

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

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«____» _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Системи та методи штучного
інтелекту»**

**спеціальності 122 «Комп'ютерні науки та інформаційні
технології»**

на тему: «Система автоматичного оцінювання нерухомого майна»

Виконав :

студент IV курсу, групи КА-65

Іванов Даніїл Сергійович

Керівник:

доцент Дідковська М.В

Консультант з економічної частини:

доцент, к.е.н. Шевчук О.А

Консультант з нормоконтролю:

доцент, к.т.н. Коваленко А.Є.

Рецензент:

доцент, к.т.н. Заболотня Т. М.

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

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп’ютерні науки та інформаційні технології»

Освітньо-професійна програма «Системи штучного інтелекту»
ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«25» травня 2020 р.

ЗАВДАННЯ

на дипломну роботу студенту
Іванову Даніїлу Сергійовичу

1. Тема роботи «Система автоматичного оцінювання нерухомого майна», керівник роботи Дідковська Марина Віталіївна, професор, д.т.н., затверджені наказом по університету від «25» травня 2020 р. № 1143-с
2. Термін подання студентом роботи 8.06.2020.
3. Вихідні дані до роботи програмне забезпечення для автоматичного оцінювання вартості нерухомого майна.
4. Зміст роботи аналіз існуючих методів для розв'язку задачі оцінки вартості нерухомого майна, вибір оптимальних методів, реалізація процесу навчання моделі,
реалізація програмного продукту для взаємодії з моделлю, проведення аналізу роботи побудованої системи.
5. Перелік ілюстративного матеріалу зображення існуючих рішень, зображення графіків та діаграм, які ілюструють методи розв'язання задачі регресії та роботу моделі, приклади роботи програмного продукту, презентація до виступу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв
Економічний	Шевчук О. А., к.е.н., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формування теми БДР	07.09.2019	Виконано
2	Огляд літератури за тематикою роботи та її опрацювання	07.09.2019-01.03.2020	Виконано
3	Програмна реалізація моделі	11.03.2020-10.04.2020	Виконано
4	Програмна реалізація взаємодії з моделлю	10.04.2020-18.04.2020	Виконано
5	Написання першого розділу БДР	18.04.2020-27.04.2020	Виконано
6	Узгодження теми БДР з керівником	28.04.2020	Виконано
7	Написання другого розділу БДР	01.05.2020-07.05.2020	Виконано
8	Написання третього розділу БДР	14.05.2020-27.05.2020	Виконано
9	Написання четвертого розділу БДР	27.05.2020-29.05.2020	Виконано

Студент

Іванов Д.С.

Керівник

Дідковська М.В

РЕФЕРАТ

Дипломна робота: 93 с., 6 табл., 33 рис., 3 дод., 7 джерел.

РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОГО ОЦІНЮВАННЯ НЕРУХОМОГО МАЙНА

Тема - система автоматичного оцінювання нерухомого майна.

Мета роботи – розглянути теоретичні основи прогнозування вартості нерухомого майна, підготувати базу даних для подальшого навчання системи, розробити систему для оцінювання нерухомості, використовуючи сучасні методи машинного навчання.

У даній роботі зібрано власну базу даних методом веб-скрейпингу. Розглянуто існуючі методи прогнозування вартості нерухомого майна та розроблено власну систему оцінювання квартир в місті Києві.

Результатом даної роботи є дієздатний програмний продукт у вигляді прикладного програмного інтерфейсу, який може бути інтегрованим в будь-якому іншому веб-ресурсі.

Новизною роботи є метод інтерпретації даних, який використовує набір таких характеристик, як азимут, відстань від центра міста та від найближчої станції метрополітену, для визначення рівня престижності району, в якому знаходиться квартира, а також створення системи, здатної бути підключеною до будь-якого веб-сайту або мобільного додатку.

ABSTRACT

Thesis: 93 p., 6 tabl., 33 fig., 3 appendices, 7 sources.

DEVELOPMENT OF SYSTEM FOR AUTOMATIC REAL ESTATE VALUATION

Theme: system for automatic valuation of real estate.

The purpose of the work is to consider the theoretical foundations of real estate value forecasting, to prepare a database for further training of the system, to develop a system for valuation of real estate using modern methods of machine learning.

This paper has its own database, that was compiled by web scraping. The existing methods of real estate value forecasting are considered and the own system of apartment valuation in Kyiv is developed.

The result of this work is a viable software product, that was created as an application program interface that can be integrated in any other web resource.

The novelty of this work is a method of data interpretation, which uses a set of characteristics such as azimuth, distance from the city center and the nearest subway station, to determine the level of prestige of the area in which the apartment is located, and also creation of a system capable to be integrated in any web-site or mobile application.

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ ОЦІНКИ ВАРТОСТІ НЕРУХОМОГО МАЙНА	7
1.1 Аналіз актуальності задачі оцінки вартості нерухомого майна	7
1.2 Аналіз існуючих рішень	8
1.2.1 Алгоритм компанії Zillow	8
1.2.2 Онлайн-калькулятор від компанії DOM.RIA	9
1.3 Постановка задачі дослідження	11
1.4 Формування бази даних	11
1.5 Висновки за розділом 1	14
РОЗДІЛ 2 МЕТОДИ ОЦІНЮВАННЯ ВАРТОСТІ НЕРУХОМОГО МАЙНА	15
2.1 Аналіз математичних основ задачі оцінювання вартості нерухомого майна	15
2.1.1 Кореляція	15
2.1.2 Одновимірний регресійний аналіз	17
2.1.3 Множинний регресійний аналіз	20
2.2 Методи вирішення задачі оцінки вартості нерухомого майна	22
2.2.1 Дерева рішень	22
2.2.2 Випадковий ліс	24
2.2.3 Градієнтний бустинг	26
2.3 Критерії якості роботи моделі	27
2.4 Порівняльний аналіз існуючих методів розв'язку задачі регресії	28

2.5 Алгоритм.....	30
2.5.1 Інтерпретація даних.....	30
2.5.2 Вибір методу машинного навчання.....	32
2.6 Висновки за розділом 2	33
РОЗДІЛ 3 АНАЛІЗ АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ ТА ПРАКТИЧНИХ РЕЗУЛЬТАТІВ	34
3.1 Обґрунтування вибору платформи та мови програмування	34
3.1.1 Збір бази даних	34
3.1.2 Навчання моделей	35
3.1.3 Кінцевий програмний продукт	37
3.2 Керівництво користувача кінцевим програмним продуктом	37
3.2.1 Браузерна версія	37
3.2.2 Взаємодія з API інших веб-сервісів.....	45
3.3 Аналіз роботи системи з оцінки нерухомості	46
3.3.1 Аналіз якості роботи системи	46
3.3.2 Аналіз роботи програмного продукту	47
3.4 Висновки за розділом 3	48
РОЗДІЛ 4. ФУНКЦІОНАЛЬНО-ВАРТИСНИЙ АНАЛІЗ.....	49
4.1 Постановка задачі проектування	49
4.2 Обґрунтування функцій та параметрів програмного продукту	49
4.3 Економічний аналіз варіантів розробки	56
4.4 Вибір кращого варіанта ПП техніко-економічного рівня	60
4.5 Висновки до розділу 4	61

ВИСНОВКИ	62
ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ	63
ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ.....	64
ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ ДЛЯ ТРЕНАУВАННЯ МОДЕЛІ	75
ДОДАТОК В ЛІСТИНГ КІНЦЕВОГО ПРОГРАМНОГО ПРОДУКТУ	82

ВСТУП

Останнім часом методи машинного навчання використовують для розв'язку різноманітних задач. Однією з галузей, де сучасні підходи впроваджуються в реальне життя, є ринок нерухомості.

В такій нестабільній економічній ситуації як у 2020 році, люди намагаються зберегти свої заощадження від знецінення. Одним з найпопулярніших способів зробити це — інвестувати гроші у нерухомість.

Оцінка вартості майна є першочерговою та однією з найважливіших задач в галузі нерухомості. Аналітики, ріелтори та звичайні люди витрачають дуже багато часу на аналіз ринку для того, щоб правильно оцінити майно, яке вони збираються продати.

Саме тому темою цього дослідження було обрано створення програмного продукту для автоматичної оцінки вартості нерухомого майна за допомогою методів машинного навчання.

Такий сучасний підхід допомагає заощадити багато часу, та зробити оцінку набагато точнішою.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ ОЦІНКИ ВАРТОСТІ НЕРУХОМОГО МАЙНА

1.1 Аналіз актуальності задачі оцінки вартості нерухомого майна

Інвестиції в ринок нерухомості вважаються одним з найнадійніших способів розміщення коштів. Невисока прибутковість операцій з нерухомістю компенсується їх низьким ризиком.

В останні два роки ситуація на ринку нерухомості була нестійкою. У 2019 році через досить значне зміцнення національної валюти попит на житло, а особливо первинне, різко впав. Це сталося насамперед через те, що забудовники на початку року рахували ціни по іншому курсу. Тому людям із доларовими заощадженнями невигідно було купляти житло на первинному ринку, де операції здійснюються виключно у національній валюті. Таким чином багато потенційних покупців обрали очікувальну позицію, створивши відкладений попит.

Сьогодні, у 2020 році, попит на нерухомість знову зростає. Люди намагаються зберегти свої кошти та отримати від них прибутки.

Однією з найскладніших задач спеціаліста в галузі нерухомості є задача з оцінки майна. У великих компаніях зазвичай є свій власний окремий аналітичний відділ, який і виконує цю задачу. У них є своя власна база даних, яка складається з тисяч вже проданих об'єктів. Вони використовують різні алгоритми, в тому числі й методи машинного навчання. Але, на жаль, ці програмні продукти найчастіше закриті для інших компаній, приватних спеціалістів та звичайних людей.

Маленькі компанії, на відміну від компаній-гіантів, зазвичай не мають таких ресурсів. У них нема окремого аналітичного відділу, який допомагає швидко оцінювати об'єкти. Тому ріелтори змушені виконувати цю роботу самотужки. Це займає дуже багато часу, якщо сумлінно виконувати цю задачу.

Оцінити майно вручну на основі багатьох тисяч інших об'єктів можна лише неякісно.

Отже, задача оцінки вартості нерухомого майна є дуже актуальну.

1.2 Аналіз існуючих рішень

Зважаючи на те, що задача оцінки вартості нерухомого майна не є новою, у світі є низка готових рішень. Нижче наведені приклади таких рішень.

1.2.1 Алгоритм компанії Zillow

Zillow[1] - це дуже популярна американська онлайн-платформа нерухомості, яка працює в США та Канаді.

Дім часто є найбільшою і найдорожчою покупкою, яку людина робить за своє життя. Забезпечення власників будинків надійним способом контролю за цим активом надзвичайно важливо. Zestimate, програмний продукт компанії Zillow, був створений для надання споживачам якомога більше інформації про ринок житла та вартість їх будинків.

Zestimate - це модель оцінки вартості будинків на основі 7,5 мільйонів статистичних та моделей машинного навчання, яка аналізує сотні характеристик про кожну будівлю. Zillow постійно зменшує похибку своєї моделі. У 2017 році їхня медіанна абсолютна процентна помилка становила 5%, що є дуже гарним результатом. Але вони на цьому не зупинились і організували змагання за головний приз в один мільйон доларів. Метою цих змагань було удосконалення алгоритму. На сьогоднішній день компанія Zillow може похвалитися своєю медіанною абсолютною процентною помилкою, яка приблизно дорівнює 2%.

Отже, оцінювальна модель даної компанії є однією з найкращих у світі. Та все ж таки це не означає, що задача побудови алгоритмів для українського ринку нерухомості не є актуальною. По-перше, завжди можливо щось зробити краще, і по-друге, ця компанія працює суперечконо на американському та канадському ринках нерухомості.

1.2.2 Онлайн-калькулятор від компанії DOM.RIA

Калькулятор вартості нерухомості від української компанії DOM.RIA[2] - це сервіс від Аналітичного центру DOM.RIA, за допомогою якого можна дізнатися приблизну ринкову вартість квартири. Для цього необхідно вказати її основні параметри: кількість кімнат, загальна площа, область, місто і район, де розташована квартира. Те, як виглядає даний онлайн-калькулятор, зображене на рисунку 1.1.

Як це працює

Аналітичний центр DOM.RIA проводить розрахунок вартості, виходячи з наявних даних про аналогічні об'єкти в вашому районі.

Виберіть місцезнаходження → Вкажіть параметри → Розрахуйте середню вартість

Розрахуйте вартість своєї квартири

В області *	Виберіть область
В місті *	Виберіть місто
В районі	Виберіть район
Кількість кімнат *	1 2 3 4 5+
Загальна площа, м ² *	Загальна площа, м ²

Розрахувати вартість

ПЕРЕВІРЕНІ КВАРТИРИ

Рисунок 1.1 – Калькулятор вартості нерухомості від компанії DOM.RIA

За кілька секунд користувач отримує орієнтовну вартість об'єкта, що його цікавить. Результат роботи калькулятора зображенено на рисунку 1.2.

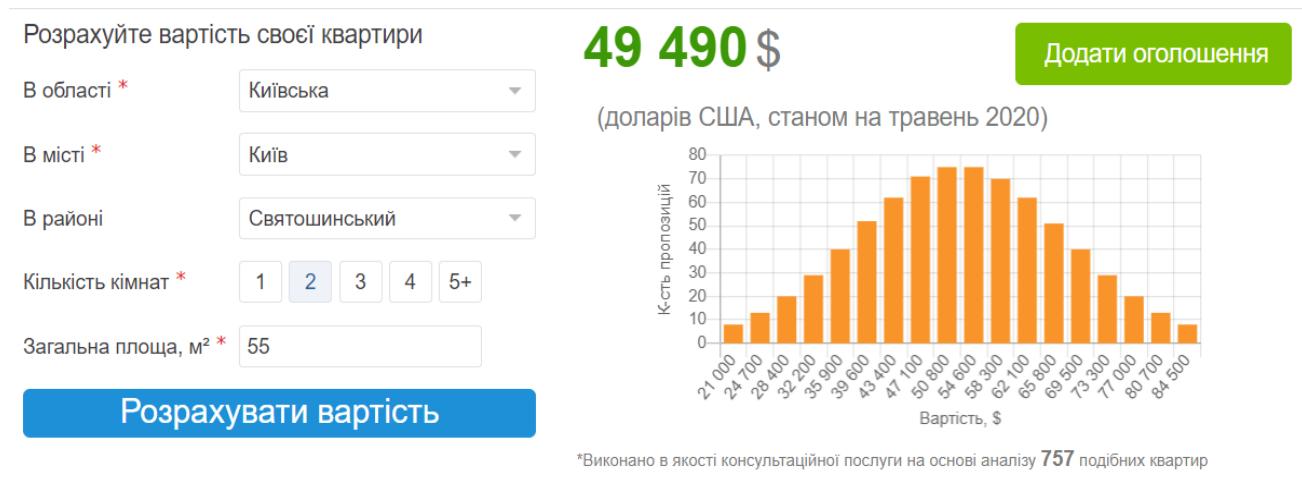


Рисунок 1.2 – Результат роботи калькулятора від компанії DOM.RIA

*Виконано в якості консультаційної послуги на основі аналізу 757 подібних квартир

Даний калькулятор використовує дуже малу кількість характеристик, а тому не може претендувати на велику точність. Отже, він може використовуватися, наприклад, фізичними особами, але не професіоналами в галузі нерухомості.

1.3 Постановка задачі дослідження.

1. Збір достатньої для роботи бази даних.
2. Аналіз математичних основ.
3. Побудова регресійних моделей.
4. Аналіз отриманих результатів.
5. Оцінка та порівняння моделей.
6. Розробка кінцевого програмного продукту.

1.4 Формування бази даних

Було вирішено виконати дане дослідження на прикладі оцінки вартості квартир в місті Києві.

Джерелом даних для роботи над створенням автоматичної системи оцінювання нерухомого майна був выбраний відомий інтернет ресурс DOM.RIA з найбільшою кількістю оголошень на момент збору даних. Крім того, перевагою даного сайту є наявність в нього задокументованого прикладного програмного інтерфейсу, або API. Спочатку, планувалось з його допомогою отримати дані по всім тридцяти п'яти тисячам об'єктів, але це не вдалося здійснити через обмеження, які накладаються на користувачів безкоштовною версією API.

Але все ж таки деякі можливості цього прикладного програмного інтерфейсу було використано, а саме: було отримано список URL-адрес

оголошень з продажу квартир в Києві та список параметрів, якими описуються ці об'єкти.

Після цього було написано програму, яка автоматично збирала інформацію про кожну квартиру зі списку оголошень по заданим параметрам. Отримана інформація записувалась в файл в форматі csv. Після закінчення роботи програми в файлі було близько тридцяти тисяч об'єктів.

Спеціалісти з машинного навчання часто кажуть, що більшу частину їхньої роботи складає збір даних, їх обробка та підготовка до навчання. Так і в даній роботі було затрачено чимало часу та зусиль саме на підготовку бази даних.

По-перше, задача збору такого об'єму інформації вимагає досить багато часу. У середньому, кожна адреса оброблялася за 2 секунди. Тобто одній програмі знадобилось би більше сімнадцяти годин безперервної роботи, щоб зібрати всю необхідну інформацію. Саме тому було зроблено декілька копій коду і запущено їх одночасно, що набагато скоротило час, необхідний для збору бази даних.

По-друге, зібрану інформацію необхідно було обробити і підготувати для подальшого використання. Було проведено базову підготовку: переведено всі ціни в долари США, всі ціни до одного типу – ціна за об'єкт, видалено об'єкти, які були опубліковані раніше ніж перше травня 2020 року.

Але найскладнішою і найважливішою задачею було видалення викидів, неактуальної інформації, помилок. Інакше, якщо не позбутися цього і спробувати навчати модель на таких неякісних даних, то модель не буде відрізнятися точністю передбачень.

Одне з головних джерел цих викидів – помилки користувачів, які створювали оголошення. Так, наприклад, дуже часто буває, що користувачі вказують ціну квартири в доларах, хоча мають на увазі гривні. Квадратний метр такої квартири може коштувати майже двісті тисяч доларів, як показано на рисунку 1.3.

The screenshot shows a real estate listing for a 3-room apartment (3к) at 116 square meters on Dragomirova Street, 14A, located near the Druzhba narodov metro station in the Pechersky district of Kiev. The listing price is listed as 22,214,000 \$. Below the price, it says '600 378 444 грн • 191 500 \$ за м² курс валют'. To the right of the text is a thumbnail image of a modern building under a blue sky with white clouds. A watermark for 'dom RIA' is visible in the top right corner of the image.

Рисунок 1.3 – Приклад помилки в оголошенні: занадто велика ціна

Також досить часто зустрічалися такі помилки: занадто велика площа квартири, як показано на рисунку 1.4, площа кухні, більша ніж загальна площа квартири, як показано на рисунку 1.5.

The screenshot shows a listing for an apartment in JKK Pokrovsky Posad. It features a grid of small thumbnail images of the interior rooms, followed by a link 'Смотреть все 19 фотографий' (View all 19 photos). Below the photos are two category buttons: 'Торг' (Trade) and 'Вторичное жилье' (Secondary housing). The main title is 'Продажа квартиры в ЖК Покровский посад'. Below the title, it says '3 комнаты • 15 этаж из 23' (3 rooms • 15th floor of 23) and 'Площадь 149000 м² • 56 м² • 46 м²' (Area 149000 m² • 56 m² • 46 m²). The listing appears to be a duplicate or a very large room.

Рисунок 1.4 – Приклад помилки в оголошенні: занадто велика площа квартири

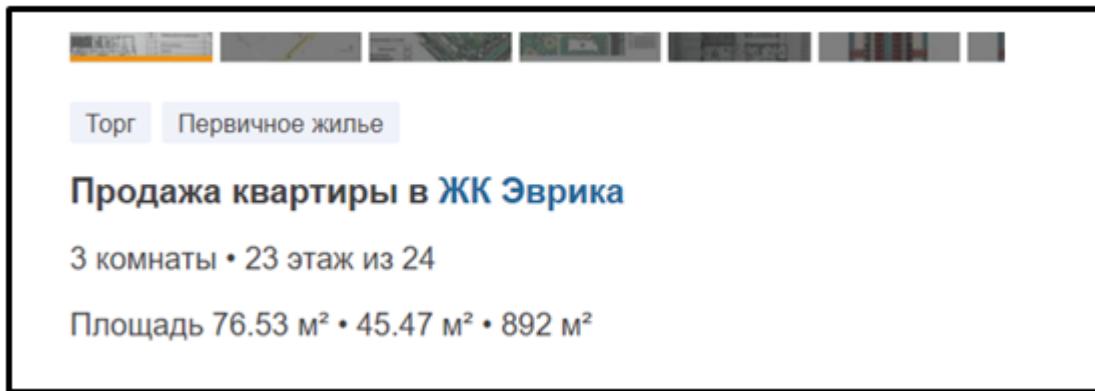


Рисунок 1.5 – Приклад помилки в оголошенні: площа кухні більша, ніж загальна площа

Крім цього, велика кількість оголошень мали неправильні координати розташування.

В результаті, після первинної обробки бази даних було відібрано 22952 об'єкта для подальшого використання.

Крім основної бази даних був зібраний список координат усіх 52-х станцій київського метрополітену.

1.5 Висновки за розділом 1

У даному розділі була розглянута актуальність задачі оцінки вартості нерухомого майна. Цей етап полягав у проведенні аналізу попиту на нерухомість в Україні, існуючих рішень на українському та іноземних ринках нерухомості.

Також була проведена формалізація постановки задачі та опис процесу формування бази даних.

РОЗДІЛ 2 МЕТОДИ ОЦІНЮВАННЯ ВАРТОСТІ НЕРУХОМОГО МАЙНА

2.1 Аналіз математичних основ задачі оцінювання вартості нерухомого майна

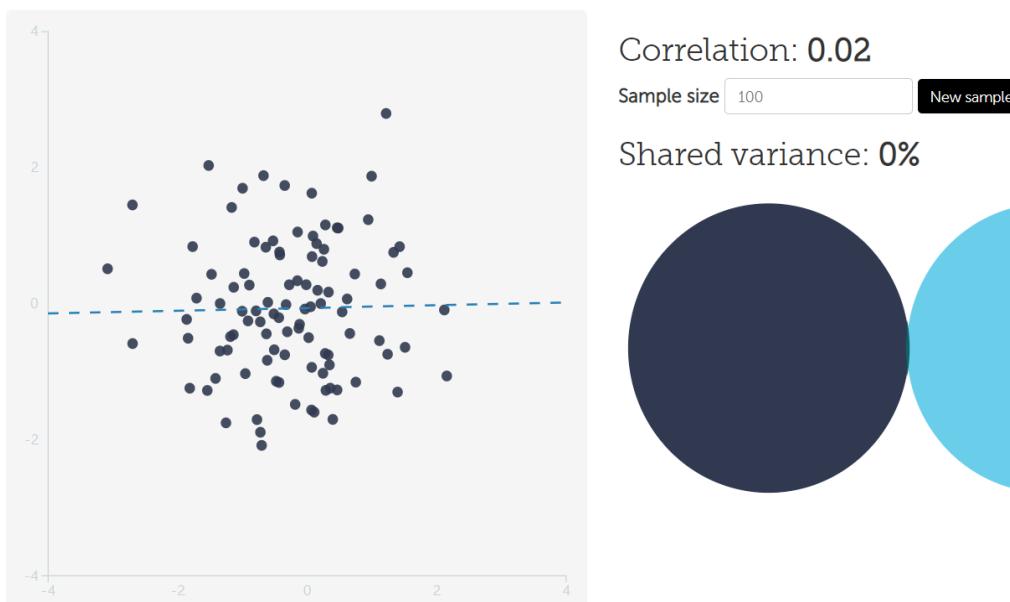
2.1.1 Кореляція

Кореляцією називають статистичний взаємозв'язок довільної кількості випадкових величин. Кореляція буває додатною та від'ємною. Кореляція додатна у випадку, якщо спостерігається прямо пропорційний зв'язок, тобто зі збільшенням однієї змінної, збільшується й інша змінна. Якщо ж при збільшенні однієї змінної, інша зменшується, то кореляція називається від'ємною.

Коефіцієнт кореляції r_{xy} описує взаємозв'язок двох кількісних змінних X та Y . Він може приймати значення від -1 до 1 включно. Від'ємний коефіцієнт кореляції відповідає від'ємній кореляції. Додатний коефіцієнт кореляції відповідає додатній кореляції. Якщо значення коефіцієнту кореляції приблизно дорівнює нулю, то кажуть, що з великою ймовірністю величини між собою ніяк не зв'язані.

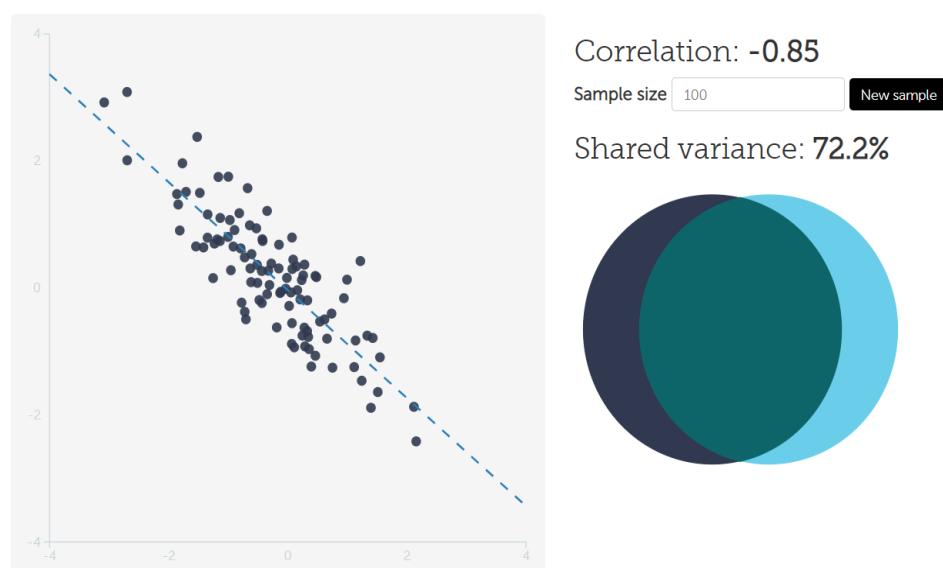
Для того, щоб дослідити взаємозв'язок двох кількісних змінних, зазвичай перш за все будують діаграму розсіювання. Такий вид графіків дають змогу візуально оцінити даний взаємозв'язок. Інколи вдається з досить тільки за допомогою такого графіка прийти до висновку, що змінні між собою не пов'язані. Наприклад, з рисунку 2.1 відображено дві кількісні величини з коефіцієнтом кореляції, близьким до нуля. Можна побачити, що точки на декартовій площині знаходяться майже горизонтально. Отже, скоріше за все зміна однієї змінної ніяк не пояснюється впливом іншої.

Slide me

Рисунок 2.1 – Діаграма розсіювання з $r_{xy}=0.02$

На рисунку 2.1 зображене лінійний взаємозв'язок двох кількісних величин з коефіцієнтом $r_{xy}=-0.85$.

Slide me

Рисунок 2.2 – Діаграма розсіювання з $r_{xy}=-0.85$

А на рисунку 2.3 можна побачити явно виражену додатну кореляцію.

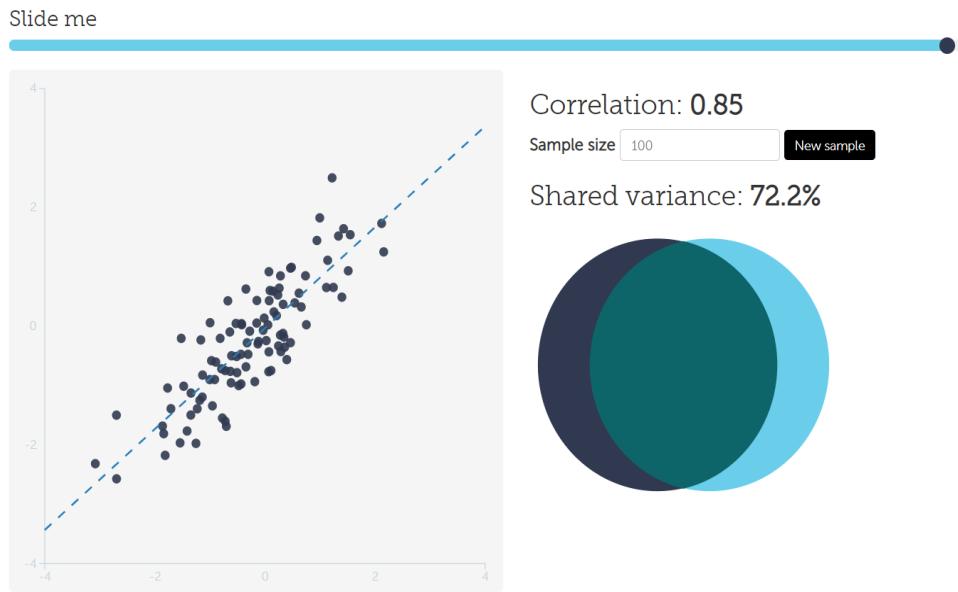


Рисунок 2.3 – Діаграма розсіювання з $r_{xy}=0.85$

Для оцінки взаємозв'язку випадкових величин, крім такого статистичного показника, як коефіцієнт кореляції, використовують також коефіцієнт детермінації.

Коефіцієнт детермінації R^2 (R – квадрат) показує в якій степені дисперсія однієї змінної обумовлена впливом іншою змінної. R^2 дорівнює квадрату коефіцієнта кореляції. R^2 приймає значення від 0 до 1 включно.

Коли ми намагаємося передбачати значення змінної, коефіцієнт детермінації допомагає нам оцінити наскільки добре в нас це виходить. Чим більше R^2 до одиниці, тим більше наші передбачені значення до реальних.

2.1.2 Одновимірний регресійний аналіз

Одновимірний регресійний аналіз дозволяє перевіряти гіпотези про взаємозв'язок однієї кількісної залежності змінної та однієї незалежної.

Прийнято позначати залежну змінну як Y , а незалежну змінну як X . В загальному випадку X – це вектор незалежних змінних. Якщо ж X – одна незалежна змінна, то таку регресію називають простою, як на рисунку 2.4.

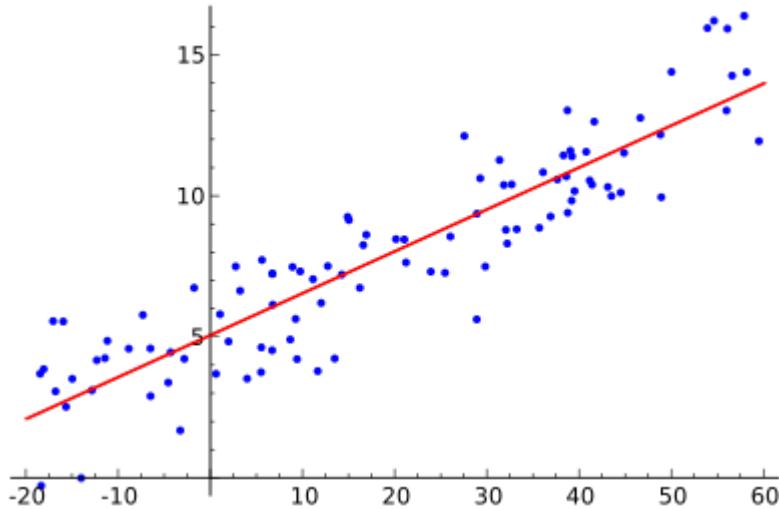


Рисунок 2.4 – Проста лінійна регресія

Лінія регресії – це така пряма, яка найкраще описує взаємозв'язок досліджуваних величин. В простому випадку вона виглядає так:

$$Y = b_0 + b_1 X \quad (2.1)$$

де Y – залежна змінна (наприклад, ціна квартири);

X – незалежна змінна (наприклад, площа квартири);

b_0 та b_1 – коефіцієнти лінійної регресії, які потрібно знайти, для того, щоб побудувати лінію регресії, яка дійсно буде найкраще описувати взаємозв'язок величин.

Наприклад, на рисунку 2.5. добре видно, що з цією задачею краще справляється лінія синього кольору.

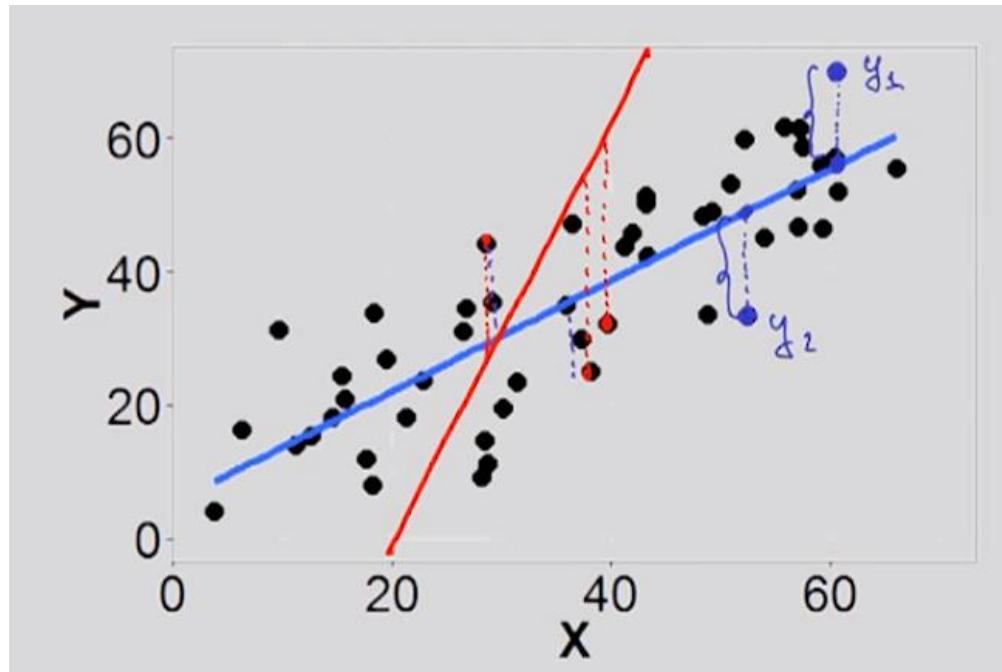


Рисунок 2.5

Щоб пояснити математичною мовою чому саме вона краще описує цей зв'язок, потрібно розглянути таке поняття, як сума квадратів похибок (залишків).

Похибка - це відстань від реальної координати точки до регресійної прямої. Ці похибки можуть буди додатними та від'ємними, тому для того, щоб позбавитися від впливу від'ємних доданків ми розглядаємо суму саме квадратів цих похибок. Чим менше цей параметр, тим точніше побудована лінія регресії.

Отже, можна зробити висновок, що синя лінія на рис. 2.5 краще описує лінійний взаємозв'язок змінних X та Y через те, що має меншу суму квадратів похибок.

Для того, щоб сума квадратів похибок була найменшою, потрібно правильно визначити коефіцієнти b_0 та b_1 . Для цього зазвичай використовують метод найменших квадратів (МНК).

Умови використання лінійної регресії з одним незалежною змінною:

- Лінійна залежність змінних;
- Нормальний розподіл залишків;

- Гомоскедастичність – постійна мінливість залишків на всіх рівнях незалежної змінної;

2.1.3 Множинний регресійний аналіз

Множинний регресійний аналіз дозволяє досліджувати вплив одразу декількох незалежних величин на одну залежну.

В множинній регресії замість ліній регресії використовується площа регресії. Окрім цього з'являються додаткова вимога до даних, які ми збираємося аналізувати, а саме: відсутність мультиколінеарності – дуже сильний взаємозв'язок між якимись із незалежних змінних. Також бажано, щоб розподіл змінних був нормальним.

Для того, щоб модель множинної регресії була ефективною необхідно вибрати які змінні використовувати, а які краще виключити з неї. Для цього є два основних варіанти:

1. Перевірка на мультиколінеарність: для цього можна побудувати діаграми розсіювання для кожної пари незалежних змінних, які надають інформацію про кореляцію між ними. Приклад такого набору діаграм зображено на рис. 2.6.

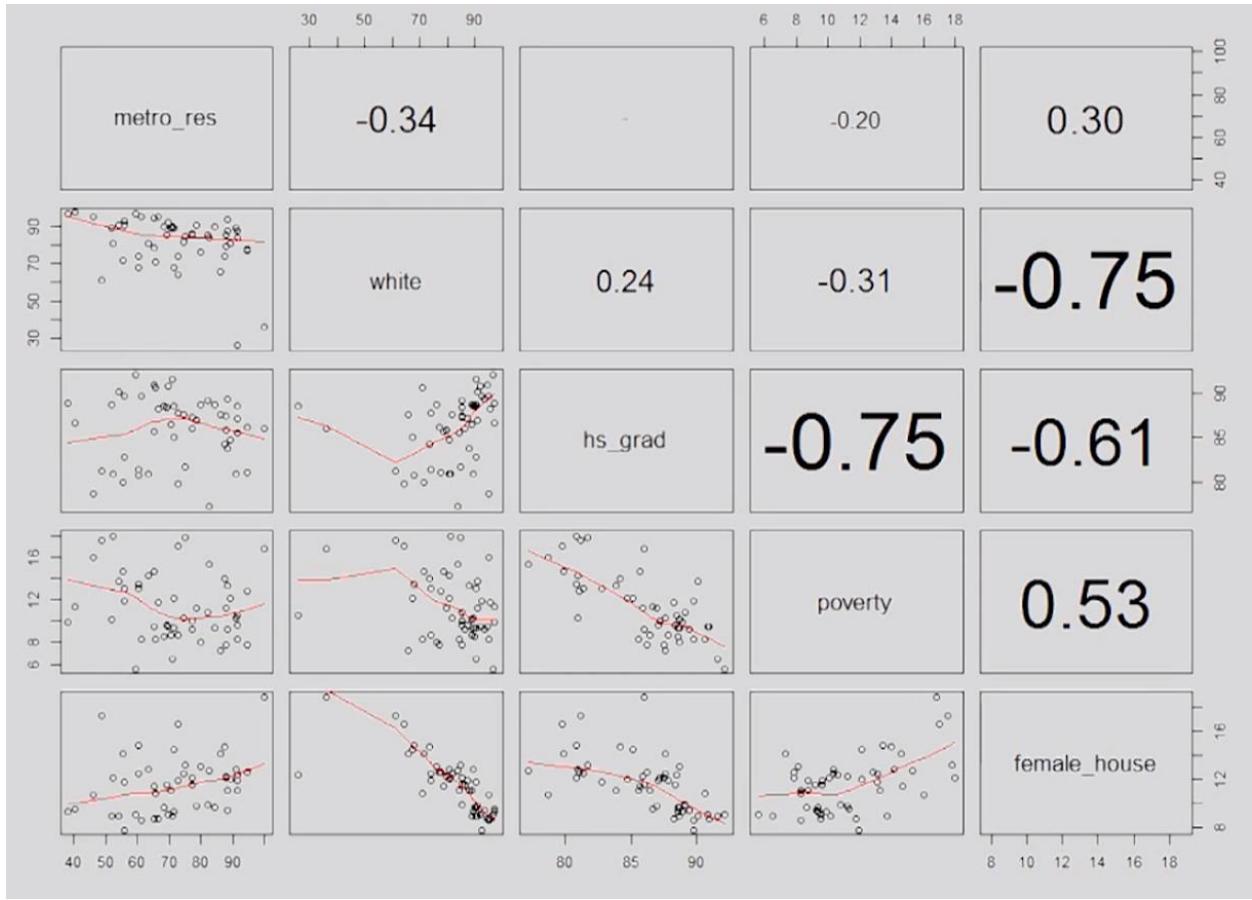


Рисунок 2.6

По діагоналі можемо бачити назви незалежних змінних, а на перетині з одної сторони діаграми розсіювання, а з іншої – значення коефіцієнту кореляції. За допомогою цих діаграм, та значень коефіцієнту кореляції можна знайти зайні змінні. Якщо якась змінна дуже сильно корелює з іншими, то таку змінну краще виключити з нашої моделі.

2. Метод підбору оптимальної моделі: спочатку будуємо модель, в яку включаємо всі змінні та розраховуємо коефіцієнт детермінації R^2 . Потім по черзі виключаємо змінні та розраховуємо R^2 : якщо він більше, то таку змінну краще виключити з моделі. Якщо більше не можемо знайти таку зміну, то знайдено оптимальну модель.

2.2 Методи вирішення задачі оцінки вартості нерухомого майна

В даному підрозділі будуть розглянуті більш досконалі методи вирішення задачі регресії за допомогою машинного навчання.

2.2.1 Дерева рішень

На сьогоднішній день існує безліч алгоритмів для розв'язку задач класифікації та регресії. Проте недоліком багатьох є відсутність змоги інтерпретувати результати, що може бути важливим в деяких прикладних задачах таких як скоринг. Натомість алгоритм «Дерева рішень» має значну перевагу з цієї сторони. Його ідея в чомусь може нагадувати деякі логічні мислення людини, крім цього, дерево легко візуалізувати, що є великим плюсом.

Приклад дерева рішень зображене на рисунку 2.7.

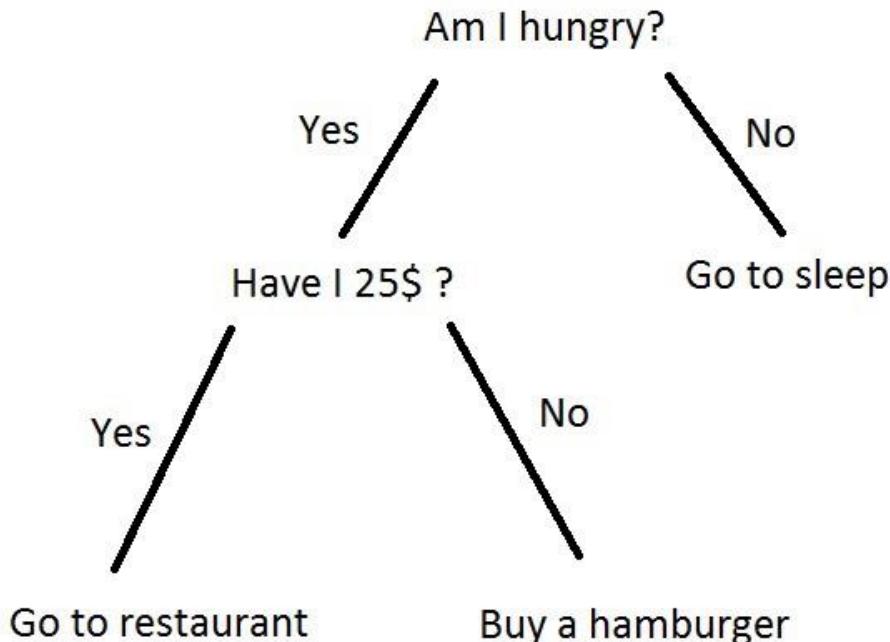


Рисунок 2.7

Суть алгоритму дерева прийняття рішення полягає в тому, що на кожному кроці відбувається розбиття по одній з ознак даних. Розбиття ж відбувається таким чином, щоб збільшити приріст інформації, зменшивши, відповідно, ентропію в групах, що утворилися після розбиття (дерево є бінарним, в кожній вершині визначається умова розбиття). Ентропія ж визначається наступним чином:

$$S = - \sum_{i=1}^N p_i \log_2 p_i, \quad (2.2)$$

де p_i – співвідношення i -того цільового значення до загальної кількості прикладів, що потрапили у дане розбиття.

Приріст інформації обчислюється наступним чином:

$$IG(Q) = S_O - \sum_{i=1}^q \frac{N_i}{N} S_i, \quad (2.3)$$

де q – число груп розбиття;

N_i – число елементів вибірки, у яких ознака Q має i -те значення.

Існують же також інші критерії розбиття, наприклад Джині, C4.5. Проте на практиці результати розбиття часто виходять однаковими у вищезгаданих алгоритмах.

В випадку задачі регресії критерієм якості стає дисперсія навколо середнього:

$$D = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - \frac{1}{\ell} \sum_{i=1}^{\ell} y_i)^2, \quad (2.4)$$

де l – кількість об'єктів в листку;

y_i – значення цільової ознаки.

Приклад апроксимації функції деревом рішення зображенено на рисунку 2.8.

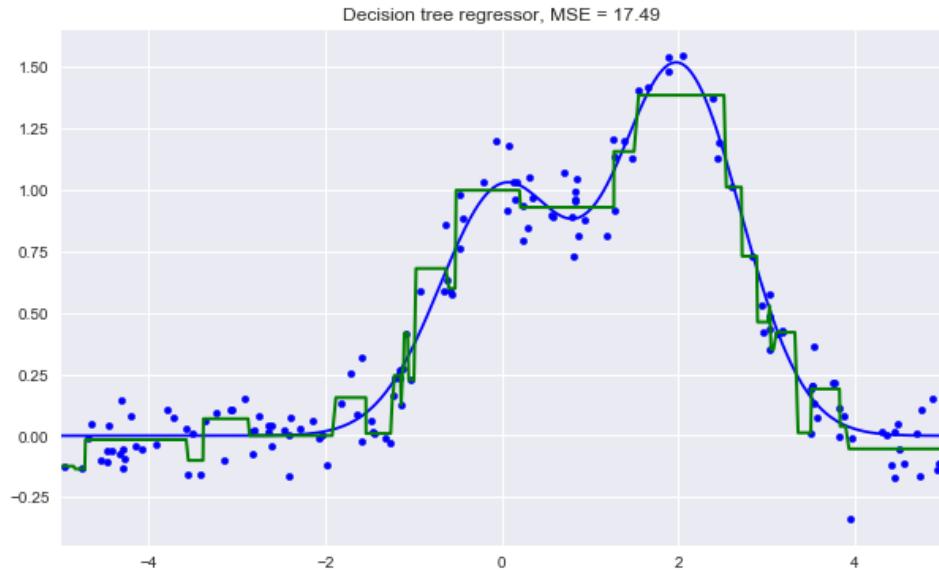


Рисунок 2.8

До переваг дерев можна також віднести стійкість до викидів, та і взагалі, дані алгоритми накладають менше вимог на дані та не потребують їх складної обробки.

2.2.2 Випадковий ліс

Проте звичайні дерева рішень хоч і є добре інтерпретуємими алгоритмами, проте вони не дають хороших результатів метрики на складних даних.

Натомість, для опису складних залежностей існують два головних підходи, які часто конкурують між собою в якості. Це глибинне навчання та ансамблі. Ми ж розглянемо один з випадків ансамблю – випадковий ліс. Хорошим прикладом ансамблів є теорема «Про журі присяжних» Кондорсе (1784). Якщо кожен з членів журі має незалежну думку і ймовірність його правильно прийнятого рішення є більшою за 0,5, то при збільшенні кількості журі, збільшується

ймовірність їх правильного рішення і ця ймовірність прямує до 1. Натомість, якщо ймовірність кожного з журі менша за 0,5, то ймовірність загалом прийнятого рішення прямує до 0 при збільшенні кількості присяжних.

$$\mu = \sum_{i=m}^N C_N^i p^i (1-p)^{N-i} \quad (2.5)$$

якщо $p > 0.5$, то $\mu > p$,

якщо $N \rightarrow \infty$, то $\mu \rightarrow 1$,

де N – кількість присяжних;

p – ймовірність правильно прийнятого рішення;

μ - ймовірність правильного фінального (загального) рішення;

m – мінімальна більшість членів журі;

C_N^i – число комбінацій із N по i .

Бутстреп – один з компонентів алгоритму роботи випадкового лісу. Даний метод був запропонований Лéо Брéйманом в 1994 році. Метод полягає в тому, що маючи вибірку X із N об'єктів ми M разів вибираємо з рівномірним розподіленням приклади з цієї вибірки складаючи нові. Тобто ми отримуємо M нових вибірок розміром N . Варто відзначити, що при такому підході в нових вибірках зможуть з'являтися повтори. Було показано, що з ймовірністю $1/3$ приклад має дублікат в цих підвибірках.

Наступним кроком є бегінг, ідея якого є в тому, що ми вчимо M дерев прийняття рішень (в загальному випадку може бути будь-який інший алгоритм) на отриманих вибірках після бутстрепа. Тоді ж фінальний результат буде середнім з усіх, що дали окремі дерева.

Також часто для кожного окремого регресора використовуються не всі ознаки, а певна частина. В комбінації з попередньо зазначеними методами це дає значну стійкість до перенавчання.

Отже, використовуючи низку менш слабких алгоритмів, можна отримати кращий результат.

2.2.3 Градієнтний бустинг

У випадку бегінгу випадкового лісу, обчислення відбуваються паралельно, тобто кожне дерево працює незалежно одне від одного. Проте, був запропонований підхід у послідовному використанні таких алгоритмів. Його ідея полягає в тому, що кожний наступний класифікатор чи регресор намагається виправити помилки попереднього. Такий метод називається бустингом.

Градієнтний бустинг - це техніка машинного навчання для задач класифікації і регресії, яка будує модель передбачення в формі ансамблю слабких моделей, зазвичай дерев рішень.

Мета будь-якого алгоритму навчання з учителем - визначити функцію втрат і мінімізувати її. Градієнтний бустинг використовує градієнтний спуск для досягнення цієї мети.

Градієнт - вектор, котрий своїм напрямком вказує напрямок найбільшого зростання деякої величини, значення якої змінюється від однієї точки простору до іншої, а по величині рівний швидкості росту цієї величини в цьому напрямку. Зворотній напрямок градієнта використовують для знаходження мінімуму.

Градієнтний спуск дуже затратний на великому обсязі даних, тому використовують стохастичний градієнтний спуск, який дає майже оптимальне рішення.

2.3 Критерії якості роботи моделі

При побудові системи передбачення цін на квартири недостатньо навчити модель та перевірити вручну три – чотири, або навіть двадцять квартир. Навіть якщо всі двадцять квартир будуть оцінені дуже точно, то це не дасть нам змоги стверджувати щось про якість роботи моделі через статистичну не значущість результатів цього експерименту через велику ймовірність випадкового вгадування.

Тому будемо ділити базу даних на дві вибірки: навчальна вибірка - 70% від усіх наявних об'єктів в базі даних, та 30% - тестова вибірка. Саме на цих 30-ти відсотках ми будемо перевіряти точність роботи моделі.

Будемо використовувати критерії якості, які найкраще підходять для задачі регресії:

1. Mean Absolute Percentage Error (MAPE) — середня абсолютна процентна помилка. Розраховується за формулою:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\Phi_i - \Pi_i}{\Phi_i} \right| \quad (2.6)$$

де Φ - фактичне значення вартості;

Π - прогнозне значення вартості.

2. MEDium Absolute Percentage Error (MedAPE) - медіанна абсолютна процентна помилка. Теж саме, що і MAPE, але замість середнього арифметичного використовується медіана, яка більш стійка до викидів.

3. Коефіцієнт детермінації R^2 , який приймає значення від 0 до 1. Чим більше це значення до одиниці, тим краще ми передбачаємо значення цільової змінної.
4. Mean Squared Error (MSE) — середній квадрат відхилення. Визначає середню суму квадратної різниці між фактичним значенням і прогнозованим значенням дляожної точки.

2.4 Порівняльний аналіз існуючих методів розв'язку задачі регресії

Порівняємо деякі з розглянутих нами методів розв'язку задачі регресії: дерева рішень, випадковий ліс та градієнтний бустинг. Випробуємо ці методи в рівних умовах на одних і тих самих даних та отримаємо наступні результати, які можна побачити на рисунках 2.9., 2.10., 2.11.:

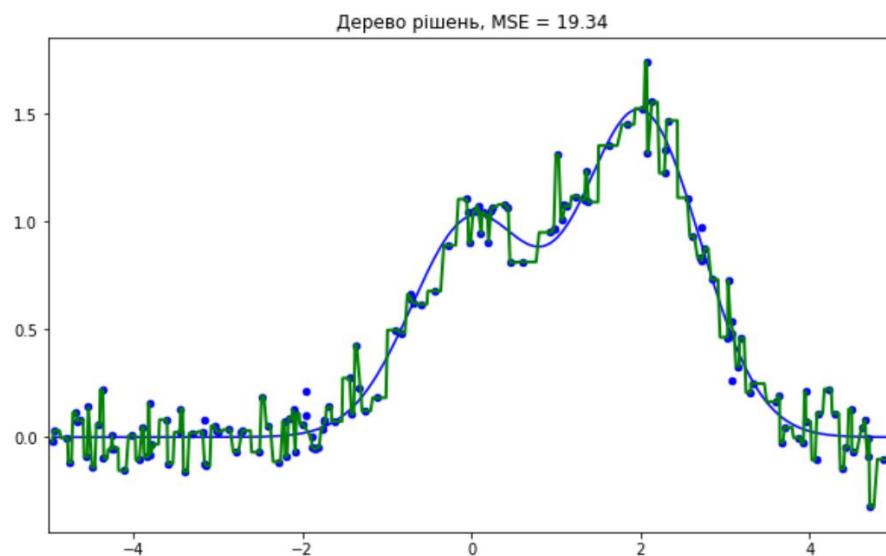


Рисунок 2.9 – Результат роботи дерев рішень, $MSE = 19.34$

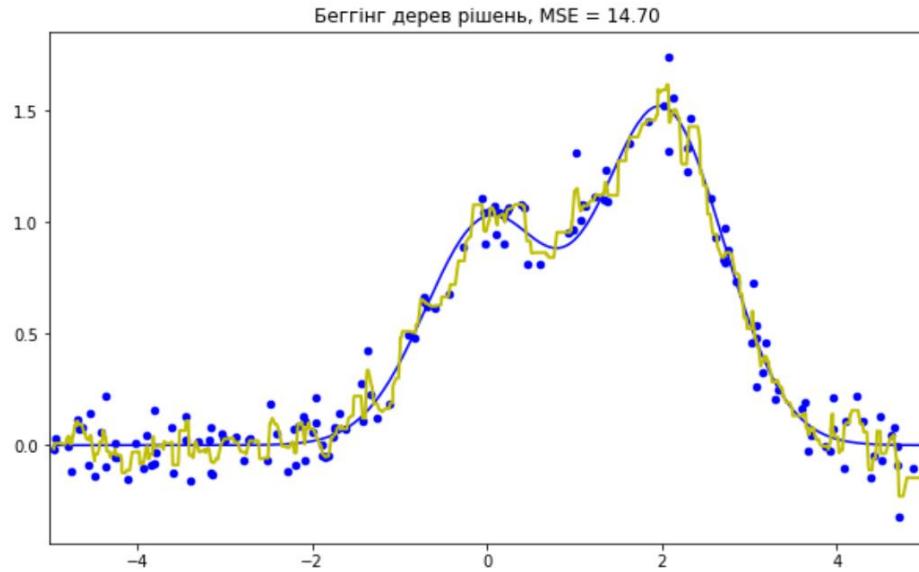


Рисунок 2.10 – Результат роботи бегінгу дерев рішень, MSE = 14.70

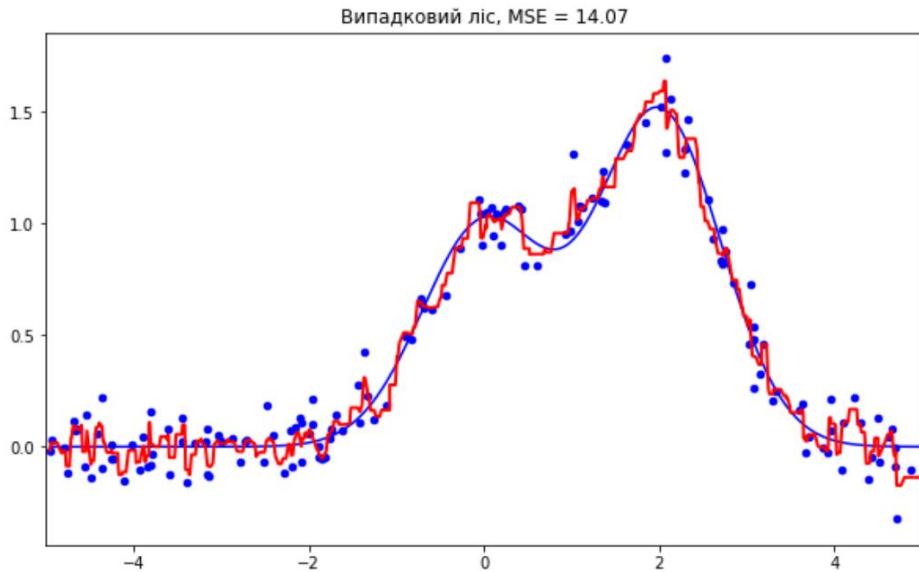


Рисунок 2.11 – Результат роботи випадкового лісу, MSE = 14.07

Як ми бачимо з графіків і значень помилки MSE, випадковий ліс з десяти дерев дає кращий результат, ніж одне дерево або бегінг з десяти дерев рішень.

2.5 Алгоритм

2.5.1 Інтерпретація даних

Одне з найважливіших завдань при розв'язуванні задачі оцінки вартості нерухомого майна - це інтерпретація даних.

Для визначення ознак, що впливають на ціну нерухомості в місті Києві, було розглянуто наступний список характеристик:

- Широта та довгота розміщення будинка
- Загальна площа об'єкта
- Житлова площа об'єкта
- Нежитлова площа об'єкта
- Площа кухні
- Кількість кімнат
- Поверх об'єкта
- Поверховість
- Вік будинку
- Тип стін будинку
- Тип опалення

Після тестування моделей з різними наборами ознак було виявлено, що тільки деякі з них сильно впливають на ціноутворення об'єктів, тому були відібрані наступні ознаки для подальшого вивчення:

- Відстань до центру міста
- Відстань до найближчої станції метрополітену
- Азимут
- Загальна площа об'єкта
- Поверховість
- Поверх об'єкта

Такі параметри, як широта та довгота розміщення будинку точно задають положення об'єкту, але використовувати їх безпосередньо для навчання не має сенсу. Але з їхньою допомогою можна згенерувати зовсім нові параметри, які впливають на ціноутворення. Після проведення багатьох експериментів було вирішено вивести з довготи та широти такі характеристики, як: відстань до центру міста, відстань до найближчої станції метрополітену, азимут – кут між напрямленням з центру міста на північ та напрямленням на місце розташування об'єкта. Ці три параметри дають змогу оцінити рівень престижності району, в якому знаходиться квартира.

Було виявлено, що усі інші параметри, які було розглянуто, не сильно впливають на ціноутворення. Однією з переваг випадкового лісу є те, що він дає змогу оцінити важливість усіх ознак, які використовуються в моделі. На рисунку 2.12. можна бачити рейтинг важливості ознак: відстань від центру міста вносить найбільший вклад (майже 40%), поверх квартири – найменший (приблизно 5%). Усі інші параметри мали ще менший відсоток, тому було вирішено ними знехтувати.

1. center_distance (0.399699)
2. metro_distance (0.213236)
3. azimuth (0.154019)
4. total_square_meters (0.092887)
5. floors_count (0.084609)
6. floor (0.055550)

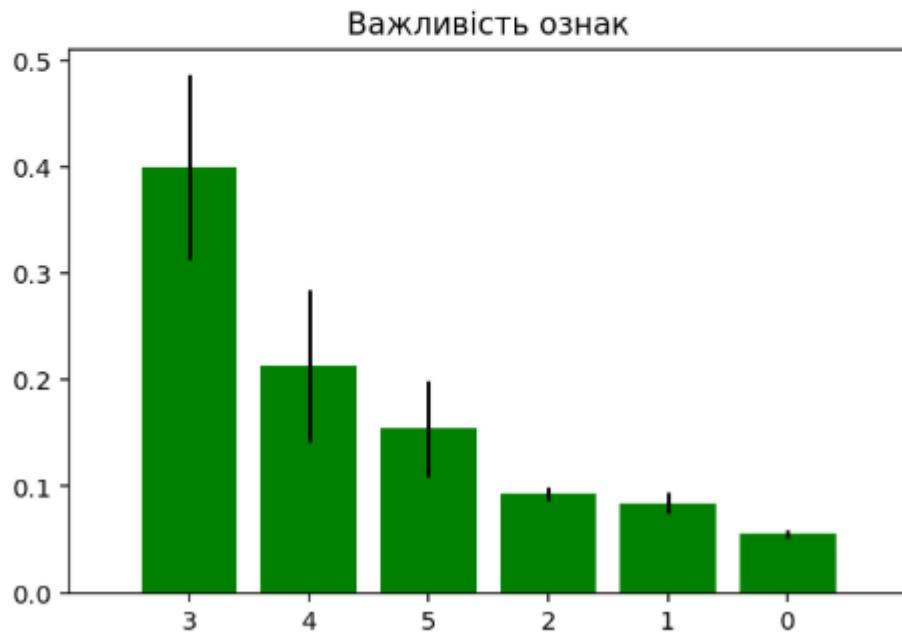


Рисунок 2.12 – Рейтинг важливості ознак в моделі «випадковий ліс»

2.5.2 Вибір методу машинного навчання

Після проведення порівняльного аналізу існуючих методів вирішення задачі регресії було прийнято рішення використовувати для подальших досліджень два методи: випадковий ліс та градієнтний бустинг.

Після низки практичних експериментів було виявлено, що точність передбачень зростає, якщо використовувати комбінацію вище зазначених методів. Саме тому кінцева модель – це середнє арифметичне передбачень випадкового лісу та градієнтного бустингу.

2.6 Висновки за розділом 2

У даному розділі було проведено аналіз математичних основ задачі регресії, розглянуто деякі методи вирішення задачі регресії: лінійна регресія, дерева рішень, випадковий ліс.

Крім цього, були визначені основні критерії якості роботи системи. Okremо було проведено порівняльний аналіз основних методів розв'язування задачі регресії в машинному навчанні.

РОЗДІЛ 3 АНАЛІЗ АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ ТА ПРАКТИЧНИХ РЕЗУЛЬТАТІВ

3.1 Обґрунтування вибору платформи та мови програмування

Мовою програмування було обрано Python 3.8. Такий вибір був зроблений через низку причин:

- Python – це дуже популярна мова програмування з великою спільнотою, яка його вдосконалює кожного дня.
- Python – це інтерпретована мова програмування, що означає, що будь-яка Python-програма може запускатися в будь-якій операційній системі.
- Python – це найпопулярніша, на даний момент, мова програмування для машинного навчання.

3.1.1 Збір бази даних

Веб-скрапер – це програма, яка автоматично збирає з веб-сайтів потрібну інформацію. Основою такої програми, написаної для даної роботи, виступає Python-бібліотека «requests». Це дуже популярна бібліотека, яка створена для зручної взаємодії з веб-сервісами.

За допомогою даної бібліотеки та API сайту DOM.RIA було зібрано список характеристик, якими описується кожна квартира на даному сайті. Після чого було також отримано список посилань на всі оголошення про продаж квартир в місті Києві.

Для збору потрібної інформації з цих сторінок веб-сайту DOM.RIA було використано регулярні вирази. Зібрана інформація записувалась в файл в форматі csv.

Використані бібліотеки:

- requests – для взаємодії з веб-сайтом DOM.RIA
- time – для оцінки ефективності роботи скрипту
- json – для збереження інформації в файлах у вигляді списків
- re – для роботи з регулярними виразами
- csv – для запису бази даних у форматі csv

Те, як виглядає база даних, після закінчення роботи веб-скрапера можна побачити на рисунку 3.1:

dataset_v2CSV
1 pri_total,currency,price_type,total_square_meters,kitchen_square_meters,living_square_meters,heating,building_year,wall_type_uk,floors_count,floor,rooms_count,publishing_date,latitude,longitude, realty_id,join 2 realty_id,join 3 72000,230,252,46,19,24,1649,1783,115,15,1,2020-05-23,59,431273767693,30,462018901403,17056212,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-dneprovskiy-perova-bulvar-16704892.html 4 87000,230,252,54,12,38,1649,442,118,3,2,2020-05-19,58,4915904812774,30,519846404299,14943737,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-solomenskiy-lohanovskogo-prospekt-17056211.html 5 6067200,240,252,126,12,75,1648,1468,109,16,4,3,2020-05-18,58,4234245,30,5137163,15621122,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-obolonskiy-priznenaya-ulitsa-14943737.html 6 59900,230,252,73,7,4,42,1648,442,110,9,4,3,2020-05-01,58,511553717032,30,60016184061,16869082,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-desnyanskiy-baltska-onore-de-ulitsa-16869082.html 7 69000,230,252,48,6,28,442,108,7,62,7,1648,41838,30,5180082,16531085,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-svyatohirskyi-anatolivska-prospekt-17041474.html 8 120000,230,252,92,6,7,4,62,7,1648,4136,108,9,1,2020-04-28,50,452891968818,30,380020406082,17014174,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-solomenskiy-lohanovskogo-prospekt-17041474.html 9 51000,230,252,38,7,7,8,20,3,1648,435,108,9,4,1,2020-04-24,50,411010349654,17012168,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-solomenskiy-lohanovskogo-prospekt-17041474.html 10 38500,230,252,46,5,9,33,3,,442,110,5,2,2,2020-04-24,50,4355654999999996,30,1855939999999997,16856391,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-solomenskiy-mihaila-dontsia-ulitsa-16856391.html 11 78800,230,252,73,5,13,71,32,52,1648,4469,1621,25,10,2,2020-04-24,50,409937764,30,506896449074,16825494,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-goloseevskiy-yasinskoho-ulitsa-16825494.html 12 48000,240,253,94,3,17,96,37,1648,1789,108,7,3,2,2020-05-18,58,42345529457,30,513692160119,16209085,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-pecherskiy-kazemira-malevicha-ulitsa-16209085.html 13 65000,230,252,46,7,29,1648,442,108,5,1,2,2020-05-13,50,418881105649,30,527488032085,16675810,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-pecherskiy-andri-barvysa-ulitsa-16675810.html 14 6072000,240,252,126,11,64,75,1648,1468,188,16,5,3,2020-05-23,50,423408575964,30,513729711045,16322827,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-pecherskiy-kazemira-malevicha-ulitsa-16322827.html 15 48000,240,253,89,16,61,49,78,1648,1789,108,7,3,2,2020-05-18,50,42341335437,30,513700206746,16213469,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-pecherskiy-kazemira-malevicha-ulitsa-16213469.html 16 48000,240,253,166,20,108,1648,1789,108,7,6,4,2020-05-18,50,4234245,30,5137163,16066781,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-goloseevskiy-lugovaya-julyanyi-ulitsa-17020920.html 17 22000,230,252,33,7,7,14,4,1653,442,111,5,2,1,2020-05-28,50,38536739473,30,45277207222,17020920,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-goloseevskiy-lugovaya-ulitsa-17020920.html 18 18500,240,253,58,,1653,1471,1462,5,2,1,2020-05-28,50,38536741671,30,452761343384,16921670,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-goloseevskiy-lugovaya-ulitsa-16921670.html 19 45000,240,253,126,11,64,75,1648,1468,108,16,3,3,2020-05-18,50,423457288901,30,513699504685,14819815,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-pecherskiy-kazemira-malevicha-ulitsa-14819815.html 20 36000,230,252,46,7,28,1648,442,110,9,2,2020-05-07,50,5014925928,30,496797342413,16707857,https://dom.ria.com/ru/ realty-perevireno-prodaja-kvartira-kiev-obolonksiy-marshala-malinovskogo-ulitsa-16707857.html

Рисунок 3.1 – Перші 20 рядків бази даних

3.1.2 Навчання моделей

Для навчання моделей було використано Jupyter Notebook – інструмент для інтерактивної розробки. Даний інструмент має такі переваги: можливість виконувати окремі частини коду, об'єднує в собі можливість писати текст, математичні формули, будувати графіки. Крім того Jupyter Notebook дуже просто почати використовувати: є можливість встановити на свій комп’ютер або користуватися онлайн-версією.

Також був використаний дистрибутив Anaconda, який об’єднує велику кількість бібліотек, які потрібні для машинного навчання та загалом для науки о

даних. Anaconda вирішує конфлікти, які виникають при одиночному встановлені бібліотек.

За основну бібліотеку, яка надає велику кількість алгоритмів машинного навчання, було взято бібліотеку Scikit-learn. Цей вибір був зроблений через широке розповсюдженість та велику кількість навчальних матеріалів з використання даної бібліотеки.

Процес роботи в Jupyter Notebook зображенено на рисунку 3.2. на прикладі виведення інформації про базу даних:

```
df.info()
df.head(5)

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22791 entries, 0 to 22950
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   price_total_usd    22791 non-null   float64
 1   total_square_meters 22791 non-null   float64
 2   kitchen_square_meters 16302 non-null   float64
 3   living_square_meters 14823 non-null   float64
 4   heating            22748 non-null   float64
 5   building_year      10355 non-null   float64
 6   wall_type_uk       22791 non-null   int64  
 7   floors_count       22791 non-null   int64  
 8   floor              22791 non-null   int64  
 9   rooms_count        22791 non-null   int64  
 10  publishing_date    22791 non-null   object 
 11  latitude           22791 non-null   float64
 12  longitude          22791 non-null   float64
 13  realty_id          22791 non-null   int64  
 14  url                22791 non-null   object 
 15  metro_distance     22791 non-null   int64  
 16  center_distance    22791 non-null   int64  
 17  price_metr         22791 non-null   float64
 18  azimuth            22791 non-null   float64
dtypes: float64(10), int64(7), object(2)
memory usage: 3.5+ MB

Out[7]:
```

	price_total_usd	total_square_meters	kitchen_square_meters	living_square_meters	heating	building_year	wall_type_uk	floors_count	floor	rooms_count	f
0	27500.0	22.0	5.0	13.0	1648.0	NaN	110	9	2	1	
1	35000.0	27.0	9.0	NaN	1648.0	NaN	108	22	18	1	
2	30000.0	29.0	7.0	14.0	1648.0	NaN	113	21	9	1	
3	64000.0	31.0	8.0	21.0	1648.0	NaN	108	6	3	1	
4	40000.0	32.0	7.0	14.0	1648.0	NaN	110	9	2	1	

Рисунок 3.2

3.1.3 Кінцевий програмний продукт

Було вирішено розробити кінцевий програмний продукт у вигляді прикладного програмного інтерфейсу (API). Вибір був зроблений через те, що API може бути використаний на будь-якому веб-сайті або в мобільному додатку. Тобто такий проект буде легко масштабувати.

Для створення API був використаний Django REST Framework (версія 3.11.0). Цей веб-фреймворк використовується разом з Django фреймворком та надає весь необхідний функціонал для побудови якісного програмного продукту. Django був обраний через його популярність, наявність детальної документації та всього необхідного функціоналу.

3.2 Керівництво користувача кінцевим програмним продуктом

3.2.1 Браузерна версія

Браузерна версія API має інтуїтивно зрозумілий інтерфейс: ви можете знайти документацію дляожної кінцевої точки API, просто відвідавши URL-адресу у своєму браузері.

На рисунку 3.3 зображено кореневу кінцеву точку «root», де користувач може прочитати можливості даного продукту та перейти за одним з двох посилань: перше посилання веде на сторінку «users», друга - на сторінку «flats».

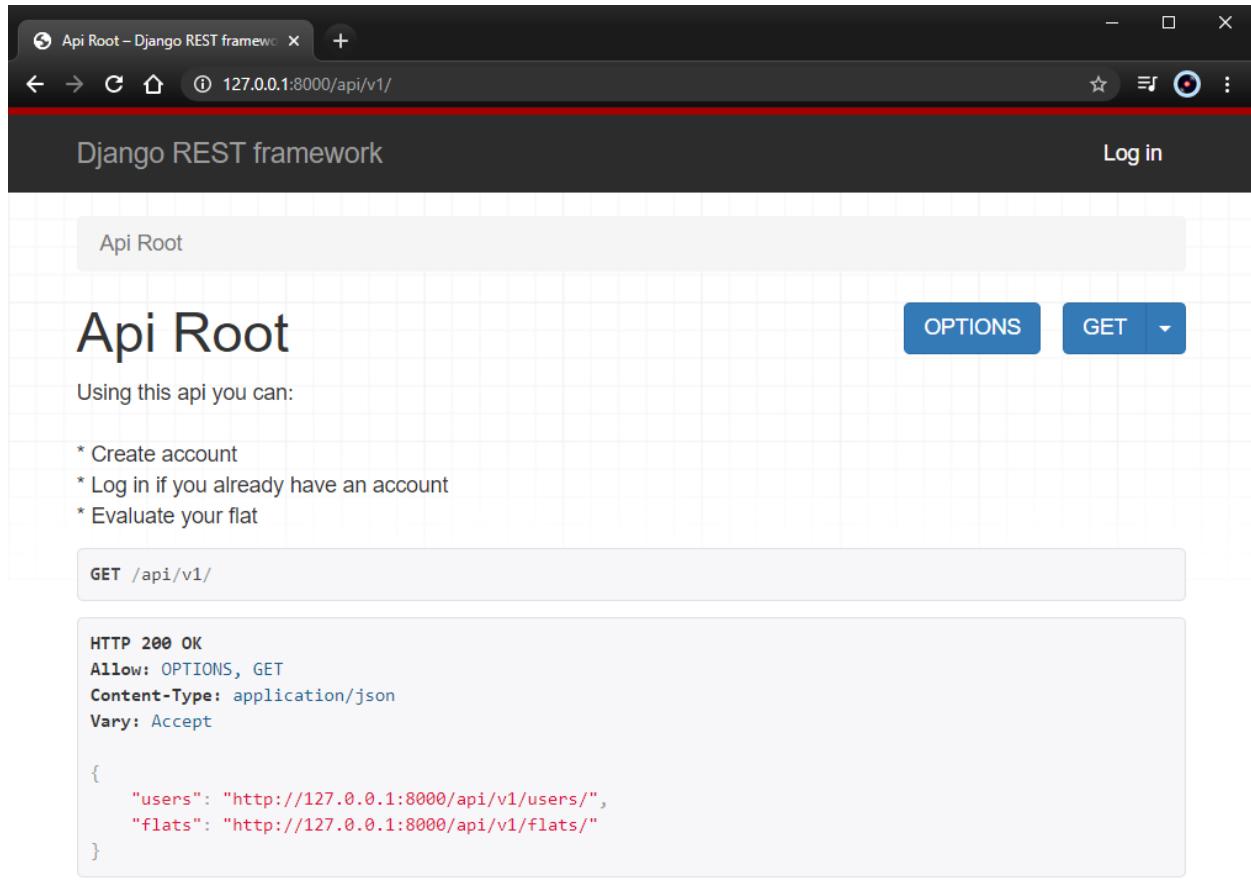


Рисунок 3.3 – Кінцева точка «root»

На рисунку 3.4 зображено кінцеву точку «users», де користувач має змогу проглянути імена усіх зареєстрованих користувачів та зареєструвати нового користувача, скориставшись формою, яка знаходиться внизу сторінки «users».

```

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 6,
            "username": "user"
        },
        {
            "id": 2,
            "username": "admin"
        },
        {
        }
    ]
}

```

Рисунок 3.4 – Кінцева точка «users»

На рисунку 3.5 зображене як виглядає кінцева точка «flats» для неавторизованого користувача: доступ закритий.

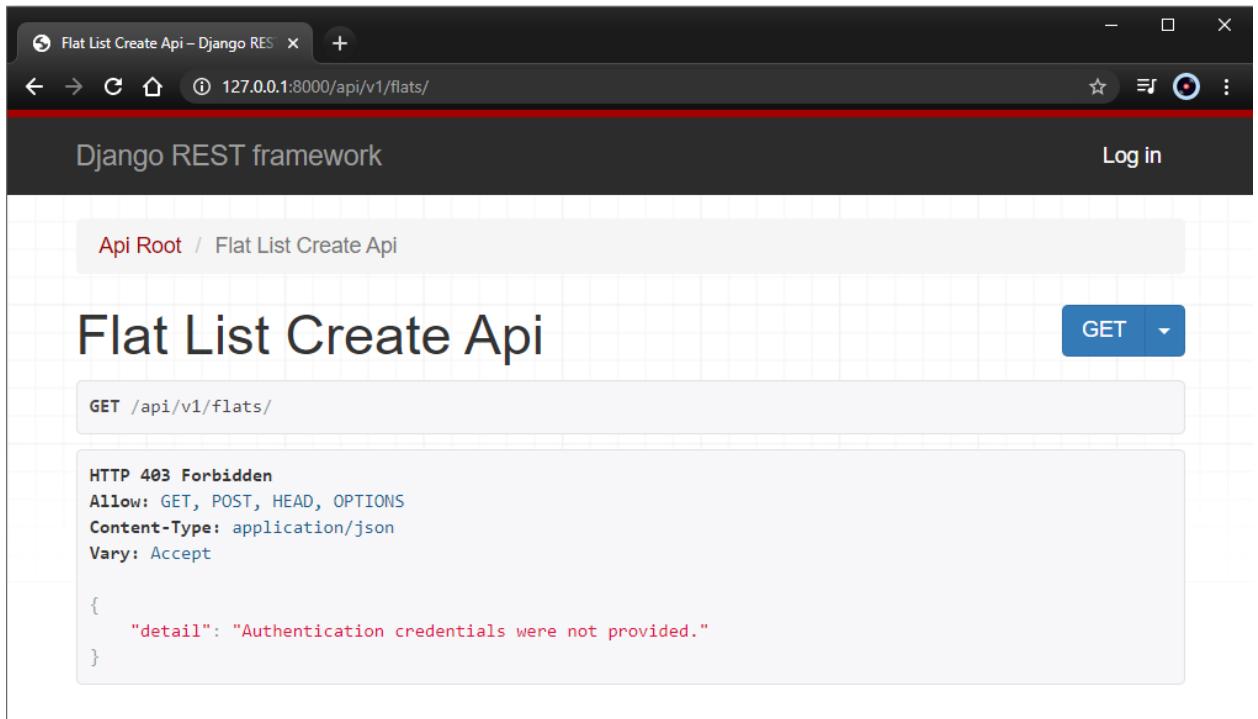


Рисунок 3.5

На рисунку 3.6 зображена кінцева точка «login», на якій користувач може зайти до свого кабінету, використовуючи унікальне ім'я користувача та пароль.

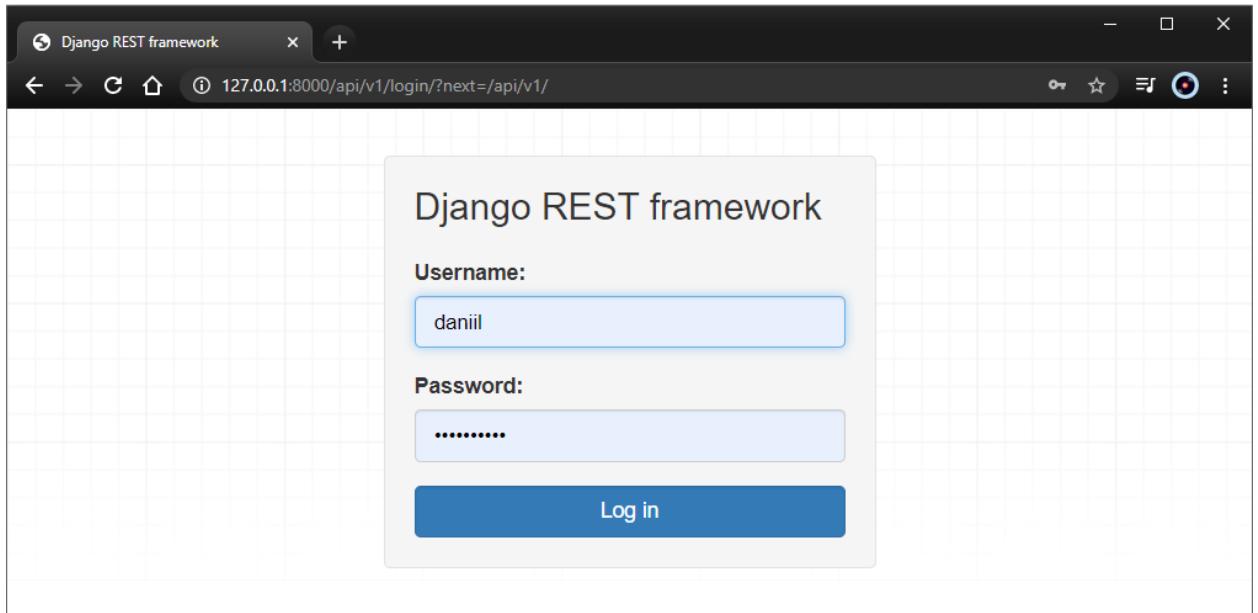


Рисунок 3.6

На рисунку 3.7 зображено кінцеву точку «flats». Таким чином вона виглядає для авторизованого користувача, котрий вже додавав квартири для їх

оцінювання. Користувач може переглядати лише свої власні об'єкти і не може бачити об'єкти інших користувачів. Також на цій сторінці можна бачити вимоги до даних.

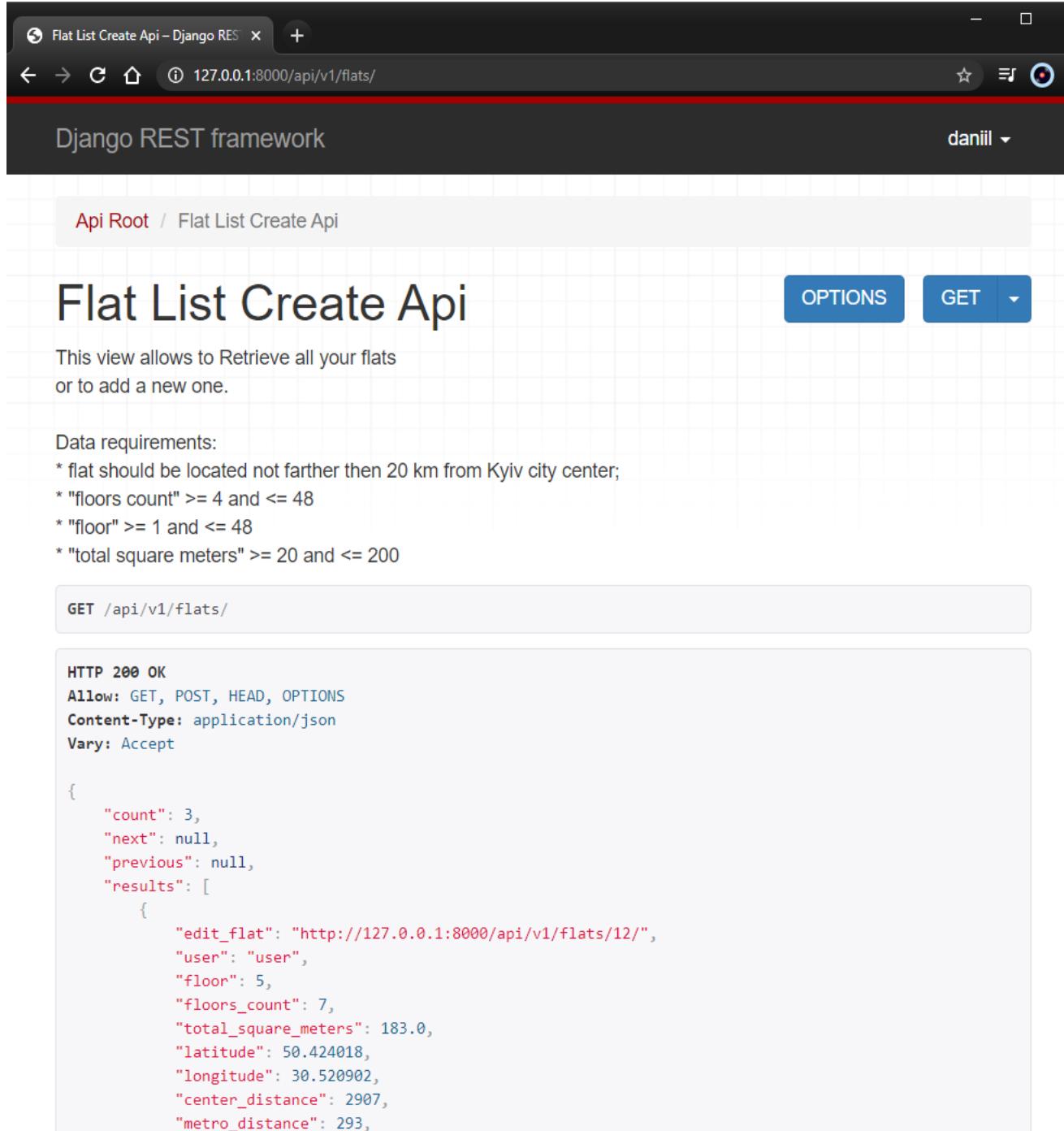


Рисунок 3.7

В кінці сторінки «flats» знаходиться форма, за допомогою якої можна додати новий об'єкт для його оцінки. Ця форма зображена на рисунку 3.8.

The screenshot shows a Django REST framework interface. At the top, it says "Django REST framework" and "danil". Below that, there is a JSON representation of an apartment object:

```

{
    "floor": 4,
    "floors_count": 9,
    "total_square_meters": 183.0,
    "latitude": 50.424018,
    "longitude": 30.520902,
    "center_distance": 2907,
    "metro_distance": 293,
    "predicted_price_meter": 1809,
    "predicted_price_total": 331053
}
]
}

```

Below the JSON, there are two tabs: "Raw data" (which is selected) and "HTML form". The "HTML form" tab contains fields for "Floor", "Floors count", "Total square meters", "Latitude", and "Longitude", each with an input field. At the bottom right of the form is a blue "POST" button.

Рисунок 3.8

Для оцінки квартири необхідно ввести такі параметри: поверх квартири, поверховість будинку, загальна площа квартири та координати квартири: широту і довготу. Після введення параметрів і натискання кнопки «POST», у разі успішної перевірки даних, в списку оцінених квартир з'явиться новий об'єкт, а користувача буде перенаправлено на сторінку для перегляду доданого об'єкта.

Якщо вимоги до даних не були задоволені, користувач побачить пояснення помилки та об'єкт не буде збережено. Приклад такої помилки наведено на рисунку 3.9.

```
POST /api/v1/flats/  
  
HTTP 400 Bad Request  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept  
  
{  
    "non_field_errors": [  
        "Flat must be located no further than 20 km from Kyiv city center."  
    ]  
}
```

Рисунок 3.9

Сторінку детальною інформацією по одному окремому об'єкту зображено на рисунку 3.10. Система додає розраховує додаткові характеристики, наприклад, відстань від центра міста, відстань до найближчої станції метрополітену. Цільова характеристика називається «predicted_price_total» - загальна передбачена ціна в доларах США.

Дана кінцева точка також дозволяє редагувати параметри квартири, за допомогою форми в кінці сторінки. Після редагування система заново перераховує передбачену ціну квартири та інші допоміжні параметри.

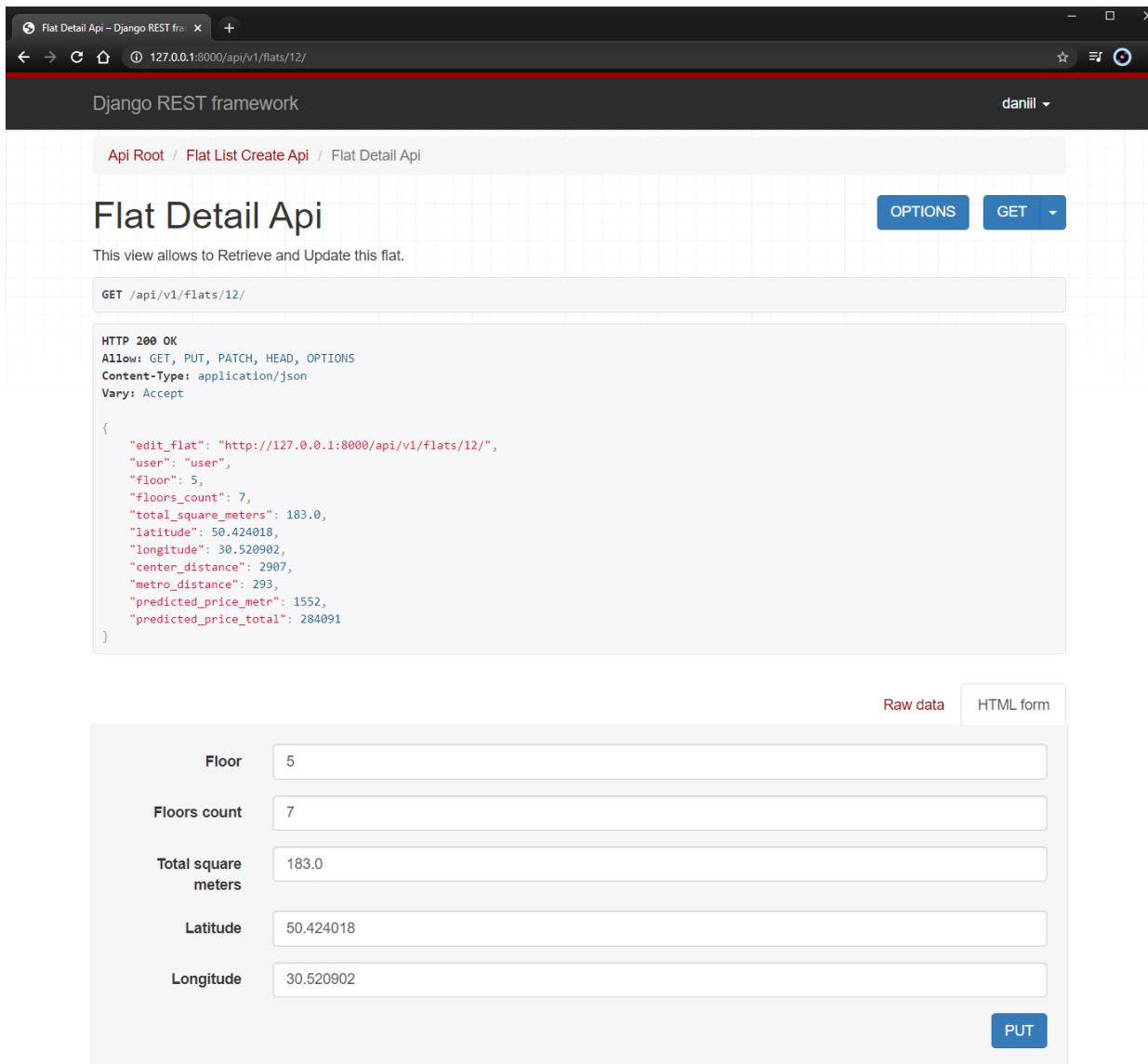


Рисунок 3.10 – Кінцева точка «flats-detail»

В майбутньому даний програмний продукт планується використовувати за бізнес моделлю freemium. Дано бізнес модель передбачає дві версії: безкоштовна, але обмежена в чомусь, та повна платна версія. Саме тому в даному API обмежено кількість запитів, яку може зробити користувач безкоштовної версії. Такий користувач має змогу надіслати не більше 500 запитів на день. При спробі зробити більше запитів, користувач отримає у відповідь помилку номер 429, та побачить повідомлення про час, через який він зможе продовжити користуватися сервісом. Приклад такої помилки зображеного на рисунку 3.11.

```
GET /api/v1/  
  
HTTP 429 Too Many Requests  
Allow: GET, OPTIONS  
Content-Type: application/json  
Retry-After: 86388  
Vary: Accept  
  
{  
    "detail": "Request was throttled. Expected available in 86388 seconds."  
}
```

Рисунок 3.11

3.2.2 Взаємодія з API інших веб-сервісів

В майбутньому планується зробити цей програмний продукт доступним в інтернеті. В цьому випадку, для того, щоб використовувати даний API на своєму веб-сайті або, наприклад, мобільному додатку, потрібно перш за все створити нового користувача. Це можна зробити через браузерну версію.

Розглянемо приклад взаємодії з даним API, використавши уже зазначену Python-бібліотеку requests. На рисунку 3.10 зображені базові методи взаємодії і перевірено роботу основних кінцевих точок API.

```

import requests

url = 'http://127.0.0.1:8000/api/v1/users/'
response = requests.get(url)
print(response.text)

url = 'http://127.0.0.1:8000/api/v1/users/'
response = requests.post(url,
                         data={'username': 'new_user', 'password': 'mypassword12', 'confirm_password': 'mypassword12'})
print(response.status_code)

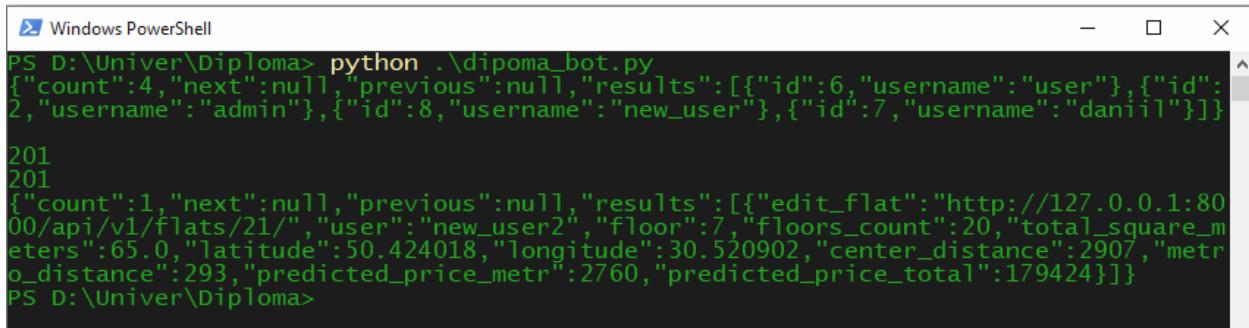
url = 'http://127.0.0.1:8000/api/v1/flats/'
response = requests.post(url,
                         data={'floor': 7, 'floors_count': 20, 'total_square_meters': 65, 'latitude': 50.424018,
                               'longitude': 30.520902},
                         auth=('new_user', 'mypassword12'))
print(response.status_code)

response = requests.get(url, auth=('new_user', 'mypassword12'))
print(response.text)

```

Рисунок 3.10 – базові методи взаємодії з API

На рисунку 3.11 можна побачити відповідь від сервера API.



```

Windows PowerShell
PS D:\Univer\Diploma> python .\dipoma_bot.py
[{"count":4,"next":null,"previous":null,"results":[{"id":6,"username":"user"}, {"id":2,"username":"admin"}, {"id":8,"username":"new_user"}, {"id":7,"username":"daniil"}]}
201
201
{"count":1,"next":null,"previous":null,"results":[{"edit_flat": "http://127.0.0.1:8000/api/v1/flats/21/", "user": "new_user2", "floor": 7, "floors_count": 20, "total_square_meters": 65.0, "latitude": 50.424018, "longitude": 30.520902, "center_distance": 2907, "metro_distance": 293, "predicted_price_metr": 2760, "predicted_price_total": 179424}]}
PS D:\Univer\Diploma>

```

Рисунок 3.11 – Відповідь від сервера API

3.3 Аналіз роботи системи з оцінки нерухомості

3.3.1 Аналіз якості роботи системи

Для оцінки якості роботи моделей з оцінювання квартир, було використано 3 основних критерія якості, які були розглянуті в другому розділі: середня абсолютна процентна помилка (MAPE), медіанна абсолютна процентна помилка (MedAPE) та коефіцієнт детермінації R^2 .

В результаті були отримані такі оцінки якості системи:

- MAPE = 12.2%
- MedAPE = 4.2%
- $R^2 = 0.84$

Такі результати є досить хорошими, зважаючи на те, що навчання моделі здійснювалося на даних з оголошень про продаж квартир, а не на даних про вже продані об'єкти. В майбутньому планується зробити систему, яка буде навчена на достовірній інформації про вже продані квартири, що повинно привести до зростання точності оцінки.

Досить велика різниця між оцінками MAPE та MedAPE може буди пояснена чутливістю оцінки MAPE до викидів.

Крім того, вважається, що системи, які мають коефіцієнт детермінації більше ніж 0.8 досить добре роблять прогнози.

Отже, можна зробити висновок, що система з оцінки квартир вмісті Києві вийшла достатньо якісною з точки зору точності передбачень.

3.3.2 Аналіз роботи програмного продукту

Усі складові програмного продукту, а саме: веб-скрапер, навчання моделі та кінцевий програмний продукт (API) працюють якісно, без помилок.

В майбутньому даною системою зможуть користуватися як звичайні люди для вирішення своїх особистих задач, так і спеціалісти у сфері нерухомості для вирішення робочих задач. Також буде змога підключити даний API до будь-якого веб-сайту або мобільного додатку.

Також планується розробити окрему систему для оцінки комерційної нерухомості.

3.4 Висновки за розділом 3

У даному розділі було обґрунтовано вибір мови програмування, платформ та бібліотек для створення програмного продукту.

Також було наведено детальне керівництво користувача кінцевим програмним продуктом.

Крім того, було проведено аналіз роботи системи, її якості з точки зору точності передбачень. Також було сплановано дії для покращення системи в майбутньому.

РОЗДІЛ 4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

4.1 Постановка задачі проектування

Спроектувати програмний продукт для автоматичної оцінки нерухомого майна на прикладі оцінювання квартир у місті Києві. Програмний продукт був виконаний у вигляді прикладного програмного інтерфейсу для того, щоб його можна було легко інтегрувати з будь-яким вже існуючим веб-сервісом.

4.2 Обґрунтування функцій та параметрів програмного продукту

Виходячи з конкретних цілей, які реалізуються :

F1 – архітектура програмного додатку: а) прикладний програмний інтерфейс, б) повноцінний веб-сайт, в) мобільний додаток.

F2 – збір бази даних: а) веб-скрейпинг, б) використання відкритих баз даних, в) співробітництво з ріелторськими компаніями.

F3 – алгоритми навчання моделей: а) лінійна регресія, б) випадковий ліс.

F4 – збереження результатів роботи - а) запис результатів у сховище та вивід користувачу, б) лише вивід користувачу.

Виходячи з представлених варіантів будуємо морфологічну карту (рис.4.1).

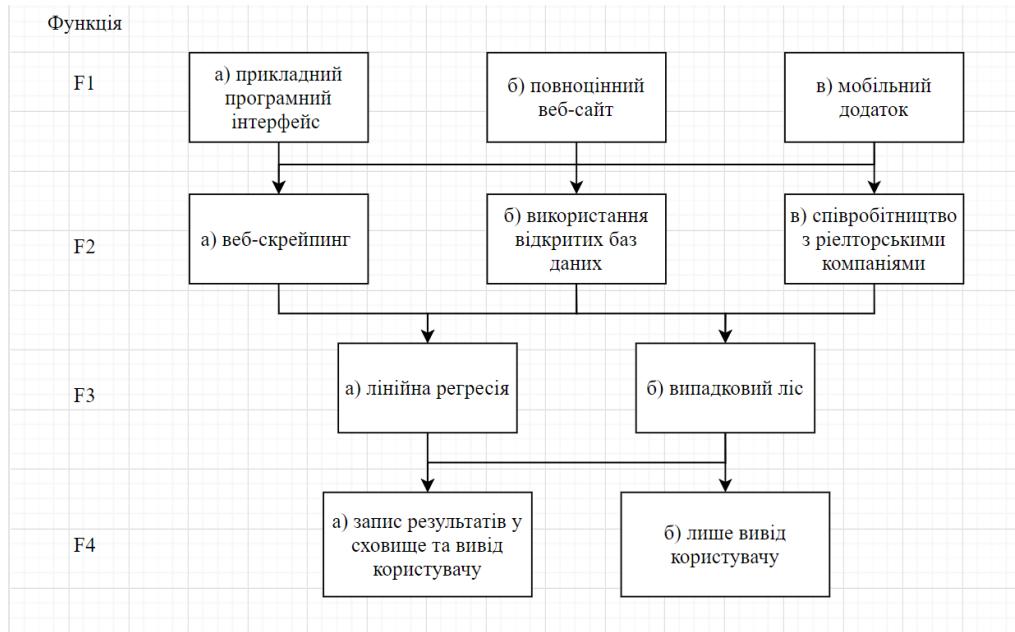


Рисунок 4.1 – Морфологічна карта

Спираючись на карту була побудована позитивно-негативна матриця(табл. 4.1).

Таблиця 4.1 – Позитивно-негативна матриця

Основна функція	Варіант реалізації	Переваги	Недоліки
F1	A	Легкість реалізації, можливість інтеграції на інших веб-ресурсах	Неможливість використовувати як окремий самостійний веб-сайт
	Б	Можливість використовувати як окремий самостійний веб-сайт, особистий бренд	Складність реалізації, великі затрати на рекламу, втрачається аудиторія потенційних клієнтів, які використовують мобільними пристроями
	В	Можливість викорис-	Складність реалізації, великі

		тovувати як окремий самостійний мобільний додаток, особистий бренд	затрати на рекламу, втрачається аудиторія потенційних клієнтів, які використовують комп'ютери
F2	A	Актуальність даних, незалежність від ріелторських компаній, достатній об'єм даних	Складність реалізації, неможливість отримати дані про вже продані об'єкти
	Б	Легкість реалізації	Неактуальність даних, малий об'єм доступних даних
	В	Актуальність даних, можливість працювати з якісними даними про вже продані об'єкти	Складність реалізації
F3	A	Легкість реалізації	Необхідність якісно обробляти дані для навчання моделі, не найкращий показник точності
	Б	Велика точність, добре справляється з викидами	Складність реалізації
F4	A	Можливість проглянути усі особисті об'єкти, які були оцінені	Потрібне надійне сховище даних, складність реалізації
	Б	Легкість реалізації	Недоступність результатів по-передніх оцінок об'єктів

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X5. На основі даних, що представлені у літературі, визначаємо мінімальні, середні отримуванні та максимально допустимі значення(табл. 4.2).

Таблиця 4.2 – Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Час розробки, людина*год	X1	55	150	350
Час збору бази даних, людина*год	X2	2	15	40
Час навчання моделі, с	X3	60	130	225
Величина об'єму пам'яті для збереження даних, Мб	X4	0	8	16

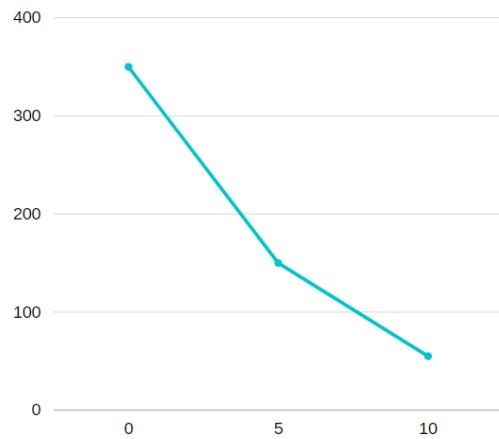


Рисунок 4.2 – Значення параметра X1

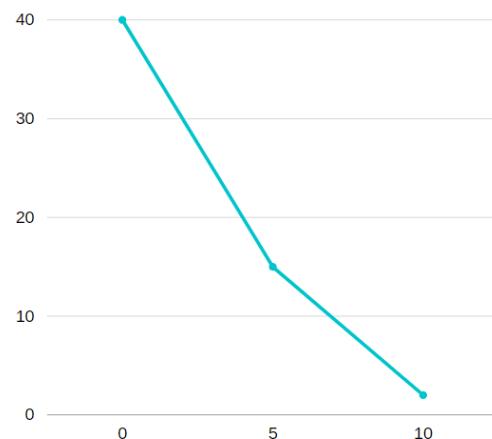


Рисунок 4.3 – Значення параметра X2

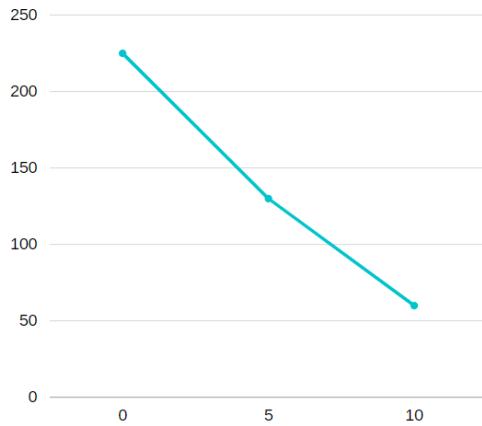


Рисунок 4.4 – Значення параметра X3

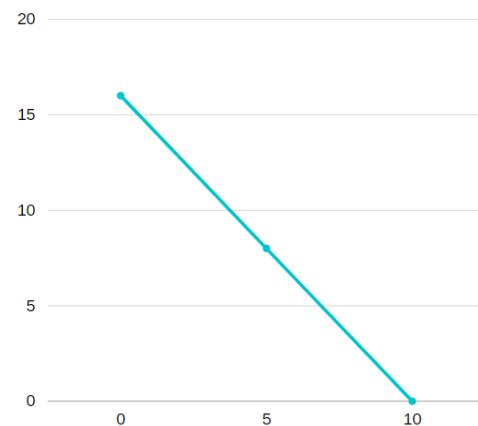


Рисунок 4.5 – Значення параметра X4

Вагомість параметрів оцінюється за допомогою методів попарного зрівняння. Ранги варіюються від 1 до 5, де 1 – найменший ранг, 5 – найбільший. Результати наведені в табл. 4.3-4.4.

Таблиця 4.3 – Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	Відхилення Δ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	4	3	3	4	4	4	4	26	8.5	72.25
X2	3	4	4	3	2	3	3	22	4.5	20.25
X3	1	1	2	2	3	1	1	11	-6.5	42.25
X4	2	2	1	1	1	2	2	11	-6.5	42.25
Разом	10	10	10	10	10	10	10	70	0	177

Таблиця 4.4 – Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	>	<	<	>	>	>	>	>	1.5

X1 та X3	>	>	>	>	>	>	>	>	1.5
X1 та X4	>	>	>	>	>	>	>	>	1.5
X2 та X3	>	>	>	>	<	>	>	>	1.5
X2 та X4	>	>	>	>	>	>	>	>	1.5
X3 та X4	<	<	>	>	>	<	<	<	0.5

Визначимо коефіцієнт конкордації:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 177}{7^2(4^3 - 4)} = 0.72 > W_k = 0.67$$

Так як коефіцієнт конкордації більше нормативного, результати вважають достовірними.

Розрахунок вагомості параметрів наведено в табл. 4.5

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри	Параметри X_i				Перший крок		Другий крок	
	X_1	X_2	X_3	X_4	b_i	K_{bi}	b_i	K_{bi}
X_1	1	1,5	1,5	1,5	5.5	0,344	21.5	0,361
X_2	0,5	1	1,5	1,5	4.5	0,281	16.25	0,275
X_3	0,5	0,5	1	0,5	2.5	0,156	9.25	0,157
X_4	0,5	0,5	1,5	1	3.5	0,219	12.5	0,207
Загалом:					16	1	59	1

Як видно з таблиці 4.5., різниця значень коефіцієнтів вагомості не перевищує 5% після другої ітерації, тому більшої кількості ітерацій не потрібно.

На основі порівняльного аналізу варіантів реалізації функцій за їх перевагами та недоліками ми обрали наступні два варіанти реалізації:

1. F1(a)=>F2(a)=>F3(б)=>F4(a)
2. F1(a)=>F2(a)=>F3(б)=>F4(б)

Таблиця 4.6

Основна функція	Варіант реалізації	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	a)X1	60	9.83	0.361	3.55
F2	a)X2	15	5	0.275	1.38
F3	б)X3	120	6.36	0.157	0.1
F4	a)X4	12	2.5	0.207	0.52
	б)X4	4	7.5	0.207	1.55

Обрахуємо коефіцієнти якості кожного з варіантів розробки:

$$K_{я1} = 3.55 + 1.38 + 0.1 + 0.52 = 5.55$$

$$K_{я2} = 3.55 + 1.38 + 0.1 + 1.55 = 6.58$$

Оскільки варіант 2 має більший коефіцієнт якості, він є найкращим.

4.3 Економічний аналіз варіантів розробки

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

При цьому два варіанти мають різні додаткові завдання:

- 3.1. Запис результатів до бази даних. Для першого варіанту.
- 3.2. Вивід результатів лише на екран. Для другого варіанту.

У варіанті 1 присутнє наступне додаткове завдання під номером 3.1.

У варіанті 2 присутнє наступне додаткове завдання під номером 3.2.

За ступенем новизни завдання 1 відноситься до групи А, завдання 2, 3.1 відноситься до групи Б, завдання 3.1 відноситься до групи В.

За складністю алгоритми, які використовуються в завданні 1 належать до 1 групи, завдання 2, 3.1, 3.2 – до 3 групи.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_p = 90$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_n = 1.7$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{ck} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{ct} = 0.8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 * 1.7 * 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для інших завдань. Для другого завдання (використовується алгоритм третьої групи складності , ступінь новизни Б), тобто $T_p = 19$ людино-днів, $K_p = 0.9$, $K_{ck} = 1$, $K_{ct} = 0.8$. Отже:

$$T_2 = 27 * 0.9 * 0.8 = 19.44 \text{ людино-днів.}$$

Аналогічно для завдання 3.1 та 3.2:

$$T_3 = 27 * 0.9 * 0.8 = 19.44$$

$T_p = 12$ людино-днів, $K_p = 0.6$, $K_{ck} = 1$, $K_{ct} = 0.8$

$$T_4 = 12 * 0.6 * 0.8 = 5,76$$

Визначимо повну трудомісткість варіантів (людино-днів):

$$T_1 = 122,4 + 19,44 + 19,44 = 161,28$$

$$T_2 = 122,4 + 19,44 + 5,76 = 147,6$$

Найбільш трудомістким завданням є 1. Далі вважається, що робочий день складає 8 годин, в тижні п'ять робочих днів. В розробці бере участь два програміста з окладом 9000 грн та аналітик з окладом 8500 грн. Визначимо середню заробітну плату за годину:

$$C_q = \frac{2 * 9000 + 8500}{3 * 21 * 8} = 52,58$$

Тоді заробітну плату визначимо за формулою:

Заробітна розробників за варіантами становить:

$$C_{3П} = 52,58 * 8 * 161,28 = 67840,8$$

$$C_{3П} = 52,58 * 8 * 147,6 = 62086,5$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{ВІД} = 67840,8 * 0,22 = 14924,98$$

$$C_{ВІД} = 62086,5 * 0,22 = 13659,03$$

Визначимо витрати на оплату однієї машино-години. Так як одна машина обслуговує двох програмістів з окладом 9000 грн. з коефіцієнтом зайнятості 0,6 та одного аналітика з окладом 8500, то для трьох машин отримаємо:

$$C_r = 12 * 9000 * 2 * 0,6 + 12 * 8500 * 0,6 = 190\ 800 \text{ (грн)}$$

Враховуючи додаткову заробітну плату 40%

$$C_{3П} = 190\ 800 * (1 + 0,4) = 267\ 120 \text{ (грн)}$$

Відрахування на соціальне страхування 22%

$$C_{ВІД} = 267\ 120 * 0,22 = 58\ 766,4 \text{ (грн)}$$

Розрахуємо амортизаційні підрахунки (амортизація 25%, вартість ЕОМ 17000 грн)

$$C_A = K_{TM} * K_A * \Pi_{PR} = 1,15 * 0,25 * 17000 = 4887,5 \text{ (грн)}$$

Розрахуємо витрати на ремонт та профілактику:

$$C_P = K_{TM} * \varPhi_{PP} * K_P = 1,15 * 17000 * 0,05 = 977,5 \text{ (грн)}$$

Розрахуємо ефективний годинний фонд часу ПК за рік:

$$T_{EF} = (365 - 104 - 11 - 16) * 8 * 0,8 = 1497,6 \text{ год}$$

Розрахуємо витрати на електроенергію

$$C_{EL} = 1497,6 * 0,6 * 0,8 * 1,75 = 1257,98 \text{ грн}$$

Накладні витрати рівні:

$$C_H = 17000 * 0,67 = 11\ 390 \text{ (грн)}$$

Отже експлуатаційні витрати:

$$C_{EKC} = 267\ 120 + 58\ 766,4 + 4\ 887,5 + 977,5 + 1257,98 + 11\ 390 = \\ 344\ 399,38 \text{ (грн)}$$

Собівартість однієї машино-години буде:

$$C_{M-G} = \frac{344\ 399,38}{1497,6} = 229,97 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, то витрати на оплату машинного часту в залежності від обраного варіанту, складають:

$$C_m = 229,97 * 8 * 161,28 = 296\ 716,49 \text{ грн.}$$

$$C_m = 229,97 * 8 * 147,6 = 271\ 548,58 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = 296\ 716,49 * 0,67 = 198\ 800,04 \text{ (грн)}$$

$$C_H = 271\ 548,58 * 0,67 = 181\ 937,55 \text{ (грн)}$$

Отже, вартість розробки програмного продукту за варіантами становить:

$$C_{\text{ПП}} = 67840,8 + 14924,98 + 296\ 716,49 + 198\ 800,04 = 578\ 282,31 \text{ (грн)}$$

$$C_{\text{ПП}} = 62086,5 + 13659,03 + 271\ 548,58 + 181\ 937,55 = 529\ 231,66 \text{ (грн).}$$

4.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня:

$$K_{\text{ТЕР1}} = \frac{5,55}{578\ 282,31} = 9,60 * 10^{-6}$$

$$K_{\text{ТЕР2}} = \frac{6,58}{529\ 231,66} = 12,43 * 10^{-6}$$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{TEP2} = 12,43 * 10^{-6}$.

4.5 Висновки до розділу 4

Отже враховуючи всі дослідження, що описані вище, можна сказати, що 2 варіант реалізації є найбільш оптимальним зі сторони якісно-економічної оцінки. Його коефіцієнт техніко-економічного рівня складає $12,43 * 10^{-6}$.

Розробка цього варіанту передбачає такі обов'язкові завдання як:

Серед завдань між якими ставився вибір в даному варіанті реалізовані такі завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;
3. Вивід результатів лише на екран;

ВИСНОВКИ

В даній роботі була розглянута актуальність задачі оцінки вартості нерухомого майна. Було проведено аналіз попиту на нерухомість в Україні, аналіз існуючих рішень: американська компанія Zillow, українська компанія DOM.RIA.

Також було проведено аналіз математичних основ задачі регресії, розглянуто деякі методи вирішення задачі регресії, від найпростіших базових й до більш складних та точних. Були визначені основні критерії якості, якими доцільно користуватися для оцінки роботи моделей.

Було зібрано власну базу даних для навчання моделей. Інформацію було взято з оголошень про продаж квартир на веб-сайті DOM.RIA.

Було створено модель для оцінки квартир у місті Києві. Створено також програмний продукт – прикладний програмний інтерфейс (API), який використовує дану модель.

Крім цього, було проаналізовано роботу системи, проведено аналіз якості її роботи. Також було зазначено майбутні дії для покращення й удосконалення даної системи з оцінювання вартості нерухомого майна.

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Zillow: a web-site. URL: <https://www.zillow.com/> (Last accessed: 12.03.2020).
2. DOM.RIA: a web-site. URL: <https://dom.ria.com/ru/kalkuljator-stoimosti-kvartiru/> (Last accessed: 7.04.2020).
3. Open Machine Learning course. URL: <https://mlcourse.ai/> (Last accessed: 17.05.2020).
4. Основы статистики. Анатолий Карпов. URL: <https://stepik.org/course/76/promo> (Last accessed: 12.05.2020).
5. Django Framework Documentation. URL: <https://docs.djangoproject.com/en/3.0/> (Last accessed: 18.04.2020).
6. Django REST Framework Documentation. URL: <https://www.django-rest-framework.org/> (Last accessed: 18.04.2020).
7. Azimuth in Python. URL: <https://pastebin.com/PHeWmiEN> (Last accessed: 10.04.2020).

ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

СИСТЕМА АВТОМАТИЧНОГО ОЦІНЮВАННЯ НЕРУХОМОГО МАЙНА

Науковий керівник
Дідковська М.В.

Виконав
Іванов Даніїл

МЕТА РОБОТИ

1) Сформувати достатню для роботи базу даних.

2) Проаналізувати існуючі методи побудови регресійних моделей.

3) Розробити програмний продукт для оцінки вартості квартир у місті Києві.

АКТУАЛЬНІСТЬ



Інвестиції



Час



Суб'єктивність

IVANOV DANIIL | 2020

ПРИКЛАДИ ІСНУЮЧИХ РІШЕНЬ

1. АМЕРИКАНСЬКА КОМПАНІЯ ZILLOW
2. УКРАЇНСЬКА КОМПАНІЯ DOM.RIA

Розрахуйте вартість своєї квартири

В області *

В місті *

В районі

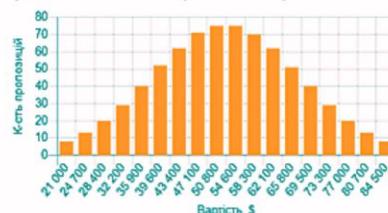
Кількість кімнат *

Загальна площа, м² *

Розрахувати вартість**49 490 \$**

Додати оголошення

(доларів США, станом на травень 2020)



*Виконано в якості консультаційної послуги на основі аналізу 757 подібних квартир

Онлайн-калькулятор від компанії DOM.RIA.

IVANOV DANIIL | 2020

ФОРМУВАННЯ БАЗИ ДАНИХ

Веб
парсер



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22791 entries, 0 to 22950
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   price_total_usd    22791 non-null   float64
 1   total_square_meters 22791 non-null   float64
 2   kitchen_square_meters 16302 non-null   float64
 3   living_square_meters 14823 non-null   float64
 4   heating            22748 non-null   float64
 5   building_year      10355 non-null   float64
 6   wall_type_uk       22791 non-null   int64  
 7   floors_count       22791 non-null   int64  
 8   floor               22791 non-null   int64  
 9   rooms_count        22791 non-null   int64  
 10  publishing_date    22791 non-null   object 
 11  latitude           22791 non-null   float64
 12  longitude          22791 non-null   float64
 13  realty_id          22791 non-null   int64  
 14  url                22791 non-null   object 
 15  metro_distance     22791 non-null   int64  
 16  center_distance    22791 non-null   int64  
 17  price_metr         22791 non-null   float64
 18  azimuth            22791 non-null   float64
dtypes: float64(10), int64(7), object(2)
memory usage: 3.5+ MB
```

IVANOV DANIIL | 2020

ОБРОБКА БАЗИ ДАНИХ

★ Продаю Зк квартиру 116 кв. м, Драгомирова улица 14А
возле метро Дружбы народов в районе Печерский в Киеве

22 214 000 \$ за объект
600 378 444 грн • 191 500 \$ за м² курс валют



Торг Вторичное жилье

Продажа квартиры в ЖК Покровский посад

3 комнаты • 15 этаж из 23

Площадь 149000 м² • 56 м² • 46 м²

Торг Первичное жилье

Продажа квартиры в ЖК Эврика

3 комнаты • 23 этаж из 24

Площадь 76.53 м² • 45.47 м² • 892 м²

IVANOV DANIIL | 2020

МЕТОДИ ПОБУДОВИ РЕГРЕСІЙНИХ МОДЕЛЕЙ

ЛІНІЙНА РЕГРЕСІЯ

переваги:

- Добре вивчені
- Дуже швидкі
- Практично поза конкуренцією, коли ознак дуже багато

недоліки:

- Багато вимог до даних
- Погано працюють зі складними нелінійними залежностями

IVANOV DANIIL | 2020

МЕТОДИ ПОБУДОВИ РЕГРЕСІЙНИХ МОДЕЛЕЙ

ВИПАДКОВИЙ ЛІС

переваги:

- Висока точність передбачення
- Майже не чутливий до викидів в даних
- Існують методи оцінювання значущості окремих ознак в моделі

недоліки:

- Складніше інтерпретувати результати, на відміну від одного дерева
- Потребує багато пам'яті

IVANOV DANIIL | 2020

МЕТОДИ ПОБУДОВИ РЕГРЕСІЙНИХ МОДЕЛЕЙ

ГРАДІЕНТНИЙ БУСТИНГ



переваги:

- Може описати досить складну функцію
- В стандартних завданнях дуже часто є найефективнішим алгоритмом



недоліки:

- Схильний до перенавчання

IVANOV DANIIL | 2020

АЛГОРИТМ

Випадковий ліс



градієнтний бустинг

IVANOV DANIIL | 2020

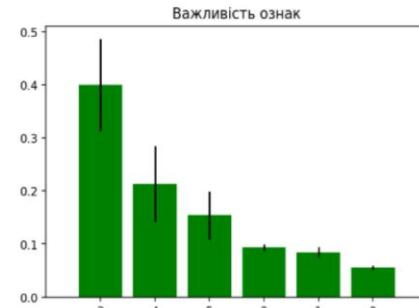
АЛГОРИТМ

Інтерпретація даних

Latitude + Longitude =

- center_distance - відстань до центра міста
- metro_distance - відстань до найближчої станції метрополітену
- azimuth - кут між напрямком на північ та на центр міста

1. center_distance (0.399699)
2. metro_distance (0.213236)
3. azimuth (0.154019)
4. total_square_meters (0.092887)
5. floors_count (0.084609)
6. floor (0.055550)



Рейтинг важливості ознак в моделі Random forest.

IVANOV DANIIL | 2020

ЗАСОБИ ДЛЯ РОЗРОБКИ

Мова програмування

Python 3.8

Збір бази даних

requests – для взаємодії з веб-сайтом DOM.RIA
re – для роботи з регулярними виразами

Навчання моделей

Jupyter Notebook
Scikit-learn

Кінцевий програмний продукт

Django REST Framework

IVANOV DANIIL | 2020

КРИТЕРІЇ ЯКОСТІ

**1) Mean Absolute
Percentage Error (MAPE)**

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\Phi_i - \Pi_i}{\Phi_i} \right|$$

**2) MEDium Absolute
Percentage Error (MedAPE)**

3) Коефіцієнт детермінації R^2

IVANOV DANIIL | 2020

АНАЛІЗ ЯКОСТІ РОБОТИ СИСТЕМИ

MAPE = 12.2%

MedAPE = 4.2%

$R^2 = 0.84$

IVANOV DANIIL | 2020

КЕРІВНИЦТВО КОРИСТУВАЧА API

БРАУЗЕРНА ВЕРСІЯ

```

Django REST framework
Content-Type: application/json
Vary: Accept

{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 6,
            "username": "user"
        },
        {
            "id": 7,
            "username": "daniil"
        }
    ]
}
  
```

Raw data HTML form

Username: Required: 150 characters or fewer. Letters, digits and @/./_- only.

Password:

Confirm password:

POST

IVANOV DANIIL | 2020

КЕРІВНИЦТВО КОРИСТУВАЧА API

БРАУЗЕРНА ВЕРСІЯ

```

Django REST framework
Content-Type: application/json
Vary: Accept

{
    "count": 4,
    "next": null,
    "previous": null,
    "results": [
        {
            "floor": 4,
            "floors_count": 9,
            "total_square_meters": 183.0,
            "latitude": 50.424018,
            "longitude": 30.520902,
            "center_distance": 2907,
            "metro_distance": 293,
            "predicted_price_meter": 1809,
            "predicted_price_total": 33085
        }
    ]
}
  
```

Raw data HTML form

Floor:

Floors count:

Total square meters:

Latitude:

Longitude:

POST

IVANOV DANIIL | 2020

КЕРІВНИЦТВО КОРИСТУВАЧА API

ПРОГРАМНА ВЗАЄМОДІЯ

```
import requests
import json

url = "http://127.0.0.1:8000/api/v1/users/"
response = requests.post(url, data={
    'username': 'new_user',
    'password': 'mypassword12',
    'confirm_password': 'mypassword12'})
print(response.status_code)

url = "http://127.0.0.1:8000/api/v1/flats/"
response = requests.post(
    url,
    data={'floor': 7, 'floors_count': 20, 'total_square_meters': 65,
           'latitude': 50.424018, 'longitude': 30.520902},
    auth=('new_user2', 'mypassword12'))
print(response.status_code)

response = requests.get(url, auth=('new_user2', 'mypassword12'))
print(response.json())
```

```
PS D:\Univer\Diploma> python .\dipoma_bot.py
201
201
[{"count": 2, "next": None, "previous": None, "results": [{"edit_flat": "http://127.0.0.1:8000/api/v1/flats/22",
  "user": "new_user2", "floor": 7, "floors_count": 20, "total_square_meters": 65.0, "latitude": 50.424018,
  "longitude": 30.520902, "center_distance": 2907, "predicted_price_metr": 2760, "predicted_price_total": 179424}, {"edit_flat": "http://127.0.0.1:8000/api/v1/Flats/23/", "user": "new_user2", "floor": 7, "floors_count": 20, "total_square_meters": 65.0, "latitude": 50.424018, "longitude": 30.520902, "center_distance": 2907, "predicted_price_metr": 2760, "predicted_price_total": 179424}]]
```

КЕРІВНИЦТВО КОРИСТУВАЧА API

ПРОГРАМНА ВЗАЄМОДІЯ

Freemium

Безкоштовна версія:
>= 500 запитів на день

Платна версія:
без обмежень

GET /api/v1/

HTTP 429 Too Many Requests
 Allow: GET, OPTIONS
 Content-Type: application/json
 Retry-After: 86388
 Vary: Accept

```
{
  "detail": "Request was throttled. Expected available in 86388 seconds."
}
```

Приклад помилки №429.

ВИСНОВКИ

- Проведено аналіз актуальності задачі оцінки вартості нерухомого майна, досліджено існуючі рішення.
- Проведено аналіз існуючих методів машинного навчання для розв'язку задачі регресії.
- Зібрано власну базу даних.
- Розроблено алгоритм та побудовано за ним модель оцінки вартості квартир з медіанною абсолютною процентною помилкою, рівною 4,2%.
- Створено прикладний програмний інтерфейс.

IVANOV DANIIL | 2020

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ

- Розміщення API на сервері для відкритого користування.
- Використання даних про вже продані об'єкти.
- Розробка окремої системи для оцінки комерційної нерухомості.
- Розробка версій для використання в інших містах України.

IVANOV DANIIL | 2020

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ ДЛЯ ТРЕНАВАННЯ МОДЕЛІ

kyiv_flats_valuation.ipynb

```

import xgboost as xgb
import pandas as pd
import numpy as np
from geopy.distance import geodesic
import mpmath as math
from sklearn.metrics import mean_absolute_error, r2_score, median_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
import pickle

def get_azimuth(latitude, longitude):
    """Returns angle between north and city center."""
    city_center_coordinates = [50.450075, 30.524205]
    rad = 6372795

    llat1 = city_center_coordinates[0]
    llong1 = city_center_coordinates[1]
    llat2 = latitude
    llong2 = longitude

    lat1 = llat1*math.pi/180.
    lat2 = llat2*math.pi/180.
    long1 = llong1*math.pi/180.
    long2 = llong2*math.pi/180.

    cl1 = math.cos(lat1)
    cl2 = math.cos(lat2)
    sl1 = math.sin(lat1)
    sl2 = math.sin(lat2)
    delta = long2 - long1
    cdelta = math.cos(delta)
    sdelta = math.sin(delta)

    y = math.sqrt(math.power(cl2*sdelta,2)+math.power(cl1*sl2-sl1*cl2*cdelta,2))
    x = sl1*sl2+cl1*cl2*cdelta

```

```

ad = math.atan2(y,x)

x = (cl1*sl2) - (sl1*cl2*cdelta)
y = sdelta*cl2
z = math.degrees(math.atan(-y/x))

if (x < 0):
    z = z+180.

z2 = (z+180.) % 360. - 180.
z2 = - math.radians(z2)
anglerad2 = z2 - ((2*math.pi)*math.floor((z2/(2*math.pi)))) )
angledeg = (anglerad2*180.)/math.pi

return round(angledeg, 2)

def get_metro_distance(latitude, longitude):
    """Returns distance to the nearest metro station."""
    flat_coordinates = [latitude, longitude]
    distances = list(map(lambda x, y: geodesic(flat_coordinates, [x, y]).meters,
                         metro_stations['latitude'], metro_stations['longitude']))

    metro_distance = min(distances)

    return int(metro_distance)

def get_center_distance(latitude, longitude):
    city_center_coordinates = [50.450075, 30.524205]
    center_distance = geodesic(city_center_coordinates, [latitude, longitude]).meters

    return int(center_distance)

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

def median_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)

```

```

return np.median(np.abs((y_true - y_pred) / y_true)) * 100

def print_metrics(prediction, val_y):
    val_mae = mean_absolute_error(val_y, prediction)
    median_AE = median_absolute_error(val_y, prediction)
    r2 = r2_score(val_y, prediction)

    print("R\u00b2: {:.2}'.format(r2))
    print("Середня абсолютна похибка: {:.3}"
%'.format(mean_absolute_percentage_error(val_y, prediction)))
    print('Медіанна абсолютна похибка: {:.3}'
%'.format(median_absolute_percentage_error(val_y, prediction)))

flats_path = 'kyiv_flats_dataset.csv'
df = pd.read_csv(flats_path)
df.head(5)
df.info()

metro_path = 'kyiv_metro_stations.csv'
metro_stations = pd.read_csv(metro_path)
metro_stations.info()
metro_stations.head(5)

flats_path = 'kyiv_flats_dataset.csv'
df = pd.read_csv(flats_path)

df['price_metr'] = df['price_total_usd']/df['total_square_meters']

city_center_coordinates = [50.450075, 30.524205]
df['azimuth'] = list(map(lambda x, y: get_azimuth(x, y), df['latitude'], df['longitude']))

df = df.loc[(df['center_distance'] < 15000)]
df['azimuth'] = df['azimuth'].round(0)

df.info()
df.head(5)
df.describe()

```

```
# Deleting of outliers
first_quartile = df.quantile(q=0.25)
third_quartile = df.quantile(q=0.75)
IQR = third_quartile - first_quartile
outliers = df[(df > (third_quartile + 1.5 * IQR)) | (df < (first_quartile - 1.5 * IQR))].count(axis=1)
outliers.sort_values(axis=0, ascending=False, inplace=True)

outliers = outliers.head(2000)
df.drop(outliers.index, inplace=True)
```

```
df.info()
```

```
# Assign the target variable
y = df['price_metr']

# List of features which we will use building the models
features = [
    'floor',
    'floors_count',
    'total_square_meters',
    'center_distance',
    'metro_distance',
    'azimuth',
]
```

```
X = df[features]
```

```
# Perform random sampling of data ( 0.75/0.25 )
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
```

```
#Random Forest model
rf_model = RandomForestRegressor(n_estimators=300,
                                 n_jobs=-1,
                                 bootstrap=False,
                                 criterion='mse',
                                 max_features=3,
                                 random_state=1,
```

```

        max_depth=30,
        min_samples_split=5
    )

rf_model.fit(train_X, train_y)
rf_prediction = rf_model.predict(val_X).round(0)

print_metrics(rf_prediction, val_y)

# XGBoost model
xgb_model = xgb.XGBRegressor(objective ='reg:gamma',
                               learning_rate = 0.01,
                               max_depth = 20,
                               n_estimators = 2000,
                               nthread = -1,
                               eval_metric = 'gamma-nloglik',
                               )

xgb_model.fit(train_X, train_y)
xgb_prediction = xgb_model.predict(val_X).round(0)

print_metrics(xgb_prediction, val_y)

# Final model (combination of Random Forest and XGBoost)
prediction = rf_prediction * 0.5 + xgb_prediction * 0.5

print_metrics(prediction, val_y)

# Importance of attributes in the Random forest model
importances = rf_model.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf_model.estimators_],
            axis=0)
indices = np.argsort(importances)[::-1]

print("Рейтинг важливості ознак:")
for f in range(X.shape[1]):
    print("%d. %s (%f)" % (f + 1, features[indices[f]], importances[indices[f]]))

```

```

plt.figure()
plt.title("Важливість ознак")
plt.bar(range(X.shape[1]), importances[indices], color="g", yerr=std[indices],
align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()

```

```

# Evaluation of single apartments
# Apartments URL: https://dom.ria.com/ru/realty-prodaja-kvartira-kiev-
predslavinskaya-ulitsa-16232758.html
# Total price: 320 000 USD.

```

```

flat = pd.DataFrame({
    'floor':[4],
    'floors_count':[7],
    'total_square_meters':[183],
    'latitude':[50.424018],
    'longitude':[30.520902]
})

```

```

flat['center_distance'] = get_center_distance(flat['latitude'][0], flat['longitude'][0])
flat['metro_distance'] = get_metro_distance(flat['latitude'][0], flat['longitude'][0])

```

```

flat['azimuth'] = list(map(lambda x, y: get_azimuth(x, y), flat['latitude'],
flat['longitude']))
flat['azimuth'] = flat['azimuth'].round(0)

```

```

flat = flat.drop('latitude', axis=1)
flat = flat.drop('longitude', axis=1)

```

```

rf_prediction_price_metr = rf_model.predict(flat).round(2)
xgb_prediction_price_metr = xgb_model.predict(flat).round(2)

```

```

rf_total_price = rf_prediction_price_metr*flat['total_square_meters'][0]
xgb_total_price = xgb_prediction_price_metr*flat['total_square_meters'][0]

```

```

avg_price_metr = rf_prediction_price_metr * 0.5 + xgb_prediction_price_metr * 0.5

```

```
avg_total_price = (rf_prediction_price_metr * 0.5 + xgb_prediction_price_metr *  
0.5)*flat['total_square_meters'][0]  
  
print(f'Ціна за метр, предсказана моделью Random Forest:  
{int(rf_prediction_price_metr)} дол.')  
print(f'Ціна за метр, передбачена моделлю XGBoost:  
{int(xgb_prediction_price_metr)} дол.')  
print("Середня ціна за метр по двум моделям: {int(avg_price_metr)} дол.")  
  
print(f'Ціна за всю квартиру, передбачена моделлю Random Forest:  
{int(rf_total_price)} дол.')  
print(f'Ціна за всю квартиру, передбачена моделлю XGBoost:  
{int(xgb_total_price)} дол.')  
print(f'Середня ціна за всю квартиру по двум моделям: {int(avg_total_price)} дол.')  
  
# Save the model to disk with pickle  
filename = 'rf_model.sav'  
pickle.dump(rf_model, open(filename, 'wb'))  
filename = 'xgb_model.sav'  
pickle.dump(xgb_model, open(filename, 'wb'))
```

ДОДАТОК В ЛІСТИНГ КІНЦЕВОГО ПРОГРАМНОГО ПРОДУКТУ

kyiv-flats-api/api/views.py

```
from rest_framework.decorators import (
    api_view,
    permission_classes,
)
from rest_framework.response import Response
from rest_framework.reverse import reverse
from rest_framework.permissions import AllowAny

@api_view(['GET'])
@permission_classes([AllowAny])
def api_root(request, format=None):
    """Using this api you can:
    * Create account
    * Log in if you already have an account
    * Evaluate your flat
    """
    return Response({
        'users': reverse('user-list-api', request=request, format=format),
        'flats': reverse('flat-list-api', request=request, format=format),
    })
```

kyiv-flats-api/api/permissions.py

```
from rest_framework.permissions import BasePermission
```

```
class IsOwnerOfFlat(BasePermission):
    message = 'You must be the owner of this object.'

    def has_object_permission(self, request, view, obj):
        return obj.user == request.user or request.user.is_staff

class IsOwnerOfProfile(BasePermission):
    message = 'You must be the owner of this object.'

    def has_object_permission(self, request, view, obj):
        return obj == request.user or request.user.is_staff
```

kyiv-flats-api/api/admin.py

"""Admin interface"""

```
from django.contrib import admin
from api.flats.models import Flat

class FlatModelAdmin(admin.ModelAdmin):
    list_display = ["user", "floor", "floors_count",
                   "total_square_meters", "center_distance",
                   "center_distance", "predicted_price_total"]
```

```

search_fields = ["user"]

class Meta:
    model = Flat

admin.site.register(Flat, FlatModelAdmin)

```

kyiv-flats-api/api/flats/views.py

```

from rest_framework.generics import (
    RetrieveUpdateAPIView,
    ListCreateAPIView,
)
from .models import Flat
from rest_framework.permissions import AllowAny, IsAuthenticated
from .serializers import FlatSerializer
from api.permissions import IsOwnerOfFlat
from . import services

```

```
class FlatListCreateAPIView(ListCreateAPIView):
```

```
    """This view allows to Retrieve all your flats
    or to add a new one.
```

Data requirements:

* flat should be located not farther then 20 km from Kyiv city center;

```

* "floors count" >= 4 and <= 48
* "floor" >= 1 and <= 48
* "total square meters" >= 20 and <= 200
"""

serializer_class = FlatSerializer
permission_classes = [IsAuthenticated]

def get_queryset(self):
    if self.request.user.is_staff:
        queryset = Flat.objects.all()
    else:
        queryset = Flat.objects.filter(user=self.request.user)

    return queryset

def perform_create(self, serializer):
    create_update(self, serializer)

class FlatDetailAPIView(RetrieveUpdateAPIView):
    """This view allows to Retrieve and Update this flat.

Data requirements:
* flat should be located not farther then 20 km from Kyiv city center;
* "floors count" >= 4 and <= 48
* "floor" >= 1 and <= 48
* "total square meters" >= 20 and <= 200

```

```
"""

queryset = Flat.objects.all()
serializer_class = FlatSerializer
permission_classes = [IsOwnerOfFlat]

def perform_update(self, serializer):
    create_update(self, serializer)

def create_update(self, serializer):
    latitude = serializer.validated_data['latitude']
    longitude = serializer.validated_data['longitude']
    total_square_meters = serializer.validated_data['total_square_meters']
    center_distance = services.get_center_distance(latitude, longitude)
    metro_distance = services.get_metro_distance(latitude, longitude)
    azimuth = services.get_azimuth(latitude, longitude)
    data = {
        'floor': serializer.validated_data['floor'],
        'floors_count': serializer.validated_data['floors_count'],
        'total_square_meters': total_square_meters,
        'latitude': latitude,
        'longitude': longitude,
        'center_distance': center_distance,
        'metro_distance': metro_distance,
        'azimuth': azimuth,
    }
    predicted_price_total = services.predict_price(data)
    predicted_price_metr = int(predicted_price_total/total_square_meters)
```

```
serializer.save(  
    user=self.request.user,  
    center_distance=center_distance,  
    metro_distance=metro_distance,  
    azimuth=azimuth,  
    predicted_price_total=predicted_price_total,  
    predicted_price_metr=predicted_price_metr,  
)
```

kyiv-flats-api/api/flats/urls.py

```
from django.urls import path  
from .views import FlatListCreateAPIView, FlatDetailAPIView  
  
urlpatterns = [  
    path("", FlatListCreateAPIView.as_view(), name='flat-list-api'),  
    path('<int:pk>/', FlatDetailAPIView.as_view(), name='flat-api-detail'),  
]
```

kyiv-flats-api/api/flats/services.py

```
import os  
import pickle  
import pandas as pd  
from geopy.distance import geodesic  
import mpmath as math
```

```
metro_path = os.path.join(os.path.dirname(os.path.realpath(__file__)),  
'../../static/data/kyiv_metro_stations.csv')  
metro_stations = pd.read_csv(metro_path)  
  
rf_path = os.path.join(os.path.dirname(os.path.realpath(__file__)),  
'../../static/data/rf_model.sav')  
loaded_rf_model = pickle.load(open(rf_path, 'rb'))  
xgb_path = os.path.join(os.path.dirname(os.path.realpath(__file__)),  
'../../static/data/xgb_model.sav')  
loaded_xgb_model = pickle.load(open(xgb_path, 'rb'))  
  
def get_azimuth(latitude, longitude):  
    city_center_coordinates = [50.450075, 30.524205]  
    rad = 6372795  
  
    llat1 = city_center_coordinates[0]  
    llon1 = city_center_coordinates[1]  
    llat2 = latitude  
    llon2 = longitude  
  
    lat1 = llat1 * math.pi / 180.  
    lat2 = llat2 * math.pi / 180.  
    long1 = llon1 * math.pi / 180.  
    long2 = llon2 * math.pi / 180.  
  
    cl1 = math.cos(lat1)  
    cl2 = math.cos(lat2)
```

```

sl1 = math.sin(lat1)
sl2 = math.sin(lat2)
delta = long2 - long1
cdelta = math.cos(delta)
sdelta = math.sin(delta)

y = math.sqrt(math.power(cl2 * sdelta, 2) + math.power(cl1 * sl2 - sl1 * cl2 * cdelta,
2))
x = sl1 * sl2 + cl1 * cl2 * cdelta
ad = math.atan2(y, x)

x = (cl1 * sl2) - (sl1 * cl2 * cdelta)
y = sdelta * cl2
z = math.degrees(math.atan(-y / x))

if (x < 0):
    z = z + 180.

z2 = (z + 180.) % 360. - 180.
z2 = - math.radians(z2)
anglerad2 = z2 - ((2 * math.pi) * math.floor((z2 / (2 * math.pi))))
angledeg = (anglerad2 * 180.) / math.pi

return round(angledeg, 2)

def get_metro_distance(latitude, longitude):
    """Returns distance to the nearest metro station."""

```

```
flat_coordinates = [latitude, longitude]
distances = list(map(lambda x, y: geodesic(flat_coordinates, [x, y]).meters,
                     metro_stations['latitude'], metro_stations['longitude']))

metro_distance = min(distances)

return int(metro_distance)

def get_center_distance(latitude, longitude):
    city_center_coordinates = [50.450075, 30.524205]
    center_distance = geodesic(city_center_coordinates, [latitude, longitude]).meters

    return int(center_distance)

def predict_price(data):
    flat = pd.DataFrame({
        'floor': [data['floor']],
        'floors_count': [data['floors_count']],
        'total_square_meters': [data['total_square_meters']],
        'center_distance': [data['center_distance']],
        'metro_distance': [data['metro_distance']],
        'azimuth': [data['azimuth']],
    })

    rf_prediction_price_metr = loaded_rf_model.predict(flat).round(2)
```

```

xgb_prediction_price_metr = loaded_xgb_model.predict(flat).round(2)

rf_total_price = rf_prediction_price_metr * flat['total_square_meters'][0]
xgb_total_price = xgb_prediction_price_metr * flat['total_square_meters'][0]

avg_total_price = (rf_total_price * 0.5 + xgb_total_price * 0.5)

return int(avg_total_price)

```

kyiv-flats-api/api/flats/serializers.py

```

from rest_framework.serializers import (
    ModelSerializer,
    ReadOnlyField,
    ValidationError,
    HyperlinkedIdentityField,
)

from .models import Flat
from . import services

class FlatSerializer(ModelSerializer):
    user = ReadOnlyField(source='user.username')
    edit_flat = HyperlinkedIdentityField(view_name='flat-api-detail')

    class Meta:
        model = Flat
        fields = [

```

```

'edit_flat',
'user',
'floor',
'floors_count',
'total_square_meters',
'latitude',
'longitude',
'center_distance',
'metro_distance',
'predicted_price_metr',
'predicted_price_total',

]

extra_kwargs = {
    'center_distance': {'read_only': True},
    'metro_distance': {'read_only': True},
    'predicted_price_metr': {'read_only': True},
    'predicted_price_total': {'read_only': True},
}

def validate(self, attrs):
    """Check that the center distance not longer than 20 km."""
    latitude = attrs['latitude']
    longitude = attrs['longitude']
    center_distance = services.get_center_distance(latitude, longitude)

    if center_distance > 20000:
        raise ValidationError("Flat must be located no further than 20 km from Kyiv
city center.")

```

```
    return attrs
```

kyiv-flats-api/api/flats/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxValueValidator, MinValueValidator

class Flat(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    floor = models.PositiveIntegerField(validators=[MinValueValidator(1),
                                                    MaxValueValidator(48)])
    floors_count = models.PositiveIntegerField(validators=[MinValueValidator(4),
                                                       MaxValueValidator(48)])
    total_square_meters = models.FloatField(validators=[MinValueValidator(20),
                                                       MaxValueValidator(200)])
    latitude = models.FloatField()
    longitude = models.FloatField()
    center_distance = models.PositiveIntegerField(null=True,
                                                   validators=[MaxValueValidator(20000)])
    metro_distance = models.PositiveIntegerField(null=True)
    azimuth = models.PositiveIntegerField(null=True,
                                         validators=[MinValueValidator(0), MaxValueValidator(360)])
    predicted_price_metr = models.PositiveIntegerField(null=True)
    predicted_price_total = models.PositiveIntegerField(null=True)
```