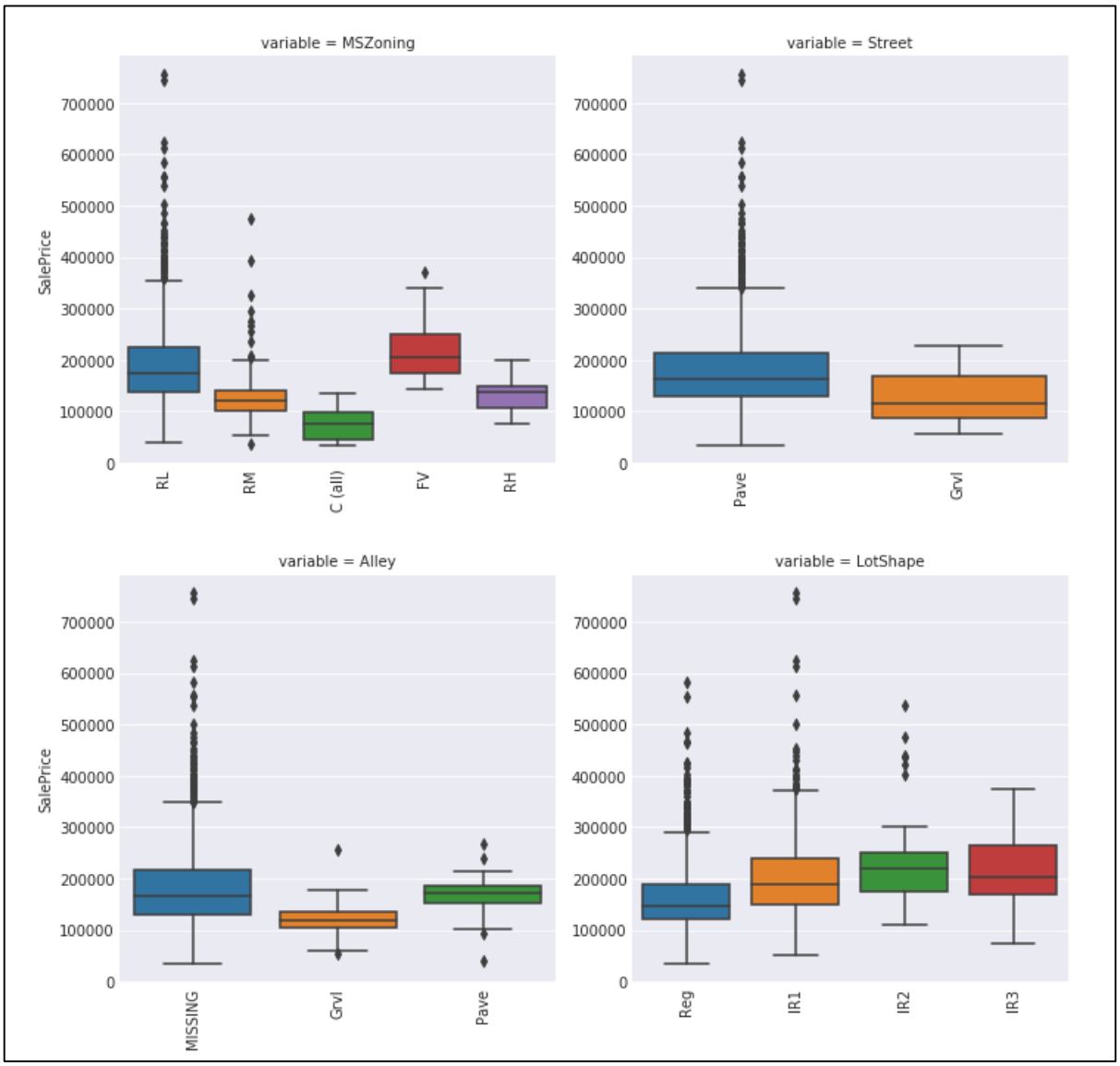

Рисунок В.2 – Графіки залежності категоріальних змінних

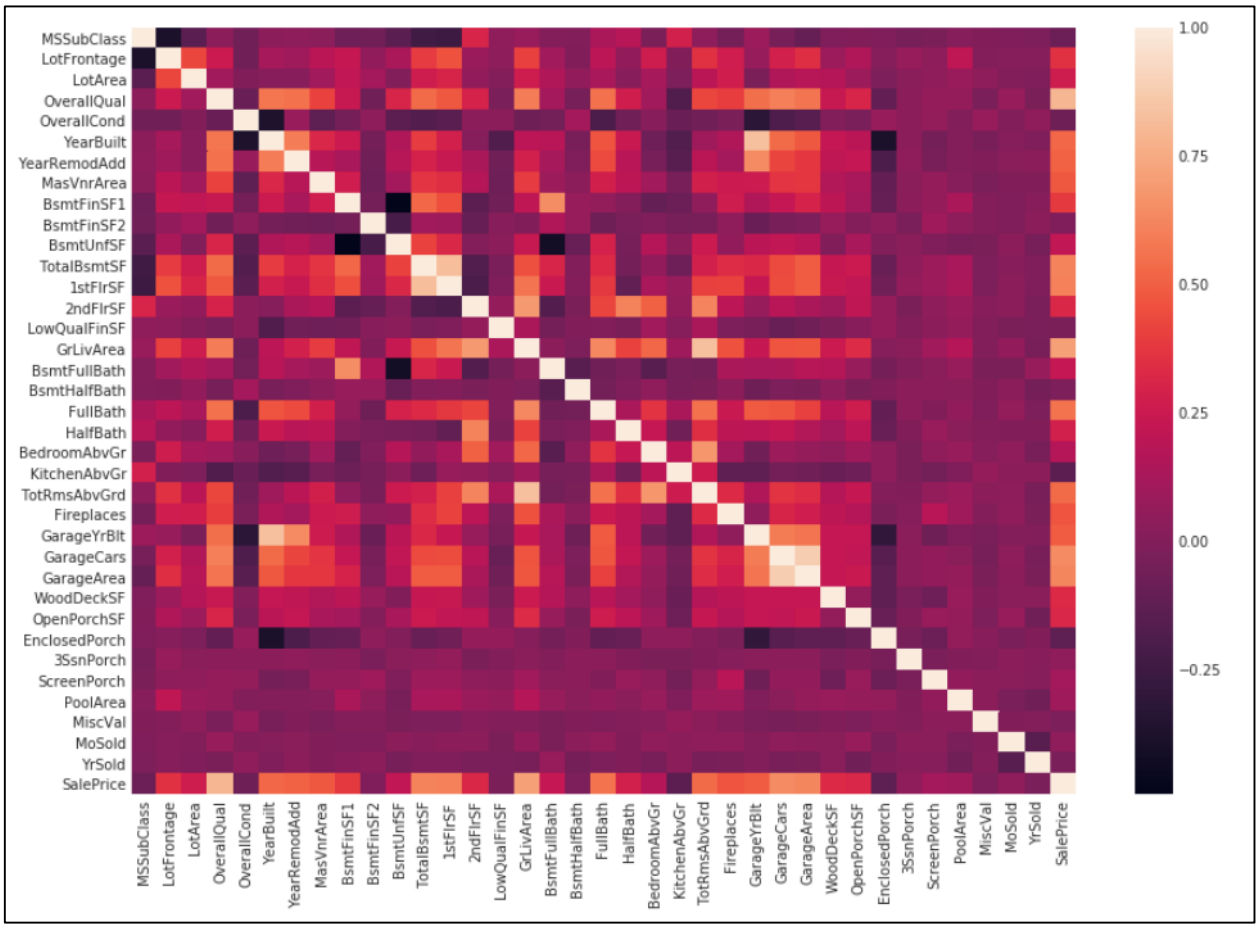

Рисунок В.3 – Кореляційна матриця

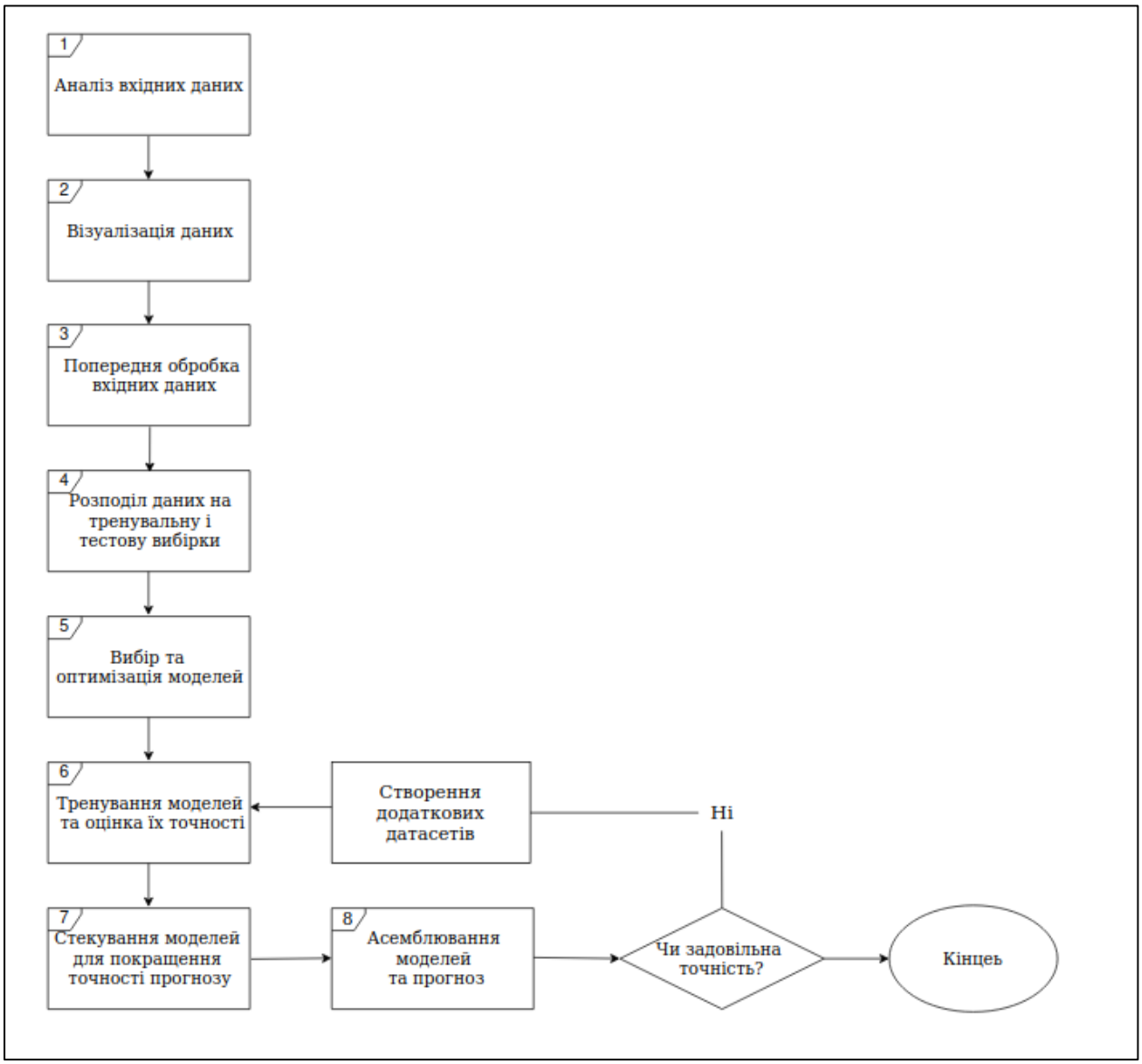

Рисунок В.4 – Граф-схема інформаційної технології

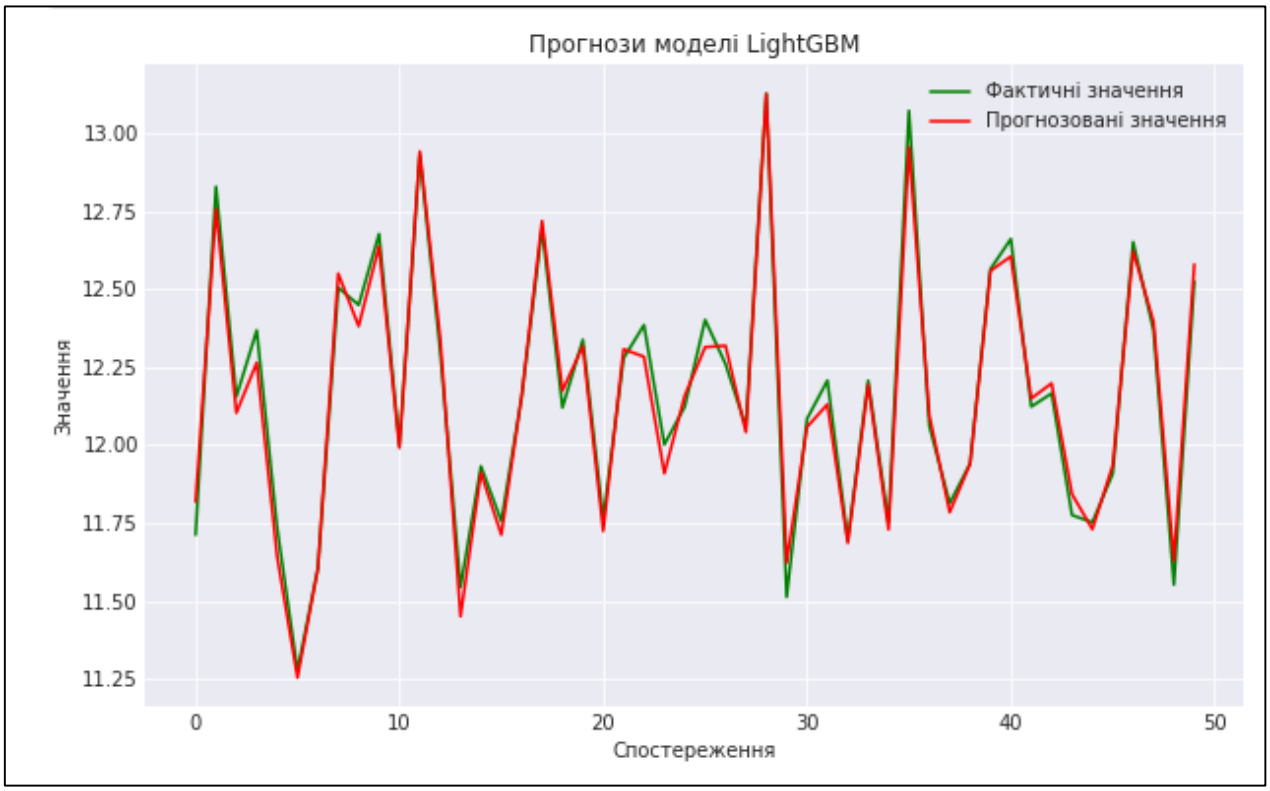

Рисунок В5 – Прогноз моделі LightGBM

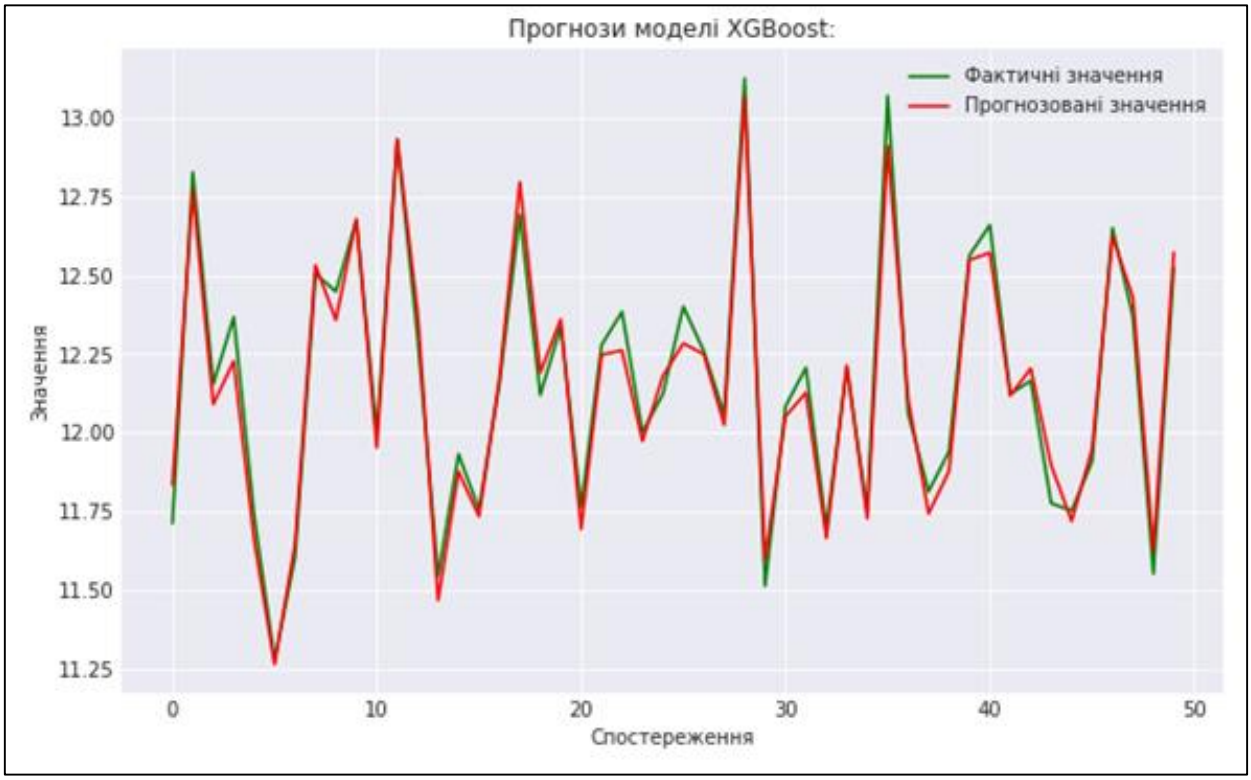

Рисунок В.6 – Прогноз моделі XGBoost

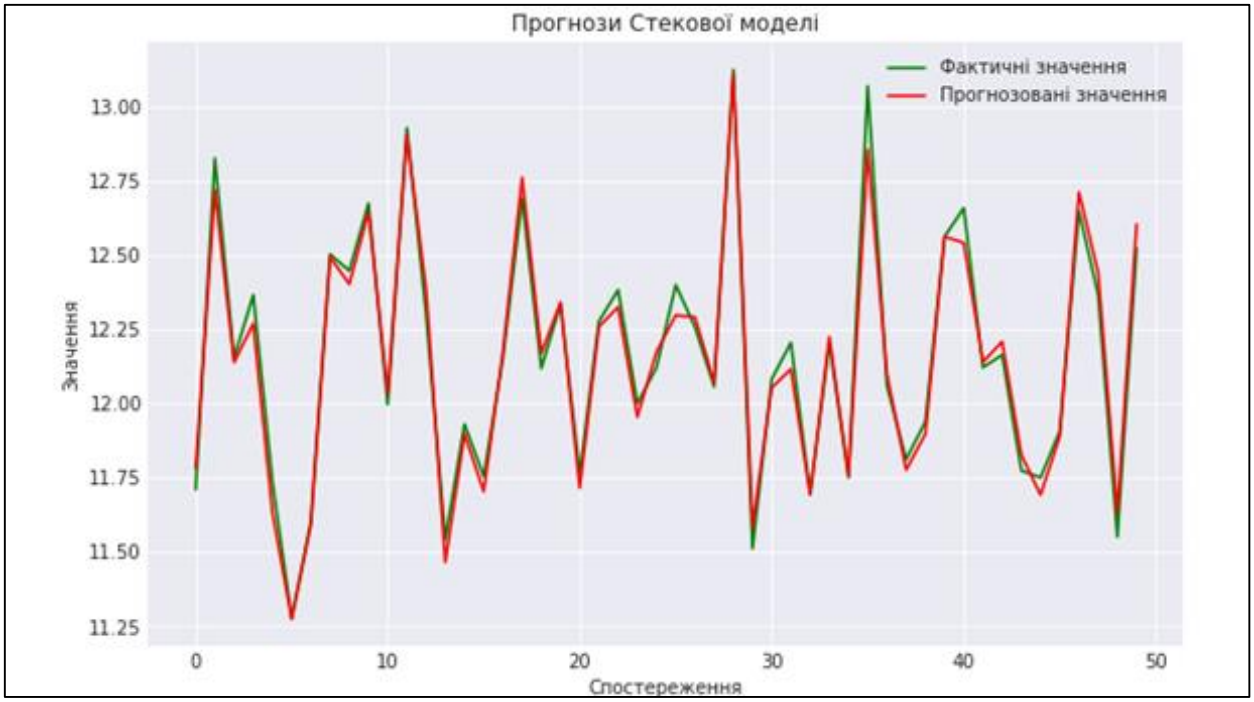

Рисунок В.7 – Прогноз моделі StackedRegressor