Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК *На правах рукопису*

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “Інтелектуальна система оцінки нерухомості на базі машинного навчання”

(назва згідно з наказом ректора)

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ

(позначення)

Студент

ІПЗ-42 **Владислав ОЛЕЙНИКОВ**

(шифр групи)(підпис) (дата)(розшифровка підпису)

Науковий керівник

д.т.н., доц. **Олексій БИЧКОВ**

(посада) (підпис) (дата)(розшифровка підпису)

Консультант з питань нормконтролю

Фах. **Тамара ЧАПОВСЬКА**

(посада) (підпис) (дата)(розшифровка підпису)

Допускається до захисту

Завідувач кафедри

д.т.н., доц. **Олексій БИЧКОВ**

(посада) (підпис) (дата)(розшифровка підпису)

Київ-2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра програмних систем і технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій

_____ Олексій БИЧКОВ
(підпис) (прізвище та ініціали)

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
СТУДЕНТУ

Олейникову Владиславу Юрійовичу
(прізвище, ім’я, по-батькові)

1. Тема бакалаврської роботи “Інтелектуальна система оцінки нерухомості на базі машинного навчання ”

керівник проекту (роботи) Бичков Олексій Сергійович, д.т.н., доцент
затверджені наказом вищого навчального закладу від **“** 11 ” листопада 2020 р. № 6

2. Строк здачі студентом закінченої роботи 10 травня 2021 р.

3. Вихідні дані до проекту (роботи) Алгоритми та основні концепції методів машинного навчання та аналізу великого обсягу даних

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз основних концепцій машинного навчання та огляд конкурентів
2. Дослідження методів амсамлевого машинного навчання
3. Отримання даних для аналізу
4. Розробка програмного забезпечення та налаштування інструментів

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

1. Матриця кореляції характеристик квартири (HeatMap) (Рис. 1, ст. 32)
2. Графік зміни точності розрахунків «випадкового лісу» в залежності від кількості дерев (рис. 2, ст. 37)
3. Графік зміни точності розрахунків «випадкового лісу» в залежності від кількості ознак для розщеплення (рис. 3, ст. 38)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	О.С. Бичков		
2	О.С. Бичков		
3	О.С. Бичков		
4	О.С. Бичков		

7. **Дата видачі завдання** 14 листопада 2020 р.

Керівник _____ (Олексій БИЧКОВ)

Завдання прийняв до виконання ____ _____ (Владислав ОЛЕЙНИКОВ)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	**Назви етапів бакалаврської роботи**	**Строк виконання етапів роботи**	**Примітка**
1	Підбір і вивчення літератури	03.12.2020	виконано
2	Аналіз концепцій та алгоритмів машинного навчання	17.12.2020	виконано
3	Дослідження методів ансамлевого машинного навчання	28.01.2021	виконано
4	Первинна підготовка даних	12.02.2021	виконано
5	Навчання моделей під наявний датасет	13.03.2021	виконано
6	Оптимізація гіпер-параметрів	29.03.2021	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	10.04.2021	виконано

Студент – бакалавр _____(Владислав ОЛЕЙНИКОВ)

Керівник роботи _____(Олексій БИЧКОВ)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 51 с., 7 рис., 1 додат., 15 джерел.

Тема: Інтелектуальна система оцінки нерухомості на базі машинного навчання.

Об’єкт дослідження: методи підготовки та аналізу даних.

Мета роботи: розробка систему для передбачення цін на об’єкти нерухомості.

Предмет дослідження: сервіси для автоматичної оцінки вартості об’єктів нерухомості.

Результати дослідження:

Досліджено методи попереднього аналізу та покращення якості даних для аналізу та основних алгоритмів та підходів до аналізу даних у прикладній області. Створено веб-додаток з інтерактивним інтерфейсом для зручної візуалізації результатів дослідження.

Висновок

В результаті роботи було створено навчені моделі аналізу даних. Також в результаті проведеної роботи було створено повноцінний сервіс для аналізу об’єктів нерухомості. Створена система проходить тестування впровадження на підприємство.

МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ, КЛАСТЕР, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ВИПАДКОВИЙ ЛІС, ПІДГОТОВКА ДАНИХ, ОБ’ЄКТИ НЕРУХОМОСТІ, БУСТИНГ, МАСИВИ ДАНИХ

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 51 с., 7 рис., 1 доп., 15 источников.

Тема: Интеллектуальная система оценки недвижимости на базе машинного обучения.

Объект исследования: методы подготовки и анализа данных**.**

Цель работы: разработка системы для предсказания цен на объекты недвижимости.

Предмет исследования: сервисы для автоматической оценки стоимости объектов недвижимости.

Результаты исследования:

Исследованы методы предварительного анализа и улучшения качества данных для анализа и основных алгоритмов и подходов к анализу данных в прикладной области. Создано веб-приложение с интерактивным интерфейсом для удобной визуализации результатов исследования.

Вывод

В результате работы было создано обученные модели анализа данных. Также в результате проведенной работы была создана полноценный сервис для анализа объектов недвижимости. Созданная система проходит тестирование внедрения на предприятие.

МАШИННОЕ ОБУЧЕНИЕ, КЛАССИФИКАЦИЯ, КЛАСТЕР, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, СЛУЧАЙНЫЙ ЛЕС, ПОДГОТОВКА ДАННЫХ, ОБЪЕКТЫ НЕДВИЖИМОСТИ, БУСТИНГ, МАССИВЫ ДАННЫХ

SUMMARY

Final qualifying bachelor's thesis: 51 pp., 7 figs.,1 appendices, 15 sources.

Topic: Intelligent real estate appraisal system based on machine learning

Object of research: methods of data preparation and analysis.

Purpose: to develop a system for predicting real estate prices.

Subject of research: services for automatic valuation of real estate.

Results of the research: Methods of preliminary analysis and data quality improvement for analysis and basic algorithms and approaches to data analysis in the application area are studied. A web application with an interactive interface for easy visualization of research results has been created.

Conclusion

As a result, trained data analysis models were created. Also, as a result of this work, a full-fledged service for the analysis of real estate was created. The created system is tested for implementation at the enterprise.

MACHINE LEARNING, CLASSIFICATION, CLUSTER, INTELLECTUAL SYSTEM, RANDOM FOREST, DATA PREPARATION, REAL ESTATE OBJECTS, BUSTING, DATA ARRAYS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	-	база даних
Датасет	-	набір даних
ІС	-	інтелектуальна система
ML	-	машинне навчання
ОбН	-	об’єкт нерухомості
Фічі	-	(у ML) параметри над якими проводиться аналіз
ПЗ	-	програмне забезпечення
Гіпер-параметри	-	параметри для моделей машинного навчання

ВСТУП

Актуальність теми

Оцінка нерухомості залишається одним з найбільш популярних видів оціночної діяльності, яка зазвичай включає в себе розрахунок вартості об'єкта або окремих форм взаємодії з об’єктом, як наприклад оренда.

Найчастіше, вартість нерухомості є очевидною для її власників або потенційних покупців. Однак, дана оцінка часто буває не об'єктивною та зменшує можливість ефективно розпоряджатися наявно. власністю. Власники нерухомості досить часто неефективно керують своєю власністю, а у процесі

продажу визначають її ринкову ціну самостійно. Дані проблеми негативно впливають на прибутковість і в кінцевому підсумку на вартість нерухомості в цілому. Також, процес оцінки нерухомості є складним та займає багато часу.

Мета і задачі дослідження

Метою бакалаврської роботи є розроблення інформаційної системи, що забезпечує процес автоматичної оцінки нерухомості та оптимізує процес аналізу наявного ринку житлових об’єктів.

Кінцевий результат роботи повинен включати в себе навчанні моделі оцінки на основі машинного навчання, використанні алгоритми формування моделі з можливістю повторного навчання, методи підготовки даних та методи покращення параметрів для використаних моделей машинного навчання.

Досягнення мети включало розв’язання таких **задач**:

- огляд існуючих систем оцінки нерухомості, тобто огляд конкурентів;
- попередня підготовки даних для аналізу;
- визначення ефективних методів та алгоритмів аналізу даних;
- реалізація сервісу для формування та використання нетренованої моделі оцінки нерухомості;

Об’єктом дослідження є методи підготовки та аналізу великих масивів даних.

Предметом дослідження є сервіси для автоматичної оцінки вартості об’єктів нерухомості.

Методи дослідження

Для створення моделі машинного навчання було досліджено навяні сервіси оцінки нерухомості. Також було досліджено типові методи підготовки та аналізу даних. Для навчання моделей було використанно ансамлеві методи машинного навчання, дерева пошуку рішень, градієнтний бустинг, кластеризація k-mean.

Наукова новизна отриманих результатів

Досліджено методи машинного навчання та комбінацію результатів різних моделей машинного навчання. У досліджуваній системі були використані удосконалені моделі машинного навчання задля підвищення якості результатів остаточної моделі.

Практичне значення одержаних результатів

В результаті виконаної роботи було отримано модель машинного навчання, яка використовує реальні об’єкти нерухомості на території України, тобто вона підлаштована під умови локального ринку, тобто їх можна використовувати при проведенні процесу оцінки нерухомості. Система готова до впровадження у роботу організації. Наразі проходить тестове впровадження у київському підприємстві.

Особистий внесок студента

Основним результатом є:

1. проведення процесу попередньої підготовки даних;

2. аналіз вхідних даних;

3. побудова моделей машинного навчання та поєднання результатів для підвищення точності розрахунків

Структура та обсяг роботи

Робота викладена на 51 сторінках друкованого тексту, який складається із вступу, чотирьох розділів, висновків, списку використаних джерел (15 найменування). Робота містить 7 рисунків та 1 додаток, обсягом 1 стор.

РОЗДІЛ 1

АНАЛІЗ ОСНОВНИХ КОНЦЕПЦІЙ МАШИННОГО НАВЧАННЯ ТА ОГЛЯД КОНКУРЕНТІВ

У наш час майже всі задачі, які пов'язані з оцінкою умовного об’єкту, класифікації, кластеризації або аналізу вирішують за допомогою машинного навчання. Тому, враховуючи сучасні темпи розвитку технології, можливості, які надають нам передові технології і знання, які генеруються щоденно у величезних розмірах, можна зрозуміти, що без машинного навчання більшість сфер сучасного світу може не отримувати ту ефективність, швидкість і точність, яку надає машинне навчання.

З основних причин, через які у даній роботі будуть використовуватися методи машинного навчання можна виділити такі, як:

- Популяризація машинного навчання у світі за рахунок наявності технологій, які можуть швидко обробляти і аналізувати величезні масиви даних
- Можливість знаходження нетривіальних зв’язків між параметрами об’єкту аналізу, про які неможливо дізнатися за допомогою класичних методів аналізу
- Ефективність сучасних методів машинного навчання. Деякі методи машинного навчання покращують результати роботи систем у десятки, а інколи і сотні разів
- Можливість автоматизувати процес і перекласти відповідальність за оцінку об’єкта нерухомості на обчислювальні технології

Ще одним важливим питанням, яке ми повинні розглянути – вибір конкретної методології машинного навчання. Машинне навчання має декілька основних типів методів, а саме:

- Класичне навчання
- Навчання з підкріпленням
- Ансамблеві методи навчання

- Нейронні мережі та глибоке навчання

Кожний з цих напрямів має конкретні цілі та підходи, що містять специфіку досліджуваної теми та вигляду бажаних результатів.

Класичне навчання поділяється на навчання без вчителя та з вчителем. Зазвичай навчання без вчителя використовують при кластеризації даних, пошуку правил у вибірці або зменшення розмірності.

При **кластеризації** модель сама визначає характеристики, за допомогою яких відбувається розділення на групи, тобто модель сама поділяє дані на групи(кластери).

Пошук правил є дуже подібний, за виключенням, що у даної методики є більш конкретна задача задля якої зазвичай відбувається пошук закономірностей.

Процес **зменшення розмірності** потрібний у випадках, коли нам потрібно зменшити розмір наших даних, але зберегти їх якість, тобто знайти такі характеристики, які зможуть узагальнити результати.

За допомогою навчання з вчителем вирішуються прості та стандартні задачі аналізу даних. Зазвичай навчання з вчителем поділяють на:

- **Класифікація**
- **Регресія**

Дані способи використовуються, коли у нас є конкретні параметри за якими поділяються дані(тобто класи на які хочемо розбити).

Навчання з підкріпленням використовують там, де завданням стоїть не аналіз даних, а виживання в реальному середовищі. Середовищем може бути навіть відеогра. Середовищем може бути реальний світ. Як приклад - автопілот Тесли, який вчиться правилам дорожнього руху та діяти у екстремальних ситуаціях на дорозі.

Знання про навколишній світ таким роботу можуть бути корисні, але чисто для довідки. Не важливо скільки даних він містить, у нього все одно не вийде передбачити всі ситуації. Тому його мета - мінімізувати помилки, а не розрахувати всі ходи. Робот вчиться виживати в просторі з максимальною вигодою: кількістю набраних балів у грі, часом поїздки в Теслі.

Виживання в середовищі і є ідея навчання з підкріпленням. Розумні моделі роботів-пилососів і самоврядні автомобілі навчаються саме так: їм створюють віртуальний місто (часто на основі карт справжніх міст), населяють випадковими пішоходами і відправляють вчитися.

Самою популярною на даний момент способом машинного навчання є нейронні мережі та глибоке навчання, яке використовується на великих об’ємах даних. Основні сфери використання

- Машинний переклад текстів
- Обробка зображень та відео
- Синтез та аналіз мови
- Знаходження об’єктів на фото
- Всі сфери з інших алгоритмів навчання

На жаль, для ефективного використання зазвичай потребує дуже великих об’ємів реальних даних на яких відбувається тренування моделі. Тому було прийнято рішення використання нейронних мереж було прийнято рішення використовувати ансамблеві методи машинного навчання.

Ансамблеві методи машинного навчання використовуються у:

- Пошукових системах
- Роботі з комп’ютерним бачення
- Сфери використання класичних алгоритмів (з більшою точністю)

Саме ці методи машинного навчання найбільш підходять для сфери нашого завдання, а саме оцінка об’єктів нерухомості.

Для кращого розуміння трендів та кращих рішень щодо оцінки вартості об’єкту нерухомості було проаналізовано портал Kaggle, а саме змагання на

оцінку вартості житла, а також сервіс Zillow, який вважається одним з найкращих по показникам оцінки житла в США.

Аналіз даних робіт показав, що існують не очевидні ознаки, що впливають на результати роботи алгоритм:

- Комбінації різних моделей показують різні результати в залежності від підходу до їх використання
- виникають значні труднощі з інтерпретацією різних ознак, а також з вибором метрик для них.

Завдання, яке вирішується в даній роботі, є головним завданням, що вирішуються на момент заснування найбільшого американський порталу про нерухомості Zillow.

Zillow - це онлайновий сервіс в сфері нерухомості в США, до якого прийшли його засновники Річ Бартон і Ллойд Фрінк після того, як вони провели безліч годин заносячи дані в таблицю, в якій порівнювали привабливість пропозицій, щоб після складних обчислень визначити цінність і переваги кожного об'єкта. Засновники прийшли до висновку, що мільйони людей стикаються з цим же завданням і терплять ті ж незручності при класифікації і впорядкування інформації, яку накопичують перш ніж здійснити будь-яку операцію на ринку нерухомості. Так і народилася ця ідея, наділити всіх бажаючих тією ж інформацією і правилами
визначення вартості нерухомості за певними параметрами,
що використовують у своїй роботі ріелтори.

ВИСНОВОК ДО РОЗДІЛУ 1

Отже, виходячи з проаналізованих джерел та аналізу конкурентів можна сказати, що для вирішення задачі оцінки об’єктів нерухомості зазвичай використовуються методи ансамлевого машинного навчання, класичні методи аналізу та, в деяких випадках, нейронні мережі. Також зазвичай використовують регресійний аналіз, а також класифікацію та кластеризацію. Дані методи можна об’єднувати для покращення результатів розрахунків.

Було прийнято рішення використовувати ансамблеві методи машинного навчання для створення програмного забезпечення, яке вирішує питання оцінки об’єкту нерухомості. Для цього необхідно провести дослідження основних алгоритмів, підходів та методиках які, використовуються у ансамлевих методах машинного навчання.

РОЗДІЛ 2

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЕВОГО МАШИННОГО НАВЧАННЯ

2.1 Беггінг

2.1.1 Загальні принципи і методологія

У паралельних методах ми розглядаємо різних учнів незалежно один від одного друга і, таким чином, можна навчати їх одночасно. Найбільш відомим з таких підходом є «беггінг» (від «bootstrap aggregation»), метою якого є створення ансамблевої моделі, яка є більш надійною, ніж окремі моделі, її складові.

Ідея беггінгу полягає в тому, що при відсутності великої навчальної вибірки можна створювати багато випадкових вибірок з вихідної простим вибором з заміщенням. У цьому випадку часто розглядають однорідних слабких учнів, навчають їх паралельно і незалежно один від одного, а потім об'єднують їх, слідуючи деякому детермінованому процесу усереднення. Хоча елементи в вибірках можуть перетинатися або дублюватися, на практиці все ж результати об'єднання з багатьох вибірок виявляється точніше, ніж тільки по одній початковій.

При навчанні моделі, незалежно від того, чи маємо ми справу з проблемою класифікації чи регресії, ми отримуємо функцію, яка приймає вхідні дані, повертає вихідні дані і визначається щодо навчального датасета. Через теоретичного розкиду навчального датасета підібрана модель також схильна до мінливості: якби спостерігався інший датасета, ми отримали б іншу модель.

Процес беггінгу має декілька етапів. Спочатку ми генеруємо кілька бутстреп вибірок так, щоб кожна нова бутстреп вибірка виконувала роль ще одного незалежного датасета, взятого з істинного розподілу. Потім ми можемо навчити слабкого учня для кожної з цих вибірок і агрегувати їх так, щоб ми усереднено подали їх результати і, таким чином, отримали модель ансамблю з розкидом меншим, ніж її окремі компоненти. Так як бутстреп вибірки є

приблизно незалежними і однаково розподіленими, те ж саме стосується і навчених слабких учнів. Потім усереднення результатів слабких учнів не змінює очікуваний відповідь, але зменшує його розкид (так само, як усереднення незалежних однаково розподілених випадкових величин зберігає очікуване значення, але зменшує розкид).

Отже, припустимо, що у нас є L бутстрап вибірок (апроксимації L незалежних датасета) розміру B. Це позначається:

$\{z_1^1, z_2^1, ..., z_B^1\}, \{z_1^2, z_2^2, ..., z_B^2\}, ..., \{z_1^L, z_2^L, ..., z_B^L\}$ $z_b^l \equiv b\text{-th observation of the } l\text{-th bootstrap sample}$

А потім об'єднаємо їх деяким процесом усереднення, щоб отримати модель ансамблю з меншим розкидом.

Існує кілька можливих способів об'єднати кілька моделей, навчених паралельно. Для завдання регресії вихідні дані окремих моделей можуть бути буквально усереднені для отримання вихідних даних моделі ансамблю. Для завдання класифікації клас, що передбачається кожною моделлю, можна розглядати як голос, а клас, який отримує більшість голосів, є відповіддю моделі ансамблю (це називається мажоритарних голосуванням).

Що стосується завдання класифікації, ми також можемо розглянути ймовірності кожного класу, що передбачаються усіма моделями, усереднити ці ймовірності і зберегти клас з найвищою середньою ймовірністю (це називається м'яким голосуванням). Середні значення або голосу можуть бути простими або зваженими, якщо будуть використовуватися будь-які відповідні їм ваги.

Ще ми можемо згадати, що одним з великих переваг беггінга є його паралелізм. Оскільки різні моделі навчаються незалежно один від одного, при необхідності можуть використовуватися методи інтенсивного розпаралелювання.

2.1.2 Випадкові ліси

Дерева рішень є дуже популярними базовими моделями для ансамблевих методів. Сильні учні, що складаються з декількох дерев рішень, можна назвати «лісами». Дерева, що становлять ліс, можуть бути обрані або не глибокими (глибиною в кілька вузлів), або глибокими (глибиною в безліч вузлів, якщо не в повну глибину з усіма листами). Неглибокі дерева мають менший розкид, але більш високу зміщення, і тоді для них найкращим вибором стануть послідовні методи. Глибокі дерева, з іншого боку, мають низьке зміщення, але високий розкид і, таким чином, є гарним вибором для беггінга, який в основному спрямований на зменшення розкиду.

Випадковий ліс є яскравим прикладом методу беггінга, де глибокі дерева, навчені на бутстрап вибірках, об'єднуються для отримання результату з більш низьким розсіюванням. Проте, випадкові лісу також використовують інший прийом, щоб кілька навчених дерев були менш корельованими один з одним: при побудові кожного дерева замість вибору всіх ознак з датасета для генерації бутстрапа ми збираємо і зберігаємо тільки випадкове їх підмножина для побудови дерева (зазвичай однакове для всіх бутстреп вибірок).

Вибірка за ознаками дійсно призводить до того, що всі дерева не дивляться на одну і ту ж інформацію для прийняття своїх рішень і, таким чином, зменшують кореляцію між різними повертаються вихідними даними. Інша перевага вибірки за ознаками полягає в тому, що вона робить процес прийняття рішень більш стійким до відсутнім даними: значення спостереження (з навчальної датасета чи ні) з відсутніми даними можна відновлювати за допомогою регресії або класифікації на основі дерев, які враховують тільки ті ознаки, де дані не відсутні. Таким чином, алгоритм випадкового лісу поєднує в собі концепції беггінга і вибору підпростору випадкових об'єктів для створення більш стійких моделей.

2.2 Бустинг

У роботі 1984 року були представлені теоретичні основи PAC-моделі (Probably Approximately Correct) ймовірнісно приблизно коректне навчання, при якому розглядає можливість поліпшити алгоритм класифікації та регресії за допомогою декількох слабких класифікаторів та регресорів.

У 1989 році Шапіро (Shapire) першим придумав такий алгоритм з поліноміальною складністю і опублікував в статті з яскравою назвою «Сила слабкого навчання», а через рік Фрейнд (Freund) розробив більш ефективну реалізацію, яка стала основою алгоритму AdaBoost, представленого в 1995 році Шапіро та Фрейнд вже разом.

Методи бустинга працюють в тому ж дусі, що і методи беггінга: ми створюємо сімейство моделей, які об'єднуються, щоб отримати сильного учня, який краще працює. Однак, на відміну від беггінга, яке в основному направлено на зменшення розкиду, бустинг - це метод, який полягає в тому, щоб адаптувати послідовно декількох слабких учнів адаптивним способом: кожна модель в послідовності підбирається, що надає великої ваги об'єктів в датасета, які погано оброблялися попередніми моделями в послідовності. Інтуїтивно, кожна нова модель фокусує свої зусилля на найбільш складних об'єктах вибірки при навчанні попередніх моделей, щоб ми отримали в кінці процесу сильного учня з більш низьким зсувом (навіть якщо вийде так, що бустинг буде при цьому зменшувати розкид). Бустинг, як і беггінг, може використовуватися як для задач регресії, так і для класифікації.

Базові моделі, які часто розглядаються для бустинга - це моделі з низьким розкидом, але з високим зміщенням. Наприклад, якщо ми хочемо використовувати дерева рішень в якості наших базових моделей, в основному ми будемо вибирати неглибокі дерева рішень з глибиною в кілька вузлів. Інша важлива причина, яка мотивує використовувати моделі з низьким розкидом, але з високим зміщенням в якості слабких учнів для бустинга, полягає в тому, що ці

моделі, як правило, вимагають менших обчислювальних витрат (кілька ступенів свободи при підборі гіпер-параметрів). Дійсно, оскільки обчислення для підгонки до різних моделей не можуть виконуватися паралельно (на відміну від беггінга), це може стати занадто дорогим для послідовного підбору декількох складних моделей.

Після того, як слабша половина учнів обрані, нам все ще потрібно визначити, як вони будуть послідовно підганяти І як вони будуть агрегуватися. Для цього існують два важливих алгоритму бустинга: adaboost (адаптивний бустинг) і градієнтний бустинг.

Ці два мета-алгоритму відрізняються тим, як вони створюють і об'єднують слабких учнів в ході послідовного процесу. Адаптивний бустинг оновлює ваги, прикріплені до кожного з об'єктів навчального датасета, тоді як градієнтний бустинг оновлює значення цих об'єктів. Ця різниця виходить з того, що обидва методи намагаються вирішити задачу оптимізації, яка полягає в пошуку найкращої моделі, яка може бути записана у вигляді зваженої суми слабких учнів.

Протягом останніх 10 років бустинг залишається одним з найбільш популярних методів машинного навчання поряд з нейронними мережами і машинами опорних векторів. Основні причини простота, універсальність, гнучкість (можливість побудови різних модифікацій), і, головне, висока узагальнююча здатність.

Бустинг над вирішальними деревами вважається одним з найбільш ефективних методів з точки зору якості класифікації. У багатьох експериментах спостерігалося практично необмежене зменшення частоти помилок на незалежної тестової вибірці в міру нарощування композиції. Більш того, якість на тестовій вибірці часто продовжували поліпшуватися навіть після досягнення безпомилкового розпізнавання всієї навчальної вибірки. Це змінило існуючий довгий час уявлення про те, що для підвищення узагальнюючої здатності необхідно обмежувати складність алгоритмів.

На прикладі бустинга стало зрозуміло, що гарною якістю можуть мати як завгодно складні композиції, якщо їх правильно налаштовувати. Згодом феномен бустинга отримав теоретичне обгрунтування. Виявилося, що зважене голосування не збільшує ефективну складність алгоритму, а лише згладжує відповіді базових алгоритмів. Кількісні оцінки узагальнюючої здатності бустинга формулюються в термінах відступу. Ефективність бустинга пояснюється тим, що в міру додавання базових алгоритмів збільшуються відступи навчальних об'єктів. Причому бустинг продовжує розсовувати класи навіть після досягнення безпомилкової класифікації навчальної вибірки.

2.3 Стекінг

Стековое узагальнення (stacked generalization), або просто стекінг (stacking) ще один спосіб об'єднання класифікаторів та регресійних моделей, вводить поняття мета-алгоритму навчання. На відміну від беггінга і бустінга, при стекінг використовуються класифікатори різної природи.

Ідея стекінгу полягає в тому, щоб вивчити декількох різних слабких учнів і об'єднати їх, навчивши метамодель для виведення передбачень, заснованих на множинних прогнозах, що повертаються цими слабкими моделями. Отже, нам потрібно визначити дві речі для побудови нашої моделі стека: L учнів, яких ми хочемо навчити, і метамодель, яка їх об'єднує.

Наприклад, для завдання класифікації ми можемо в якості слабкого учня вибрати класифікатор KNN, логістичну регресію і SVM і прийняти рішення навчити нейронну мережу в якості метамоделі. Потім нейронна мережа прийме в якості вхідних даних результати трьох наших слабких учнів і навчиться давати остаточні прогнози на їх основі.

Отже, припустимо, що ми хочемо навчити стековий ансамбль, який складався з L слабких учнів. Потім ми повинні виконати наступні кроки:

- розділити тренувальні дані на дві частини
- виберіть L слабких учнів і навчіть їх на даних першого фолда (частини)
- для кожного з L слабких учнів зробіть прогнози для об'єктів з другого фолда
- навчити метамодель вдруге, використовуючи в якості вхідних даних прогнози, зроблені слабкими учнями

В даному методі ансамблевого навчання було поділено датасет на дві частини, тому що прогнози даних, які використовувалися для навчання слабких учнів, не мають відношення до навчання метамоделі.

Таким чином, очевидним недоліком цього поділу нашого датасета на дві частини є те, що у нас є тільки половина даних для навчання базових моделей і половина даних для навчання метамоделі. Щоб подолати це обмеження, ми можемо, однак, дотримуватися певного підходу «k-fold крос-навчання», таким чином всі об'єкти можуть бути використані для навчання мета-моделі: для будь-якого об'єкта передбачення слабких учнів робиться на прикладах цих слабких учнів, навчених на k-1 фолдах, які не містять даного об'єкту. Іншими словами, він навчається по k-1 фолду, щоб робити прогнози для залишився об'єктів в будь-яких “фолдах”. Таким чином, ми можемо створити відповідні прогнози для кожного об'єкта нашого датасета, а потім навчити нашу метамодель всім цим прогнозам.

Стекінг має дві основні відмінності від беггінга і бустинга. По-перше, стекінг часто враховує різнорідних слабких учнів (комбінуються різні алгоритми навчання), тоді як беггінг і бустінг враховують в основному однорідних слабких учнів. По-друге, стекінг вчить об'єднувати базові моделі з використанням метамоделі, тоді як беггінг і бустінг об'єднують слабких учнів за допомогою детерміністичним алгоритмам.

ВИСНОВОК ДО РОЗДІЛУ 2

У результаті проведеного дослідження основних методів ансламлевого машинного навчання, отримано детальну характеристику кожного с підходів, бажанну область використання та способи імплементації та рекомендації щодо налаштування.

Всі з досліджених методів є ефективним у прикладній області використання, але для роботи з даними про об’єкти нерухомості було обрано підходи, що описувалися у розділі 2.1 та 2.2 через зручність їх використання, можливість швидкої оптимізації та економію часових ресурсів при проведенні повторних навчань моделей та проведення процесу оптимізації параметрів.

РОЗДІЛ 3
ОТРИМАННЯ ДАНИХ ДЛЯ АНАЛІЗУ

3.1 Опис зібраних даних

Після етапу вивчення теоретичних матеріалів відбувається процес попереднього аналізу проблеми. Так як для будь-якого методу машинного навчання потрібна велика кількість даних – було використано дані з дипломної роботи студента Гунько А.В. групи ІПЗ-42. А саме база даних, яка містить в собі «фічі» - тобто, параметри, які будуть використанні для навчання моделі.

Даний датасет містить дані про квартири по всій Україні, але до розгляду у даній роботі В перелік параметрів даної бази даних входять такі характеристики квартири, як:

- район міста
- підрайон міста
- належність до первинного ринку
- дата публікації
- дата кінця публікації
- тип санвузла
- рік побудови
- матеріал стін
- тип стін
- тип вікон
- тип опалення
- наявність та тип ремонту
- номер будинку
- найближче метро
- кількість кімнат
- загальна площа об’єкту

- житлова площа
- площа кухні
- поверх квартири
- поверховість будівлі
- ціна будівлі
- ціна за м²
- адрес будівлі, який перейшов у вигляд
    - довгота
    - широта
- дистанція до центру міста
- азимут(від точки центра Києва)

Азимут – горизонтальний кут між напрямком на північ та напрямком на обраний об’єкт.

Точкою центра Києва було обрано саме географічний центр міста, що знаходиться за координатами:

- широта - 50.4642
- довгота - 30.4665

Саме ці точки було обрано для коректної роботи алгоритму, так як для використання азимуту потрібно було обрано точку, яка є максимально централізованою відносно кордонів Києва і відносно якого можна буде класифікувати об’єкти Києва, у котрих комбінація азимута і дистанції є приблизно однаковою. Також у подальшому аналізі прийнято рішення залишити показники широти та довготи для порівняння їх з комбінацією азимута та відстані від центра.

3.2 Первинна обробка даних

Важливою частиною роботи з даними є відсічення неякісних параметрів, а також параметрів з відсутніми значеннями. Через специфіку ринку і сайтів, які аналізувалися для збору даних, велика кількість потенційно корисних характеристик не було наявно в будинках, що
було прибрано такі, як:

- тип санвузла
- рік побудови
- матеріал стін
- тип стін
- тип вікон
- тип опалення
- житлова площа
- наявність та тип ремонту
- номер будинку
- житлова площа
- найближче метро
- район міста
- підрайон міста

Тобто, після відкидання непотрібних параметрів і переведення даних про адресу у довготу і широту провелася фільтрація і пошук азимуту з дистанцією до центра .

Фундаментальними принципами роботи з даними для подальшого аналізу є їх нормалізація та відокремлення зайвої інформації. Задля цієї мети проводиться попередня обробка даних, фрагмент коду методу наведено нижче:

```
def preparingRawData():
# зчитування даних
    df : DataFrame = pd.read_csv("../../data.csv",encoding='utf-8')
    include = [
'url','rate','price','floor','total_floor','area','rooms','kitchen_area','lon','lat
','true_is_primary_market','published','published_end','locality']
#первинна обробка та нормалізація даних
clearAndSaveData(df,include'../../DataKiev.csv')
def clearAndSaveData(df: DataFrame, columns:list[str], savePath: str, dropNan =
True,dropDuplicates= True):
      df=  df[df.locality == 'Київ']
      df = (df-df.mean())/df.std()
      df = df[columns]
      df['azimuth'] = df.apply(lambda x:  get_azimuth(x['lat'],x['lon']), axis=1)
    df['distance'] = df.apply(lambda x:  geodesic([x['lat'],x['lon']],
city_center_coordinates).km, axis=1)
    print(df.isna().sum())
    if dropNan: df.dropna(inplace=True)
    if dropDuplicates: df.drop_duplicates(inplace=True)
    df.describe()
    df.to_csv(savePath)
```

ВИСНОВОК ДО РОЗДІЛУ 3

Після отримання даних для аналізу та проведення початкового аналізу та фільтрації даних. З 743 000 об’єктів нерухомості залишилось 220 000 по Києву, та в окремих етапах аналізу використовувалися дані, які розділенні за етапами існування публікації квартири та приналежності до первинного ринку нерухомості, що теж призводить до зменшення вибірки. В результаті даного етапу дані готові для створення моделей машинного навчання та проведення процесу оцінки об’єкта нерухомості.

РОЗДІЛ 4

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА НАЛАШТУВАННЯ ІНСТРУМЕНТІВ

4.1 Опис інструментів, які були використані

Для вирішення проблеми оцінки нерухомості було відібрано кілька популярних бібліотек для роботи з даними. Серед них:

- pandas
- numpy
- sklearn
- matplotlib
- XGBoost
- seaborn

Кожен з цих інструментів має конкретну ціль у ПЗ, що проектується.

Seaborn - бібліотека візуалізації даних Python, заснована на matplotlib. Він забезпечує інтерфейс високого рівня для малювання привабливої та інформативної статистичної графіки.

Спочатку було вирішено побудувати графік кореляції параметрів квартири за допомогою пакету seaborn.

```
features = [
      'azimuth',
      'distance',
      'floor',
      'total_floor',
      'area',
      'rooms',
      'kitchen_area',
      'lat',
      'lon'
]

X = df[features]
sns.heatmap(X.corr(),cbar=True,annot=True)
```

Дані для цього графіку та подальших розрахунків бралися з датасету, який на момент створення роботи містив в собі 20000 елементів (для Києва).

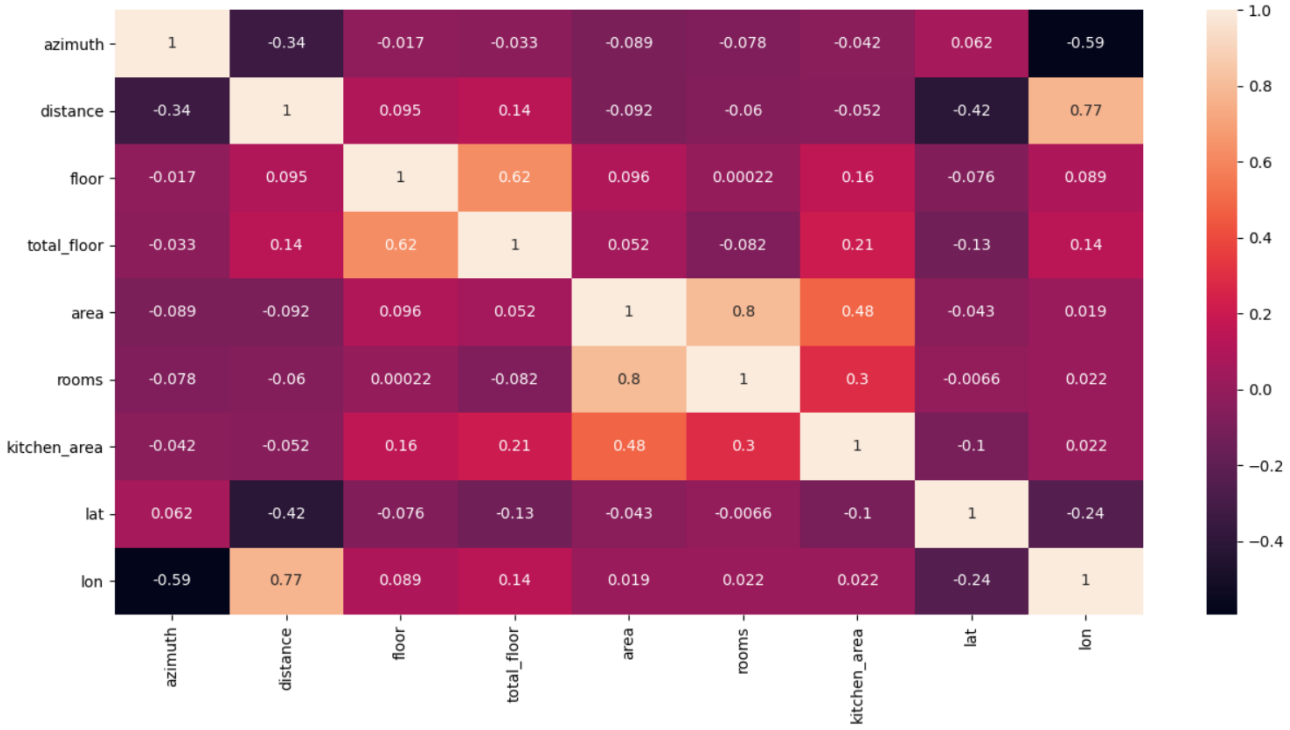

Рис. 1 Матриця кореляції характеристик квартири (HeatMap)

На даному графіку ми маємо можливість побачити як характеристики залежать від значень один одного. Але цікавою деталлю є те, що дані, що були штучно додані (distance - дистанція до центру міста, azimuth - азимут) взагалі не мають залежності з іншими параметрами. Якщо у рамках даної роботи проводити тільки регресійний аналіз початкових параметрів, то штучні параметри, очевидно, були би зайвими, але аналіз даних за допомогою «випадкового лісу» та XGBoost (екстримальний градієнтний регресійний аналіз) доводить помилковість попередніх висновків. Результати яких буде розглянуто у наступних розділах.

Pandas це високорівнева Python бібліотека для аналізу даних. Її називають високорівневою, тому що побудована вона поверх більш низкорівневої бібліотеки NumPy (написана на Ci), що є великим плюсом в продуктивності. В екосистемі Python, pandas є найбільш протянутою бібліотекою, яка швидко

розвивалася і призначена для обробки і аналізу даних. Тобто в конкретному випадку цей пакет був використаний для зчитування бази даних квартир та відкидання непотрібних, таких як викиди в регресійній моделі. Приклади використання пакету:

```
# Зчитування рядків датасету
df = pd.read_csv("dataKyivAnalize.csv", encoding='utf-8')

# Нормалізація певних показників
df['rate'] = df['rate'].round(0)
df['distance'] = df['distance'].round(0)
df['azimuth'] = df['azimuth'].round(0)

# Розрахунок квартилів
first_quartile = df.quantile(q=0.25)
third_quartile = df.quantile(q=0.75)
IQR = third_quartile - first_quartile

# Визначення викидів
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)

# Відсіювання 2500 викидів
outliers = outliers.head(2500)
df.drop(outliers.index, inplace=True)
```

В цьому прикладі пакет pandas було використано для відсіювання квартир, які знаходяться на відстані більш ніж 17 км від точки центру Києва, яку було обрано раніше, а також розраховано квартилі для роботи з «викидами».

NumPy - бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій з цими масивами. У нашому ПЗ даний пакет було використано у функціях розрахунку помилок.

Застосування пакету для математичних розрахунків:

```
# Обчислює середню абсолютну процентну помилку
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

# Обчислює медіанну абсолютну процентну помилку
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
```

Щоб оцінити якість прогнозів моделі, дані (датасета) випадковим чином ділять на навчальну і тестову вибірки. Зазвичай в співвідношенні 70/30, щоб зберегти частину прикладів в таємниці від алгоритму. Модель вчиться на першій вибірці, зіставляючи параметри квартир та їх ціни. Потім демонструємо їй характеристики квартир з тестової вибірки (але ціни з тестової не відображаємо, вона спробує вгадати їх сама). Останньою дією обчислюємо помилку, грубо кажучи, наскільки передбачена вартість відхиляється від фактичної.

Метрики якості потрібні, щоб порівнювати моделі між собою і розуміти, наскільки добре модель описує реальність. Для своїх досліджень я використовував дві метрики:

- **Середня абсолютна процентна помилка (MAPE)** - показує середнє арифметичне значення всіх абсолютних (взятих по модулю) процентних помилок прогнозу (Ф - фактичне значення вартості, П - прогнозне значення вартості). Розраховується за формулою:

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

- **Медіанна абсолютна процентна помилка (MedAPE)** - та ж ідея обчислення абсолютної відсоткової помилки, але замість середнього арифметичного беремо більш стійку до викидів медіану. Саме своєю

MedAPE в районі 2% хвалиться сайт https://www.zillow.com/, який побудував одну з найбільш точних моделей оцінки будинків в США.

Бібліотека **Scikit-learn** - найпоширеніший вибір для вирішення завдань класичного машинного навчання. Вона надає широкий вибір алгоритмів навчання з учителем і без вчителя. Одне з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх один з одним. Ще однією перевагою є широка спільнота і докладна документація. Scikit-learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання, для досліджень, а так само для новачків, які тільки робить перші кроки в області машинного навчання.

Даний пакет було використано для розрахування коефіцієнту детермінації нетренованої моделі та оптимізації вхідних параметрів для моделі навчання:

```
# Друкує розраховані значення коефіцієнта детермінації, середньої і медіанної абсолютних помилок
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('')
    print('Середня абсолютна помилка: {:.3} %'.format(mean_absolute_percentage_error(val_y, prediction)))
    print('Медіанна абсолютна помилка: {:.3} %'.format(median_absolute_percentage_error(val_y, prediction)))
```

Також, даний модуль надає змогу розбити наш датасет на дані для тренування та валідації нетренованої моделі. Розділення даних відбувається за стандартними налаштуваннями наявного інструменту, а саме 75% для тесту та 25% контрольних даних для валідації.

```
# Проводимо випадкове розбиття даних (train) і валідації (val), за замовчуванням в пропорції 0.75 / 0.25
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
```

4.2 Налаштування алгоритмів

Найважливішим модулем ПЗ для оцінювання вартості об’єктів нерухомості – є модуль, який втілює алгоритм машинного навчання. Для цієї задачі було обрано алгоритм «випадкового лісу», тобто набір дерев рішень, які зв допомогою своєї «густоти» забепечують одні з найкращих результатів машинного навчання навіть на не великих вибірках даних. Тому гарним рішенням було використання бібліотки sklearn, а саме реалізації алгоритму `RandomForestRegressor`.

```
# Створюємо регресійну модель випадкового лісу
rf_model = RandomForestRegressor(n_estimators=250,
                                 n_jobs=-1,
                                 bootstrap=True,
                                 criterion='mse',
                                 max_features=4,
                                 random_state=1,
                                 max_depth=16,
                                 min_samples_split=5
)

# Проводимо підгонку моделі на навчальній вибірці
rf_model.fit(train_X, train_y)

# Обчислюємо передбачені значення цін на основі валідаційної вибірки
rf_prediction = rf_model.predict(val_X).round(0)

# Обчислюємо і друкуємо величини помилок при порівнянні відомих цін квартир з валідаційної вибірки з передбаченими моделлю
print_metrics(rf_prediction, val_y)
```

Дуже важливим на даному етапі розробки і проектування є правильно налагодити параметри алгоритму навчання. Для цього потрібно детально ознайомитися с описом кожного параметра.

Число дерев - n_estimators

Чим більше дерев, тим краще якість, але час налаштування і роботи RF також пропорційно збільшуються. Зверніть увагу, що часто при збільшенні n estimators якість на навчальній вибірці підвищується (може навіть доходити до 100%), а якість на тесті виходить на асимптоту, тобто методом підбору можна побачити, яка кількість дерев буде оптимальна для вашої моделі.

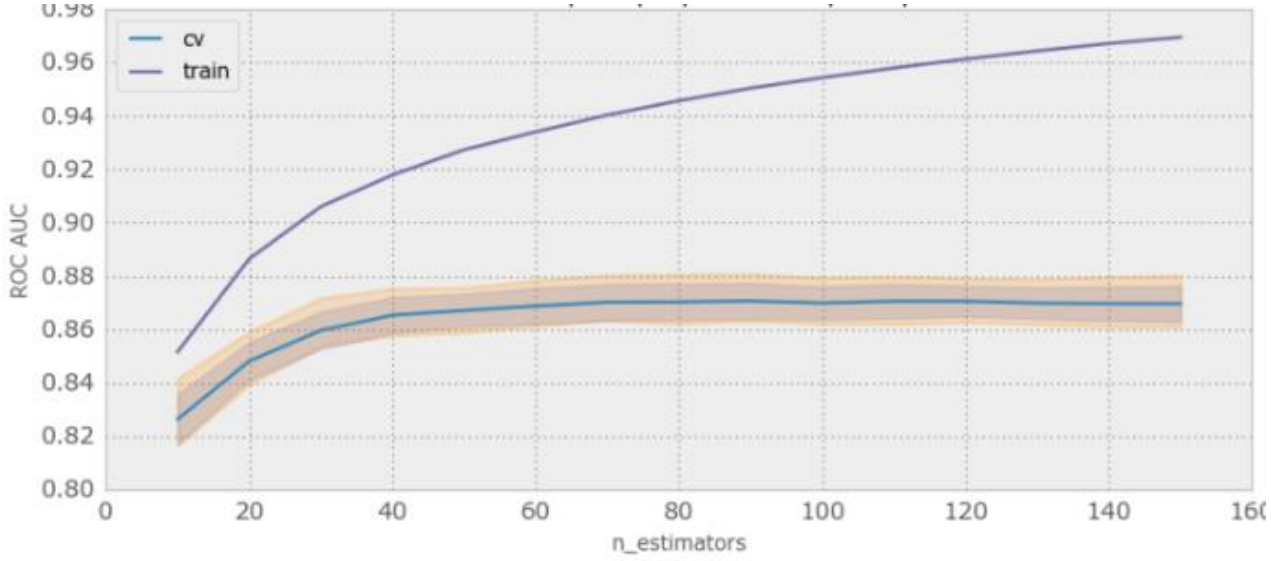

Рис.2 Графік зміни точності розрахунків «випадкового лісу» в залежності від кількості дерев

Число ознак вибору для розщеплення - max_features

Графік якості на тесті від значення цього параметра унімодальне, на навчанні він строго зростає. При збільшенні max_features збільшується час побудови лісу, а дерева стають «більш одноманітними». За замовчуванням він дорівнює sqrt (n) в задачах класифікації та n / 3 в задачах регресії. Це найважливіший параметр. Його налаштовують в першу чергу (при достатній кількості дерев у лісі). Інколи даний параметр може змінювати свою поведінку в залежності від параметрів, що будуть описані нижче, але взагалі таке відбувається не часто та на вибірках з великим обсягом параметів.

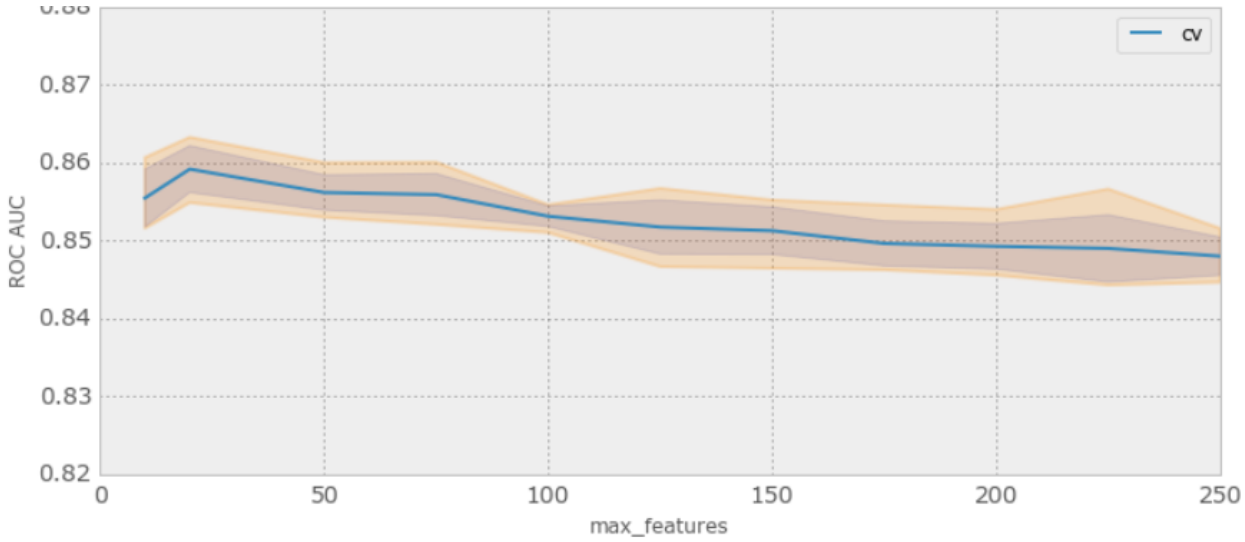

Рис.3 Графік зміни точності розрахунків «випадкового лісу» в залежності від кількості ознак для розщеплення

Мінімальна кількість об'єктів, при якому виконується розщеплення - min_samples_split

Цей параметр, як правило, не дуже важливий і можна залишити значення за замовчуванням. Графік якості на контролі може бути схожим на «гребінець». При збільшенні параметра якість на навчанні падає, а час побудови RF скорочується.

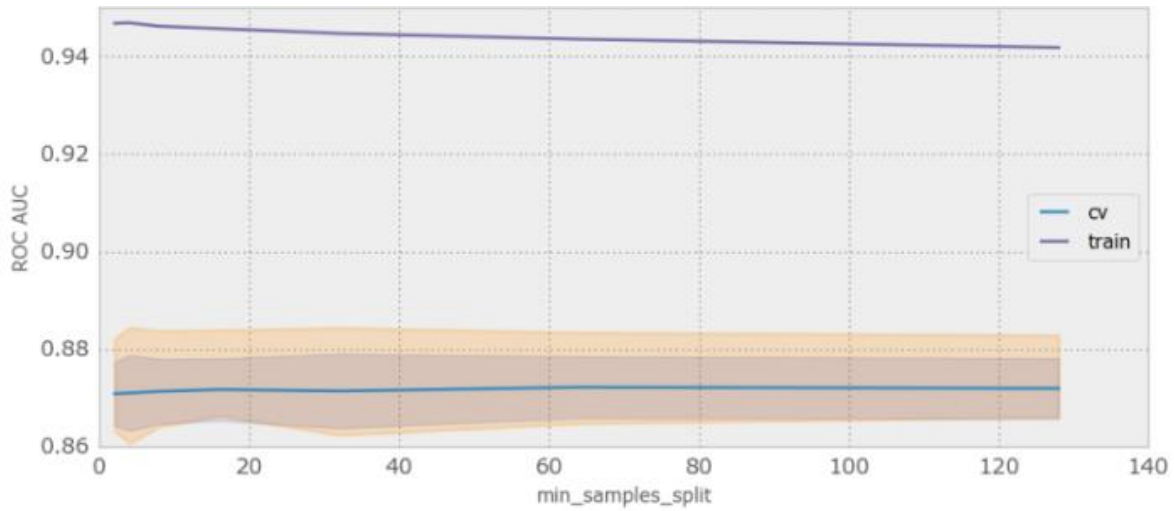

Рис.4 Графік зміни точності прорахунків «випадкового лісу» в залежності від мінімальної кількості об'єктів при якому виконується розщеплення

Обмеження на число об'єктів в листі - min_samples_leaf

Параметр, який частіш за все залишається без змін. У класичних рекомендація машинного навчання в задачах регресії рекомендується використовувати значення 5 (в бібліотеці randomForest для R так і реалізовано, в sklearn - 1).

Максимальна глибина дерев - max_depth

Чим менше глибина дерев, тим швидше будується і працює RF. При збільшенні глибини різко зростає якість на навчанні, а на контрольній виборці воно, як правило, збільшується. Рекомендується використовувати максимальну глибину (крім випадків, коли об'єктів занадто багато і виходять дуже глибокі дерева, побудова яких займає чимало часу). При використанні не глибоких дерев зміна параметрів, пов'язаних з обмеженням числа об'єктів в листі і для поділу, не призводить до істотного ефекту (листя і так виходять «великими»). Неглибокі дерева рекомендують використовувати в задачах з великим числом шумових об'єктів (викидів).

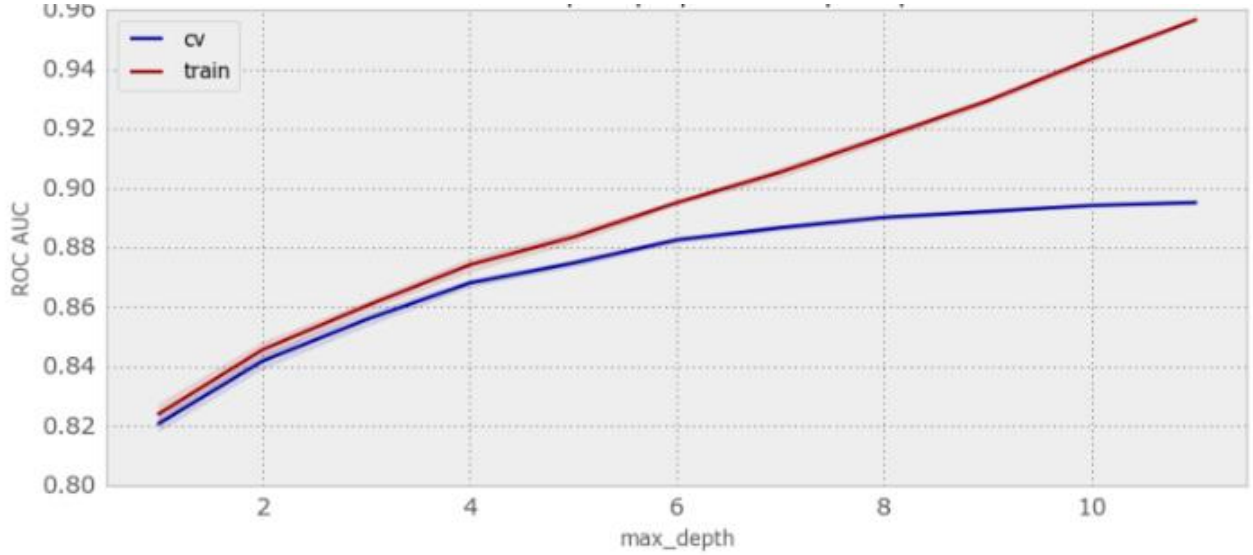

Рис.5 Графік зміни точності розрахунків «випадкового лісу» в залежності від максимальної глибини лісу

Критерій розщеплення - criterion

За змістом це дуже важливий параметр, але на даний момент не є найважливішим. У бібліотеці sklearn для регресії реалізовані два критерії: "mse" і "mae", відповідають функціям помилки, які вони мінімізують. У більшості завдань використовується mse. Порівняти їх поки не беруся, тому що mae з'явився зовсім недавно - у версії 0.18.

Беггінг – Bootstrap

Параметр зазвичай рекомендується до ввімкнення. Цей параметр відповідає за використання беггінгу у процесі навчання моделі.

Кількість задіяних процесорів - n_jobs

За замовчуванням в sklearn-івських методах n_jobs = 1, тобто випадковий ліс будується одному процесорі. Щоб істотно прискорити побудову, було використано n_jobs = -1 (будувати на максимально можливому числі процесорів).

4.3 Оптимізація основної моделі

Однією з особливістю проектуємого ПЗ є використання екстремального градієнтного бустинга, основна мета якого є поліпшення результатів основної моделі “випадкового лісу”.

XGBoost - алгоритм машинного навчання, заснований на дереві пошуку рішень і використовує фреймворк градієнтного бустинга. XGBoost і Gradient Boosting Machines (GBM) - ансамблі методів дерев, які використовують принцип бустинга слабких учнів (найчастіше, алгоритм побудови бінарного дерева рішень) за допомогою архітектури градієнтного спуску. У свою чергу, XGBoost - поліпшення фреймворка GBM через системну оптимізацію та удосконалення алгоритму, тому і було прийнято рішення використовувати його, як допоміжний алгоритм при проектуванні нашого ПЗ.

Для пошуку оптимальних параметрів було використано бібліотеку sklearn, модуль `sklearn.model_selection`. Даний модуль містить в собі кілька алгоритмів для проведення процесу Це процес виконання налаштування параметрів, або як з гіпер-параметрів з метою визначення оптимальних значень для даної моделі. Загалом, немає можливості заздалегідь дізнатися найкращі значення для гіпер-параметрів, тому в ідеалі нам потрібно спробувати всі можливі значення, щоб знати оптимальні значення.

Виконання цього мануально може зайняти значну кількість часу та ресурсів, і тому для автоматизації було використано функції GridSearchCV та RandomizedSearchCV.

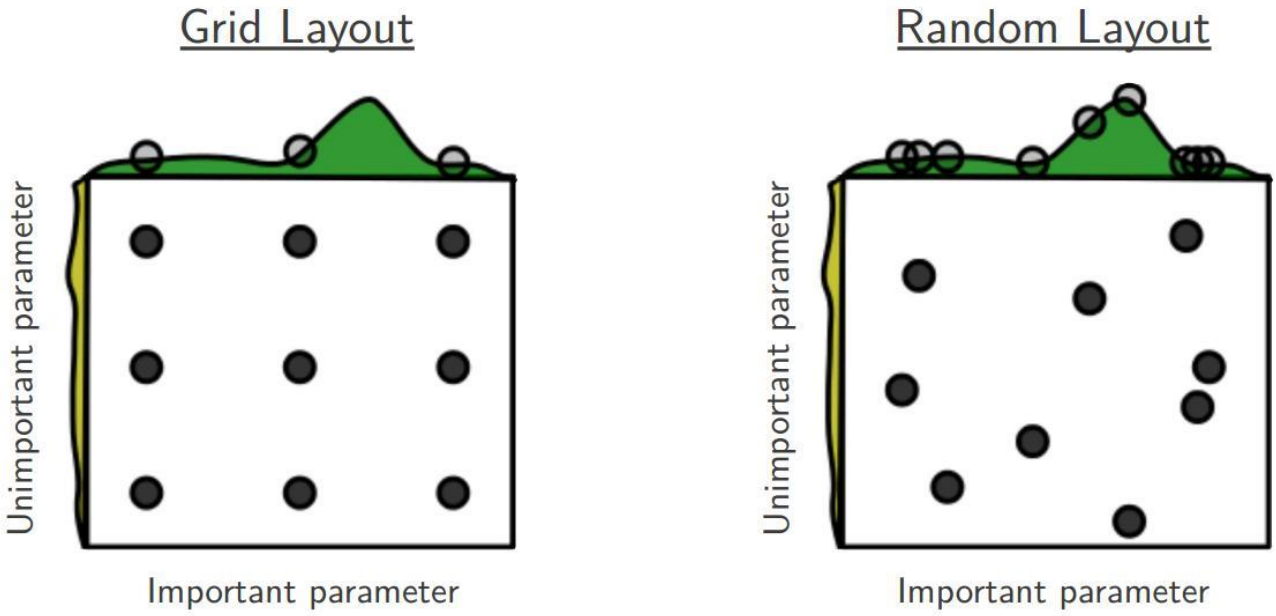

Рис.6 Графічне відображення роботи методів GridSearchCV та RandomizedSearchCV

GridSearchCV - це функція, яка входить до пакету Scikit-learn (або SK-learn) model_selection. Ця функція допомагає прокручувати заздалегідь визначені гіпер-параметри та підібрати оцінювач (модель) до навчального набору. Отже, зрештою, можливо обрати найкращі параметри з перерахованих гіпер-параметрів. Виконується це за рахунок словника, у якому ми згадуємо певний гіпер-параметр разом із значеннями, які він може приймати. Надамо приклад словника:

```
{ 'max_depth': [3, 5, 6, 10, 15, 20],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.5, 1.0, 0.1),
           'colsample_bytree': np.arange(0.4, 1.0, 0.1),
           'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
           'n_estimators': [100, 500, 1000]}
```

GridSearchCV тестує всі комбінації значень, переданих у словнику, і оцінює модель для кожної комбінації, використовуючи метод перехресної перевірки. Отже, після використання цієї функції ми отримуємо точність / втрату для кожної комбінації гіпер-параметрів, і ми можемо вибрати той, що має найкращі характеристики.

```
def tuneViaGridSearchCV(pathAnalizeData):
    data : DataFrame = pd.read_csv(pathAnalizeData,encoding='utf-8')
    y : DataFrame = data.rate
    X = data.drop(['rate'], axis=1).select_dtypes(exclude=['object'])

    data[data.columns.tolist()[-1]]
    params = { 'max_depth': [3, 5, 6, 10, 15, 20],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.5, 1.0, 0.1),
           'colsample_bytree': np.arange(0.4, 1.0, 0.1),
           'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
           'n_estimators': [100, 500, 1000]}

    xgbr = xgb.XGBRegressor(seed = 20,tree_method='gpu_hist', gpu_id=0)

    clf = GridSearchCV(estimator=xgbr,
                       param_grid=params,
                       scoring='neg_mean_squared_error',
                       n_jobs=3,
                       verbose=2)
    clf.fit(X, y)
    print("Best parameters:", clf.best_params_)
    print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```

RandomizedSearchCV - числовий метод оптимізації, які не вимагають градієнта для задачі оптимізації, тобто, RS може бути використаний для функцій, які не є безперервними або диференційованими. Такі методи оптимізації також відомі як методи прямого пошуку, без похідних чи чорної скриньки. Використовується, щоб зменшити кількість ітерацій та із випадковою комбінацією параметрів. Це корисно, коли у вас занадто багато параметрів, щоб спробувати, і ваш час навчання довший. Це допомагає зменшити вартість обчислень. Також це надало змогу за обмежений час перевірити багато комбінацій гіпер-параметрів, що було корисно при початковому аналізі різних частин датасету.

```
def tuneViaRandomizedSearchCV(pathAnalizeData):
    data : DataFrame = pd.read_csv(pathAnalizeData,encoding='utf-8')
    y : DataFrame = data.rate
    X = data.drop(['rate'], axis=1).select_dtypes(exclude=['object'])

    data[data.columns.tolist()[-1]]
    params = { 'max_depth': [3, 5, 6, 10, 15, 20],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.5, 1.0, 0.1),
           'colsample_bytree': np.arange(0.4, 1.0, 0.1),
           'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
           'n_estimators': [100, 500, 1000]}
    xgbr = xgb.XGBRegressor(seed = 20,tree_method='gpu_hist', gpu_id=0)
    clf = RandomizedSearchCV(estimator=xgbr,
                         param_distributions=params,
                         scoring='neg_mean_squared_error',
                         n_iter=25,
                         n_jobs=3,
                         verbose=2)
    clf.fit(X, y)
    print("Best parameters:", clf.best_params_)
    print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```

Результати даного модуля було використано для нашого модуля XGBoost

```
xgb_model = xgb.XGBRegressor(max_depth = 10,#was 12
        objective='reg:gamma',
        colsample_bytree = 0.7,#new
        learning_rate=0.05,#was 0.3
        n_jobs=-1,
        n_estimators=1000,#was 100
        eval_metric='gamma-nloglik',
        tree_method='gpu_hist',
        gpu_id=0)
```

4.4 Отримання результатів

Налаштувавши створену систему було проведено оцінку помилок наших методів машинного навчання на задачі оцінки об'єктів нерухомості. Кількість помилок та якісні показники після навчання за допомогою **«випадкового лісу»**:

R: 0.5853136860101833

MeanAbsErr: 37.1 %

MedAbsErr: 14.0 %

Кількість помилок та якісні показники після навчання за допомогою **XGBoost**:

R: 0.6441264889703582

MeanAbsErr: 33.9 %

MedAbsErr: 13.8 %

Кінцеві результати об’єднання:

R: 0.6345566718945927

MeanAbsErr: 35.1 %

MedAbsErr: 13.6 %

Також було **протестовано** кінцеву модель на випадковій квартирі:

```
lon = 50.5054845396644
lat = 30.414619445800785
distance =  geodesic([lat,lon], city_center_coordinates).km
azimuth =  get_azimuth(lat,lon)
predictOnTrainedModel(pd.DataFrame({
    'floor': [20],
    'total_floor': [24],
    'area': [66],
    'rooms': [3],
    'kitchen_area':[19],
    'distance': [distance],
    'azimuth': [azimuth],
    'lon': [lon],
    'lat': [lat]
}))
```

Результат розрахунку :

- Predicted price: **74000 usd**
- Реальним значенням обраної квартири було **80000 usd**.

Дані результатами входять у похибку описану у кінцевих результатах об'єднання навчальних моделей та є достатньо точними беручи до уваги досліджувану область та відгук представника компанії у середовище якої проводиться тестове залучення системи.

Також побудовано графік важливості параметрів нашого датасету:

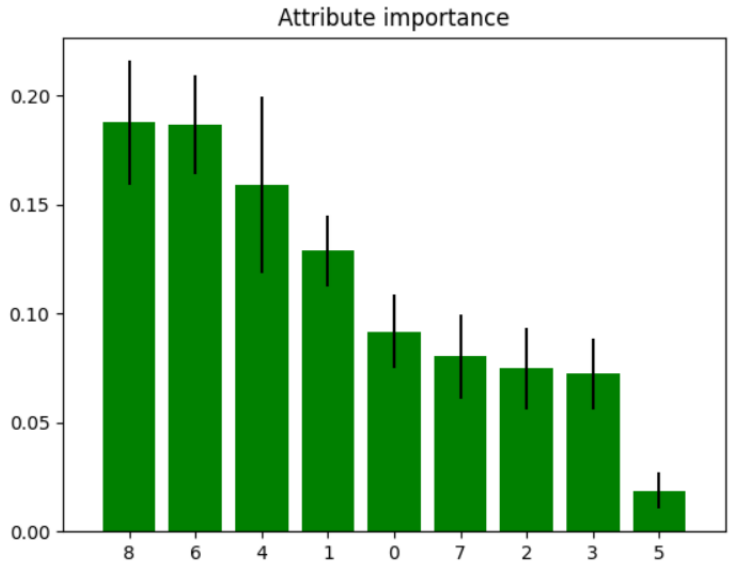

Рис.7 Графік відносної важливості параметрів датасету

Результати у чисельному вигляді:

1. lon (0.187612)
2. kitchen_area (0.186614)
3. area (0.158988)
4. distance (0.128840)
5. azimuth (0.091844)
6. lat (0.080315)
7. floor (0.074746)
8. total_floor (0.072240)
9. rooms (0.018801)

ВИСНОВОК ДО РОЗДІЛУ 4

У результаті проведення процесу розробки програмного забезпечення було створено модуль навчання моделей регресійних лісів та модель градієнтного бустинга.Також створено модуль оцінки, який взаємодіє з інтерфейсом системи, а також модель оптимізації (тюнінгу) гіпер-параметрів, який покращує якість навчальних моделей. Тобто, у результаті даного розділу було створено готове ПЗ, яке є готовим до роботи у підприємствах або у якості безкоштовного веб-сервісу, а також є протестованною мануально співробітником підприємства, в яке проводиться тестова інтеграція проекту.

ВИСНОВКИ

В рамках даної роботи було проведено ознайомлення та вивчення теоретичних матеріалів щодо машинного навчання і оцінки об’єктів нерухомості за допомогою алгоритмів у даній сфері. Також виконавши цю роботу було набуто навички створення власного ПЗ, роботи з прикладними мовами програмування, а також набуто нові теоретичні і практичні навички.

Результатами даної роботи стала система, яка містить в собі навчанні моделі машинного навчання за допомогою алгоритмів «випадково лісу», а також алгоритму XGBoost. Дані результати дають нам можливість використовувати дану модель для оцінки об’єктів нерухомості, які ще не наявні у нашому датасеті та проводити процес передбачення ціни на об’єкт нерухомості за допомогою введення параметрів цієї квартири та її місцезнаходження.

Важливим показником якості створеного ПЗ є медіанна абсолютна процентна помилка, яка чітко демонструє похибки моделі, що була створена. У системі цей показник на кінець роботи дорівнює 13.3 %.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stuart Russell, Peter Norvig: Artificial Intelligence: Pearson New International Edition: A Modern Approach (2013)
2. Sarah Guido, Andreas C. Müller: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition (2019) - (с. 123-128) інтуїтивне розуміння концепцій та інструментів для побудови інтелектуальних систем.
3. https://medium.com/machine-learning-for-humans/supervised-learning-740383a2feab - опис контрольованого навчання: регресія та класифікація. Лінійна регресія, функції втрат та градієнтний спуск.
4. https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db - порівння основних алгоритмів для бустингу даних
5. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html - офіційна документація sklearn щодо використання модуля RandomizedSearchCV
6. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/notebooks - резульати проведенного змагання щодо аналізу цін на нерухомість з кращими практиками використання
7. https://neurohive.io/ru/osnovy-data-science/vvedenie-v-scikit-learn/ - огляд основних складових бібліотеки sklearn та способів їх застосування
8. https://tproger.ru/translations/python-random-forest-implementation/ - розбір алгоритму “рандомного лісу” і практичного використання разом с мовою програмування python
9. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html# - офіційна документація sklearn щодо використання модуля GridSearchCV

10. https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1 - стаття про аналіз даних за допомою лінійної регресії для порівняння алгоритмів
11. https://yalantis.com/blog/predictive-algorithm-for-house-price/ - стаття з рекомендаціями щодо аналізу даних
12. https://towardsdatascience.com/fine-tuning-xgboost-model-257868cf4187 - рекомендації щодо “тюнингу” гіперпараметрів XGBoost моделі
13. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html - офіційна докуметація panas, розділ опису DataFrame
14. https://www.kaggle.com/c/zillow-prize-1/overview - змагання на сервісі kaggle від компанії zillow, що займається процесом оцінювання об’єктів нерухомості
15. https://en.wikipedia.org/wiki/Random_search - опис методу оптимізації RandomSearch

ДОДАТКИ

Додаток А

Можливості подальшої оптимізації

Дана система має великий потенціал для покращення, а основними кроками для цього можуть стати:

- Робота з датасетами по всій Україні та створення моделей для кожного міста
- Створення можливості управління навчанням моделі із зовнішнього інтерфейсу, тобто можливість порівняти самостійно навчанні моделі між собою
- Розширення списку параметрів, які розглядаються та пошук нових параметрів
- Потокове навчання системи з не великим інтервалом

Для більш швидкісної роботи із створеним сервісом сервер повинен мати швидкодіючий процесор - 4 ядра, 8 логічний ядер та швидкодія від 4.2 Ггц ,і відеокарту - від 4 гігабайт відеопам'яті (від NVIDIA GTX 1050 TI).