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

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

«До захисту допущено»

в. о. завідувача кафедри

_____ О.Л. Тимощук

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040303 «Системний аналіз»

на тему: «Методи аналізу геоданих для надання рекомендацій щодо розміщення громадських закладів у Києві»

Виконав:
студент IV курсу, групи КА-51
Трохимович Микола Андрійович _____

Керівник:
доцент, к. ф.-м. н. Каніовська І. Ю. _____

Консультант з економічного розділу:
доцент, к. е. н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к. т. н. Коваленко А. Є. _____

Рецензент:
доцент, к. ф.-м. н. Буценко Ю.П. _____

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

Київ – 2019 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)

Напрям підготовки (програма професійного спрямування) – 6.040303 «Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ
в. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Трохимовичу Миколі Андрійовичу

1. Тема роботи «Методи аналізу геоданих для надання рекомендацій щодо розміщення громадських закладів у Києві», керівник роботи доцент кандидат ф.-м. н., Каніовська Ірина Юріївна, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____
_____
_____

4. Зміст роботи_____
_____
_____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ М. А. Трохимович

Керівник роботи. _____ І. Ю. Каніовська

РЕФЕРАТ

Дипломна робота: 96 с., 28 рис., 8 табл., 2 додатки, 19 джерел.

DATA MINING, МАШИННЕ НАВЧАННЯ, РЕГРЕСІЯ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ГЕОПОЗИЦІОНУВАННЯ, РАНЖУВАННЯ, ЛІНІЙНА РЕГРЕСІЯ, РЕГУЛЯРИЗАЦІЯ, ДЕРЕВА ПРИЙНЯТТЯ РІШЕНЬ, ВИПАДКОВІ ЛІСИ, БУСТИНГ, НОРМАЛІЗАЦІЯ, ВІЗУАЛІЗАЦІЯ

Об’єкт дослідження – аналіз геоданих, алгоритми машинного навчання, побудова складних багаторівневих моделей.

Предмет дослідження – точкові геодані та математичні моделі побудови рекомендацій щодо геопозиціонування.

Мета роботи – аналіз точкових геоданих та дослідження можливості надання рекомендацій, щодо розташування громадських закладів.

Методи дослідження – інтелектуальний аналіз даних (англ. data mining), методи штучного інтелекту.

Актуальність роботи зумовлена тим, що сьогодні, у добу великих даних, стрімко розвивається напрям “розумне місто” (англ. smart city). Він полягає у тому, щоб використовувати накопичені знання про діяльність людей з метою покращення міської інфраструктури та рівня життя. У свою чергу, аналіз геоданих та побудова предиктивних моделей, потенційно може покращити доступність товарів та послуг для людей, та збільшити вигоду для бізнесу.

Результатом роботи є побудована модель машинного навчання для визначення рейтингу конкретного виду громадських закладів та надана рекомендація щодо геопозиціонування, картографічну візуалізація для інтерпретації роботи моделей та демонстрації результатів.

Шляхи подальшого розвитку предмету дослідження – розширення списку ознак та моделей, перевірка роботи програми для різних категорій громадських закладів. Перевірка на даних з інших міст.

ABSTRACT

Theme: “Geodata analysis methods for venues establishment recommendation in Kyiv”

Diploma work: 96 p., 28 fig., 8 tabl., 2 appendixes, 19 sources.

DATA MINING, MACHINE TRAINING, REGRESSION, RECOMMENDATION SYSTEMS, GEOPOSITION, RANKING, LINEAR REGRESSION, REGULARIZATION, DECISION TREE, RANDOM FOREST, BOOSTING, NORMALIZATION, VISUALIZATION

The object of research is the analysis of geodata, algorithms of machine learning, construction of complex multilevel models.

Subject of research is geodata and mathematical models for constructing recommendations for geo positioning.

The purpose of the work is to analyze geodatabases and to study the possibility of providing recommendations on the location of public institutions.

Methods of research – data mining, methods of artificial intelligence.

The relevance of the work is caused by fact that today, in the era of large data, the direction of "smart city" (smart city) is rapidly developing. It consists in using the accumulated knowledge about the activities of people in order to improve the city infrastructure and standard of living. In turn, analyzing geodata and constructing predictive models can potentially improve the availability of goods and services for people and increase business profits.

The result of the work is a machine learning model designed to determine the rating of a specific type of public institutions, and provided guidance on geo positioning, cartographic visualization for the interpretation of model work and demonstration of results.

Ways of further development of the subject of research - the expansion of the list of features and models, checking the program work for various categories of public institutions. Checking data from other cities.

ЗМІСТ

ВСТУП

Масове споживання послуг і товарів є найважливішим компонентом повсякденного життя городян у контексті постіндустріальної економіки. Якість послуг стала безперечною цінністю, що виділяє заклади серед інших. Сучасний міський спосіб життя вимагає добірних, доступних і різноманітних послуг. Економічний розвиток сприяє конкуренції на ринку та створює тенденцію до ускладнення та диверсифікації послуг, зокрема такий процес спостерігається і в столиці України – Києві.

Цифровізація міського життя та зростання можливостей спілкування в Інтернеті між виробником і споживачем дозволяють виробникам просувати свої послуги більш ефективно, клієнти можуть залишати свої відгуки про послуги, а аналітики в свою чергу відстежувати тенденції розвитку послуг базуючись на міських даних, що надходять з різних джерел.

Інтернет-платформи, такі як карти Google, Foursquare, Instagram, стають значними чинниками впливу на процеси споживання. Інформація про розташування, експертні оцінки закладів, фотографії, відгуки відвідувачів роблять онлайн сервіси одним з основних інструментів маркетингу. Також за сприяння державних органів управління та громадської ініціативи створюються сервіси для контролю за рухом громадського транспорту, що значно спрощують пересування по місту та є важливим джерелом даних про мобільність населення, що може бути використано для інтелектуального проектування міста, оптимізації маршрутів громадського транспорту, а також в цілях ефективного просування бізнесом своїх продуктів там де це дійсно потрібно. Також

існує тенденція перенесення даних про забудованість, мережі вулиць з архіву держгеокадастру у цифровий формат.

Актуальність роботи полягає у тому, що всі раніше згадані дані можуть сприяти розвитку бізнесу в Україні, що в свою чергу буде сприяти покращенню сервісу в громадських закладах та сприяти покращенню рівня життя в нашій країні.

Об’єктом дослідження є аналіз геоданих та алгоритмів машинного навчання, побудова складних багаторівневих моделей.

Предметом дослідження визначено точкові геодані та математичні моделі побудови рекомендацій щодо геопозиціонування.

Метою роботи є аналіз точкових геоданих та дослідження можливості надання рекомендацій, щодо розташування громадських закладів.

Методи дослідження – інтелектуальний аналіз даних (англ. data mining), методи штучного інтелекту.

Пояснювальна записка складається з чотирьох розділів. У першому розділі досліджуються та аналізуються існуючі підходи до геопозиціонування та здійснюється постановка задачі. У другому розділі розглянуто ряд математичних моделей, за допомогою яких будують прогнози. Третій розділ описує архітектуру розробленої програми, також у ньому аналізуються результати роботи алгоритму. Четвертий розділ являє собою економічну частину, в якій розробляється кошторис витрат на розробку.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ЗАДАЧІ ГЕОПОЗИЦІОНУВАННЯ ТА НАЯВНИХ ДАНИХ

1.1 Аналіз актуальності задачі геопозиціонування

Проблема визначення оптимального місця для нового закладу розглядається завжди, особливо в галузі землеустрою. Вирішення цього завдання є важливим фактором для успішності бізнесу. Традиційні підходи до проблеми враховували демографічні показники, статистику доходів та інші агреговані дані. Придбання відповідних даних зазвичай є дорогим або ж взагалі недоступним. Однак, зі зростанням соціальних мереж, останнім часом стали доступними точкові дані, що описують мобільність користувачів і популярність конкретних місць. Такі дані можна і потрібно використовувати для ефективного просування бізнесу.

1.2 Огляд існуючих практичних рішень

У даному підрозділі буде зроблено акцент на тому, як задача геопозиціонування вирішується на практиці. Варто зазначити, що спосіб знаходження найкращого місця для громадського закладу сильно залежить від фінансових можливостей підприємства.

Коли мова йде про невеликі підприємства то зазвичай такою задачею займаються локальні агентства для проведення соціологічного дослідження. Їхня робота полягає в емпіричному підрахунку потоків людей та їхнього матеріального достатку, можливо опитуванням. Зрозуміло, що результати таких досліджень є досить зміщеними та сильно залежать від людського фактору. Тому рекомендації, надані як результат такої роботи, не завжди здатні призвести до бажаного ефекту.

Більш дорогими та комплексним дослідженням займаються, наприклад, відділи роботи з великими даними мобільних операторів [1]. Їхній підхід полягає в поділі міста на умовні географічні квадрати та здійснення базової дескриптивної аналітики по таким регіонах. Також побудова так званих теплових карт як на рисунку 1.1.

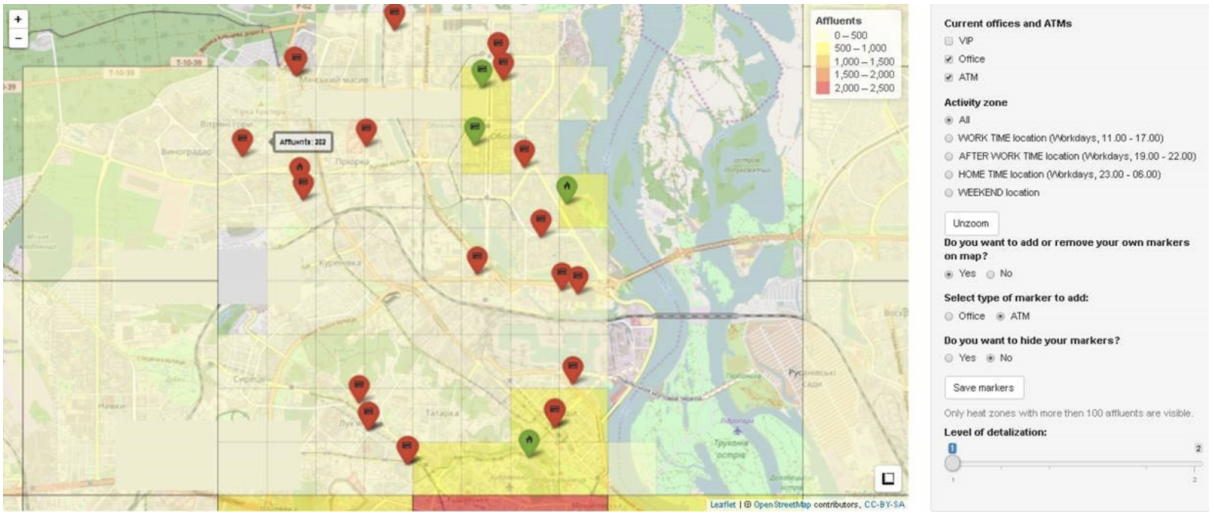


Рисунок 1.1 – Теплова карта активності абонентів Київстар

Такий підхід хоч і дає більш глибоку і комплексну оцінку, проте суттєвим недоліком є дещо грубий підхід у поділі на райони у вигляді квадратів та відсутність автоматизованих систем аналізу таких даних. Тобто залишається проблема наявності людського фактору в прийнятті рішення та можливість суб’єктивності у висновку експерта.

1.3 Опис наявних даних для аналізу

Дані становлять дуже велику цінність у сучасному світі. Їхнє використання дозволяє не тільки підвищувати рівень життя людей, а й отримувати величезні прибутки бізнесом. Тому існує велика проблема

відкритості даних. Тобто набори інформації зазвичай є або безцінною комерційною таємницею, або продаються за дуже велику ціну. Однак, високотехнологічні компанії часто роблять свої набори даних відкритими для академічного використання з метою науково-технічного розвитку людства. У даному підрозділі опишемо джерела даних, що використовувалися у дослідженні.

1.3.1 Дані з соціальної мережі Foursquare

Foursquare — це соціальна мережа, що використовує інформацію щодо локації, яка дозволяє дізнатися, де знаходяться інші користувачі, а також пропонує підказки базуючись на діях інших користувачів щодо різноманітних місць, в залежності від персональних вподобань. Користувачі можуть використовувати свій смартфон, щоб чекінитися (від англ. “check-in”) в різноманітних місцях, які відвідують.

У роботі використовуємо набір даних, що включає в себе довгострокові (близько 22 місяців з квітня 2012 по січень 2014 року) глобальні геодані, зібрані з Foursquare по всьому світу. Набір містить 90048627 чекінів, що були зроблені 2733324 користувачами у 11180160 закладах [2]. Так як академічний датасет немає повного переліку з доступних атрибутів, також було використано Foursquare API, для того щоб доповнити дані інформацією про відгуки та уподобання. Варто зазначити, що датасет є достатньо об’ємним, тому можна вважати, що в даній задачі маємо справу з так званими “великими даними” (англ. big data). Їхня обробка потребує багато часу і обчислювальних потужностей. Окремо маємо два набори даних. Перший датасет містить інформацію, що стосуються конкретних закладів, а саме їхній ключ-ідентифікатор та дані

про розташування. Інший датасет стосується взаємодії людей з відповідними закладами та містить інформацію про ідентифікатори користувача та закладу, час їхньої взаємодії.

З глобального набору даних було виокремлено лише та інформація, що стосується міста Києва. Зроблено це шляхом фільтрації за ознаками належності певній країні, та фільтрації по широті та довготі першого набору. Було прийнято, що заклад належить допустимій області, якщо він лежить у проміжку від 50°31′ до 50°57′ північної широти та 30°24′ до 30°81′ східної довготи. На першому етапі отримали 77358 закладів з глобального датасету, що належать Україні, та 24473 належать безпосередньо досліджуваній області Києва. Також була зроблена візуалізація отриманих результатів фільтрації за допомогою програмного засобу наданого за відкритою ліцензією корпорацією UBER. Варто зазначити, що заклади розподілені по території України нерівномірно, а саме більшість знаходиться у великих містах або на важливих автошляхах як представлено на рисунку 1.2.

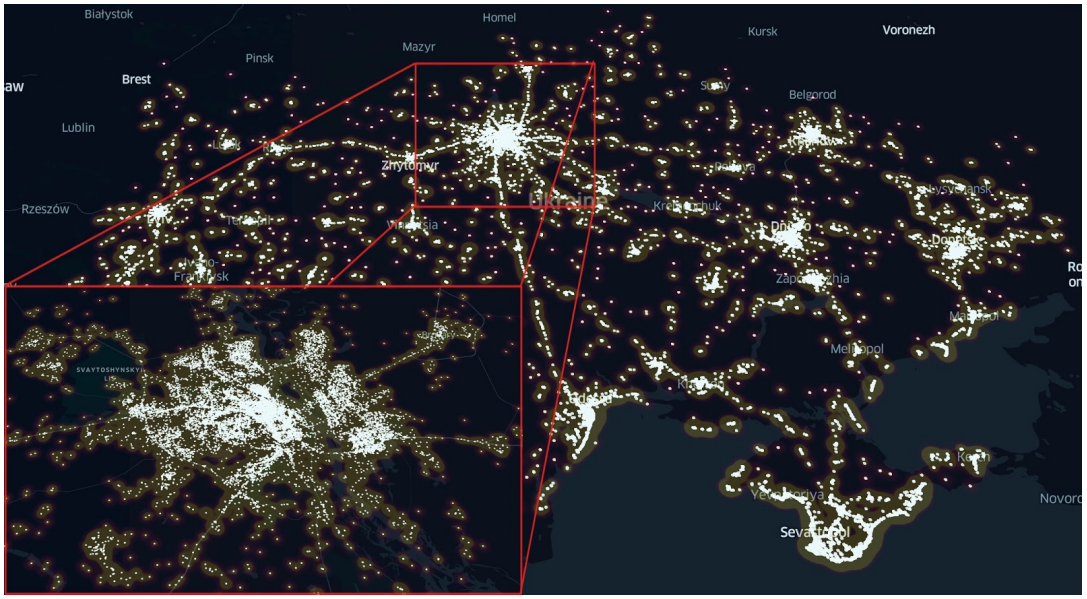


Рисунок 1.2 – Громадські заклади в наборі даних Foursquare в Україні

Візуалізація у проекті з геоданими є важливим етапом, адже вона дозволяє наочно оцінити результат і робить значно простішою його інтерпретацію. Таким чином отримали дані що стосуються Києва і містять інформацію про 24473 закладів, які належать 405 різним категоріям. Також в ході аналізу було виявлено, що 56 категорій мають не більше двох закладів, тобто є малочисельними.

Далі, знаючи, які заклади належать Києву, було проведено попередню фільтрацію чекінів, та виокремлено 217805 унікальних чекінів, які були зроблені 11168 користувачами у Києві. Проаналізувавши отримані результати, було виявлено, що існують користувачі, що зробили менше п’яти відміток і таких користувачів 54%. Вони вважалися не активними, тому в подальшому дослідженні інформація про них не використовувалася.

1.3.2 Додаткові джерела інформації

Важливим фактором при позиціонуванні бізнесу з надання послуг є розвинута мережа громадського транспорту. У ході роботи використовується дані, здобуті з EasyWay API. Зокрема даний ресурс дозволяє за заданими координатами точки здобути інформацію про наявні маршрути громадського транспорту різного роду, а саме: трамваїв, метро, тролейбусів, маршрутних таксі, автобусів. Дана інформація надається з прив’язкою до зупинки громадського транспорту, яка повинна бути у заданому радіусі від досліджуваної точки. Наприклад, на рисунку 1.3 зображено карту на якій представлено зупинки громадського транспорту в радіусі 200 метрів від заданого громадського закладу. Дану інформацію було отримано за допомогою EasyWay API. Також функціонал дозволяє

визначити “важливі” транспортні вузли, тобто такі, які мають значний пасажиропотік.

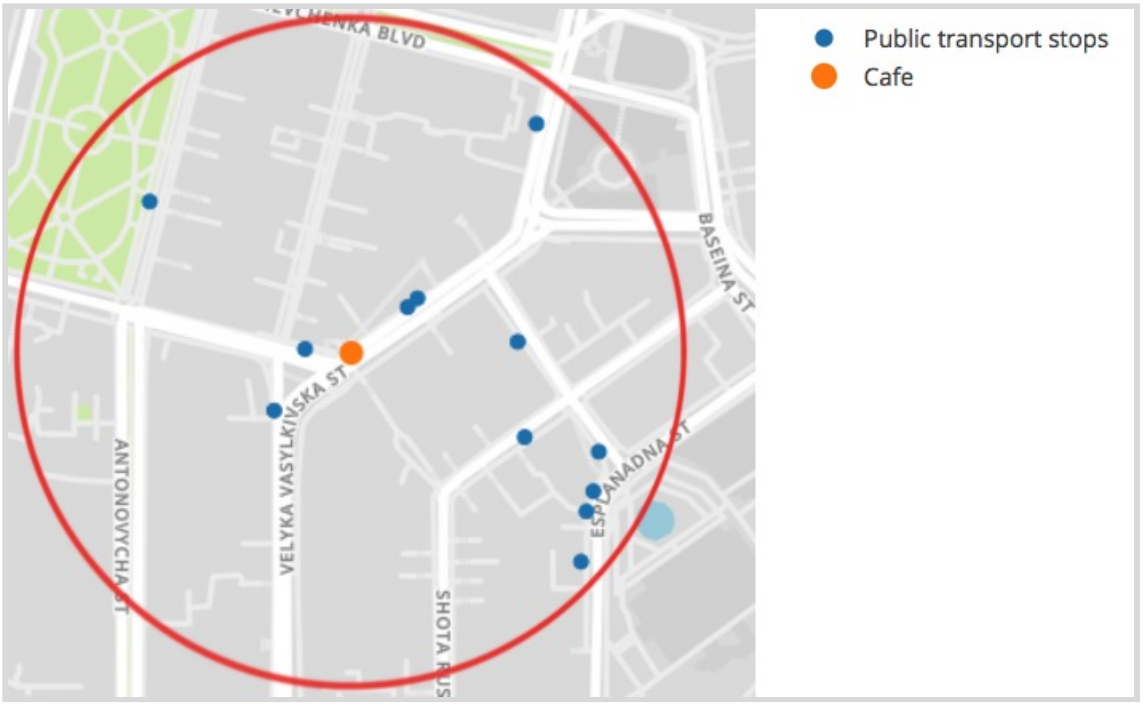


Рисунок 1.3 – Приклад роботи EasyWay API

Також використовується загальнодоступний інструмент OSMnx. OSMnx дає значні можливості для дослідників і практиків: по-перше, автоматичне завантаження політичних кордонів і меж будинків; по-друге, автоматизоване завантаження даних вуличної мережі з OpenStreetMap; по-третє, алгоритмічна корекція топології мережі; по-четверте, можливість зберігати вуличні мережі у пам’яті у вигляді шейп-файлів, файлів GraphML або SVG; і по-п'яте, можливість аналізувати вуличні мережі, включаючи обчислення маршрутів, проектування та візуалізацію мереж [3,4]. У даній роботі за допомогою OSMnx будемо аналізувати граф вуличної мережі навколо потенційного місця розташування громадського закладу.

Наприклад, можемо отримати граф вуличної мережі для карти з рисунку 1.3, що показано на рисунку 1.4.

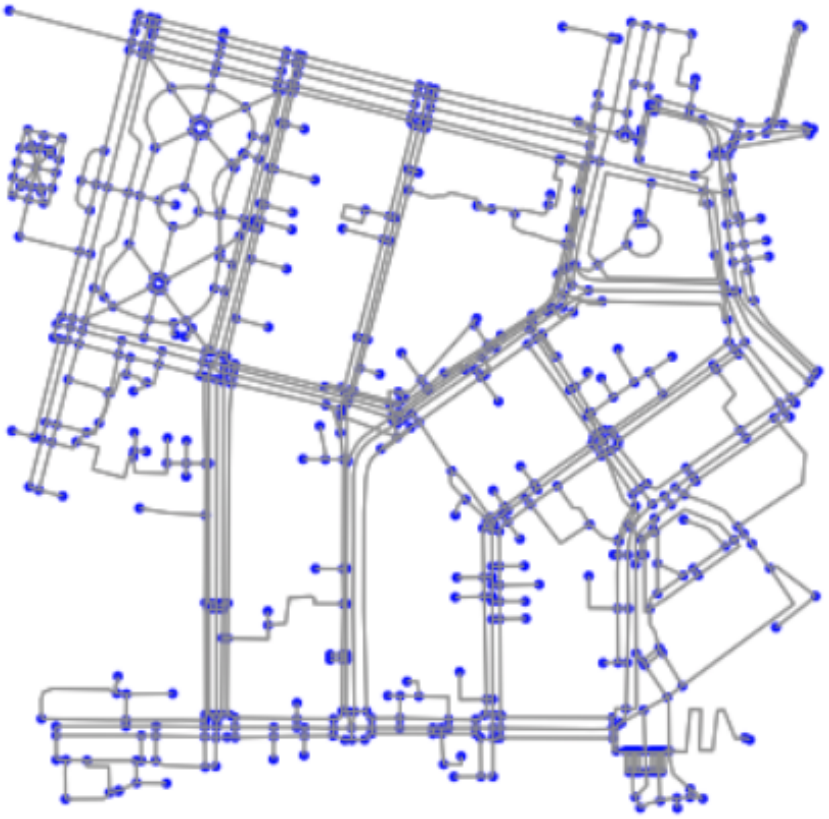


Рисунок 1.4 – Приклад вуличного графу отриманого з OSMnx

1.4 Постановка та формалізація задачі

Мета дослідження полягає в тому, щоб визначити найкраще місце серед кандидатів набору потенційних розташувань для відкриття нового громадського закладу. Під найкращим розташуванням ми будемо розуміти місце, де громадський заклад буде успішним та популярним. Найкращим способом виміряти успішність бізнесу є оцінити його прибутки, але ця інформація є майже завжди недоступною комерційною таємницею. Тому надалі будемо оцінювати заклад за кількістю “чекінів”, та називатимемо це цільовим значенням. Тобто чим більше цільове значення у закладі – тим

популярніший і успішніший є цей бізнес. Основним припущенням у формулюванні цього завдання є те, що кількість емпірично спостережуваних чекінів користувачами Foursquare можна використовувати як апроксимацію для відносної популярності закладу.

Проаналізувавши дані з різних джерел, здобудемо з них необхідні атрибути конкретних локацій, які використаємо для побудови регресійної моделі. За допомогою моделі знайдемо цільові значення для нових локацій. Далі, проранжувавши ці значення, дамо рекомендацію щодо найкращих місць для розміщення.

Отже визначено такі ключові завдання:

1) Визначити показник успішності громадського закладу;

2) Проаналізувати дані з різних джерел, спроектувати та видобути необхідні атрибути конкретних локацій;

3) Побудувати модель для відновлення функціональної залежності успішності громадських закладів від здобутих географічних ознак;

4) За допомогою моделі знайти цільові значення для нових локацій;

5) Проранжувати значення успішності для нових закладів, дати рекомендацію щодо найкращих місць для розміщення конкретного типу громадських закладів;

6) Оцінити якість роботи рекомендаційної системи.

Ранжування – завдання сортування набору елементів з міркування їх релевантності. Найчастіше релевантність розуміється по відношенню до деякого об'єкту. У нашому випадку релевантність – це кількість “чекінів”. Формально, розглянемо M елементів $E=\left\{e_j\right\}_{j=1}^{M}$. Результат роботи

алгоритму ранжування елементів E – це відображення $r : E \to R$, яке створює відповідність кожному елементу $e \in E$ вагу $r(e)$, що характеризує ступінь релевантності елемента об'єкту (чим більше вага, тим релевантніший об'єкт). При цьому, набір ваг $\{r(e)\}_{e \in E}$ задає перестановку $\pi : [1..M] \to [1..M]$ на наборі елементів елементів E (вважаємо, що безліч елементів впорядковані) виходячи з їх сортування за спаданням ваги $r(e)$.

Щоб оцінити якість ранжування, необхідно знати правильний порядок, який можна було б порівняти з результатами роботи алгоритму. Розглянемо $r^{true} : E \to [0,1]$ – функцію релевантності, що характеризує «справжню» релевантність елементів. А саме $r^{true} = 1$ – елемент ідеально підходить, $r^{true} = 0$ – повністю не релевантний. Аналогічно відповідну перестановку $\pi^{true} : E \to [1..M]$, за спаданням $r^{true}(e)$.

1.5 Висновки до розділу 1

У даному розділі було визначено актуальність дослідження, було розглянуто існуючі практичні підходи до вирішення задачі геопозиціонування. Визначено джерела даних, що будуть використовуватися у подальшому дослідженні. Зроблено постановку задачі. Проведено формалізацію практичного завдання та визначено основні етапи дослідження.

РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ ТА МЕТРИКИ

2.1 Метрики якості наданої рекомендації

Для визначення того, наскільки добре проранжовані потенційні кандидати для розміщення громадського закладу, введемо декілька метрик, які б оцінювали якість нашої рекомендації.

Ціль знаходження метрики ранжування – визначити, наскільки отримані алгоритмом оцінки релевантності $r(e)$ і відповідна їм перестановка π відповідає істинним значенням релевантності $r^{true}(e)$.

2.1.1 Precision at K ($p@K$)

Precision at K ($p@K$) – точність на K елементах, базова метрика якості ранжування для одного об'єкта. Припустимо, наш алгоритм ранжування оцінює релевантності для кожного елемента $\{r(e)\}_{e \in E}$. Відібравши поміж них перші $K \le M$ з найбільшим $r(e)$ можна обчислити частку релевантних, що і буде метрикою представленою формулою (2.1) [5].

$$p@K = \frac{\sum_{k=1}^{K} r^{true}(\pi^{-1}(k))}{K} = \frac{\text{кількість релевантних}}{K} \qquad (2.1)$$

У формулі (2.1) $\pi^{-1}(k)$ – це такий $e \in E$, який в результаті перестановки π виявився на k-ій позиції. Тому маємо, $\pi^{-1}(1)$ – елемент з найбільшою $r(e)$, $\pi^{-1}(2)$ – елемент з другою за величиною $r(e)$ і так далі.

2.1.2 Average precision at K

Precision at K – метрика проста для розуміння та імплементації, проте вона має істотний недолік, а саме вона не враховує порядок елементів. Так, якщо з десяти елементів ми вгадали тільки один, то не важливо на якій позиції він був, у будь-якому випадку $p@10 = 0.1$. При цьому очевидно, що перший елемент кращий за наступні [5].

Цей недолік враховує та виправляє метрика ранжування average precision at K ($ap@K$), яка дорівнює сумі $p@k$ за індексами k від 1 до K тільки для релевантних елементів, що нормується значенням K:

$$ap@K = \frac{1}{K} \sum_{k=1}^{K} r^{true}(\pi^{-1}(k)) \cdot p@k$$

2.1.3 Normalized discounted cumulative gain (nDCG)

Розглянемо один об'єкт і K елементів з найбільшим $r(e)$. Cumulative gain at K ($CG@K$) – базова метрика ранжування, яка працює за принципом: найкраще, якщо найбільш релевантні елементи спочатку.

$$CG@K = \sum_{k=1}^{K} r^{true}(\pi^{-1}(k))$$

Однак ця метрика має очевидний недолік: вона не враховує позицію релевантних елементів. Однак є і суттєва перевага: на відміну від попередніх метрик може бути враховане не бінарне значення релевантності r^{true}.

Discounted cumulative gain at K ($DCG@K$) – модифікація cumulative gain at K, яка дозволяє враховувати порядок елементів у списку. Реалізується це шляхом домноження релевантності елемента на коефіцієнт, що дорівнює оберненому логарифму позиції:

$$DCG@K = \sum_{k=1}^{K} \frac{2^{r^{true}(\pi^{-1}(k))}-1}{log_2(k+1)}$$

Використання логарифма у якості функції дисконтування можна пояснити такими міркуваннями: з точки зору ранжування позиції на початку списку відрізняються набагато більше, ніж позиції в кінці. Ці суб'єктивні міркування прекрасно виражаються за допомогою логарифма. Таким чином, ми вирішуємо проблему з урахуванням позицій релевантних елементів, але тільки посилюємо проблему з відсутністю нормування: якщо $CG@K$ варіюється на проміжку $[0, K]$, то $DCG@K$ приймає значення на не зовсім зрозумілому проміжку. Тому вводимо нормалізовану версію $DCG@K$ за допомогою формули (2.2):

$$nDCG@K = \frac{DCG@K}{IDCG@K}, \quad (2.2)$$

де $IDCG@K$ – це максимальне значення $DCG@K$.

Таким чином, $nDCG@K$ успадковується від $DCG@K$. Дана метрика враховує позиції елементів в списку і при цьому приймає значення в діапазоні від 0 до 1 [5].

2.2 Аналіз даних та побудова регресорів

У ході дослідження було виявлено, що різні категорії закладів по різному впливають на успішність розташування один одного [6].

Наприклад, розташування поряд декількох музеїв може підвищити успішність кожного з них. Все це тому, що люди, бажаючи побачити якнайбільше, не гаючи часу на дорогу, будуть прямувати в район скупчення музеїв. У випадку з кінотеатрами, наприклад, це буде працювати не так: мало хто буде відвідувати два кінотеатри, що знаходяться поряд. Навпаки, буде обрано тільки один з них, що знизить успішність іншого. Безумовно, цей фактор потрібно враховувати при дослідженні.

У якості досліджуваної категорії було обрано кав'ярні, так як ми маємо достатню кількість закладів в досліджуваному наборі даних, а саме ми володіємо інформацією про 250 кав'ярень. Мінімальна кількість чекінів в кав'ярні становить 1, максимальна – 153. У середньому у кожної кав'ярні є по 20 чекіни.

2.2.1 Підготовка даних про переміщення

Соціальні мережі, засновані на розташуванні, надають унікальну можливість оцінити географічну територію не тільки шляхом розгляду статично-просторової інформації. Також маємо змогу аналізувати динаміки рухів мобільних користувачів. Більш детально, ми вивчаємо переміщення користувачів між місцями, виведені з їх послідовних чекінів у різних місцях. Агреговані дані переміщень можуть бути ефективно використані для аналізу потоків користувачів до місця і взагалі в навколишньому районі [7]. Переміщення визначається як пара чекінів, здійснених анонімним користувачем, у двох різних місцях протягом періоду часу, що менший трьох годин і визначається часом початку, кінцем, первинним місцем розташування та місцем призначення [8]. Загрегувавши дані з

Foursquare, отримали інформацію про 65425 переміщень в місті Києві. Це дає нам змогу проаналізувати потоки людей та використати цю інформацію для побудови потенційних регресорів.

Проаналізуємо відстані, які люди долають під час переміщення. На рисунку 2.1 наведено графік емпіричної функції розподілу (ECDF) відстаней переміщень для всієї вибірки та окремо тільки тих переміщень кінцевою точкою яких є кав'ярня.

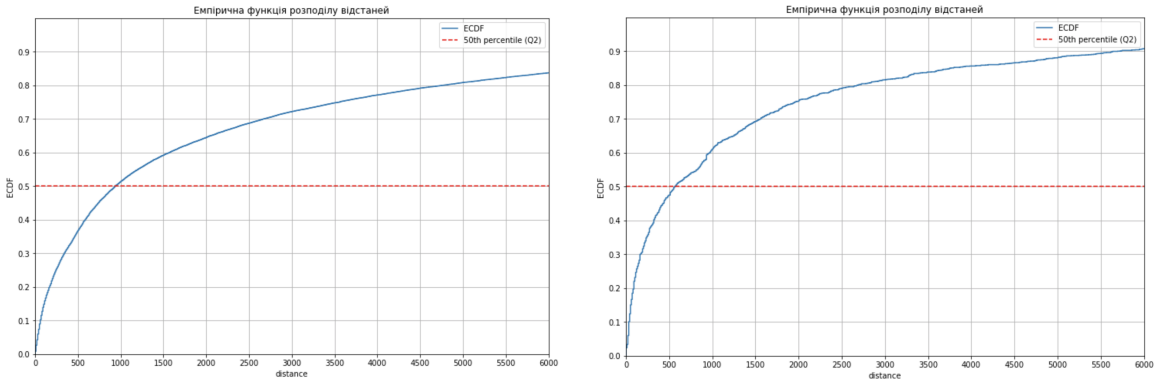


Рисунок 2.1 – Графік емпіричної функції розподілу (ECDF) відстаней переміщень для усієї вибірки та тільки до кав'ярень

Як можемо помітити, близько 50% переміщень у загальній вибірці є на відстань меншу ніж 1000 метрів. У той час коли медіана підвибірки переміщень, у яких кінцевою точкою є кав'ярня, є дещо зміщеною і має значення близьке до 500 метрів. Цей результат свідчить про те, що клієнти кав'ярень зазвичай приходять туди з не віддалених місць. Виходячи з даних міркувань, у подальшому аналізі навколишньої місцевості будемо використовувати територію в радіусі менше 500 метрів.

2.2.2 Принципи побудови регресорів для даних з Foursquare

Формально, розглядаючи набір місць L, в яких комерційне підприємство зацікавлене в розміщенні свого бізнесу, ми хочемо ідентифікувати оптимальну область $l \in L$, так що відкритий у l заклад, потенційно може залучити найбільше число відвідувачів. Область l закодована своїми координатами широти і довготи, а також радіусом r, як, наприклад, показано на рисунку 2.2.

Рисунок 2.2 – Область радіусу 200 метрів навколо кав’ярні у Києві

Проекспериментувавши з різними значеннями для радіуса, було обрано 200 метрів, оскільки це дало найкращі експериментальні результати, тобто найвищу продуктивність прогнозування за незалежними експериментами з різними значеннями r, і це також узгоджується з тим, що містобудівне співтовариство вважає оптимальним розміром локального району [7, 9].

Більш конкретно, вимірюємо щільність, ентропія (різноманітність) і конкурентоспроможність навколишнього району, аналізуючи множину $\{p \in P : dist(p, l) < r\}$ місць, що лежать в радіусі r навколо місця l. Функція *dist* позначає географічну відстань між двома місцями з загального набору закладів P.

Щільність (*density*) вимірюємо як кількість закладів навколо заданого місця. Таким чином ми оцінюємо, наскільки популярність місця залежить від кількості інших місць у тій же області. Формально:

$$density = |\{p \in P : dist(p, l) < r\}|$$

З огляду на те, що радіус r однаковий, щільність географічних областей кандидатів залежить тільки від кількості місць, які вони включають. Позначимо число сусідів місця в радіусі r як $N(l,r)$. Інтуїтивно зрозуміло, що більш щільна область може означати більшу ймовірність можливості візиту до громадського закладу.

Ентропія розташування (*place entropy*) оцінює вплив просторової гетерогенності області на популярність місця. Ми застосовуємо ентропію з теорії інформації за Клодом Шенноном до частоти типів місць в області [10]. Позначимо число місцевих сусідів типу γ як $N_\gamma(l,r)$.

$$place\ entropy = -\sum_{\gamma \in U} \frac{N_\gamma(l,r)}{N(l,r)} \times log \frac{N_\gamma(l,r)}{N(l,r)} \qquad (2.3)$$

У формулі (2.3) U позначає множину всіх можливих категорій закладів.

В цілому, територія з високими значеннями ентропії, як очікується, буде різноманітною з точки зору типів місць, тоді як найменш ентропійні райони означатимуть, що кількість чекінів є упередженою до певної категорії, наприклад, для будинку, якщо цей район є житловим.

Конкурентоспроможність (*Competitiveness*) навколишнього простору також є важливою ознакою, яку необхідно враховувати. Вважаючи тип місця для прогнозу γ_l (наприклад, кав'ярня), ми вимірюємо частку сусідніх закладів одного типу γ_l з урахуванням загальної кількості прилеглих місць. Потім ми оцінюємо ділянки в зворотному порядку, припускаючи, що найменш конкурентоспроможна область є найбільш перспективною:

$$Competitiveness = -\frac{N_{\gamma_l}(l,r)}{N(l,r)}$$

Однак варто зазначити, що конкуренція в громадських закладах може мати або позитивний, або негативний ефект. Можна очікувати, що, наприклад, розміщення бару в районі, що вважається місцем нічного життя, матиме позитивний результат, оскільки вже існує екосистема супутніх послуг і потік людей, що притягуються до цієї області. Однак оточення конкурентів може також означати, що існуючі клієнти будуть спільними [6].

Також будемо використовувати інформацію про чекіни користувачів Foursquare для оцінки якості потенційного місця розташування. Мета полягає в тому, щоб визначити, якою мірою інформація з соціальних мереж користувачів може поліпшити географічну бізнес-аналітику і які переваги над інформацією, яка використовує лише статичну просторову

інформацію. Ми розглянемо характеристики, які вимірюють загальну популярність конкретного місця та ознаки, які використовують інформацію про переміщення між місцями.

Загальну популярність ($Popularity$) знаходимо як загальну кількість чекінів, які емпірично спостерігаються серед сусідніх місць у районі спостереження:

$$Popularity = |\{(m, t) \in C : dist(m, l) < r\}| \qquad (2.4)$$

У формулі (2.4), (m, t) позначає чекін, записаний в місці $m \in P$ в час t, а C – набір всіх чекінів в наборі даних.

Щільність переміщень ($Transition\ Density$) обчислюємо припускаючи, що підвищена мобільність між місцями в районі може збільшити кількість випадкових відвідувачів до цільового місця. Формально, позначимо T – сукупний набір послідовних переміщень між місцями, $(m, n) \in T$, $m \in P$ і $n \in P$. Маємо:

$$Transition\ Density = |\{(m, n) \in T : dist(m, l) < r \wedge dist(n, l) < r\}|$$

Також визначаємо функцію для оцінки вхідного потоку ($Incoming\ Flow$) зовнішнього потоку клієнтів до заданої області. Розглядаються переходи між місцями, позначеними як $(m, n) \in T$, таким чином, що перше місце m розташоване зовні, а друге місце n – в заданій області. Формально:

$$Incoming\ Flow = |\{(m, n) \in T : dist(m, l) > r (dist(n, l) < r\}|$$

Очікуємо, що область, яка є гарним атрактором віддалених користувачів, буде перспективною для відкриття нового громадського закладу.

2.2.3 Принципи побудови регресорів для даних з EasyWay API

15 вересня 2017 року – влада в Україні представила Національну доповідь «Цілі сталого розвитку: Україна», яка визначає базові показники для досягнення Цілей сталого розвитку (ЦСР) [11]. Одним із важливих показників є саме частка використання громадського транспорту, яку планується збільшувати з кожним роком. Тому важливо враховувати також такі важливі показники при аналізі людської мобільності в місті. EasyWay API дає змогу отримати дані, про розміщення і деталі функціонування громадського транспорту у місті Києві. У нашому дослідженні ми будемо аналізувати такі показники як кількість зупинок, наявність поблизу важливих транспортних вузлів, кількість маршрутів та різноманітність видів громадського транспорту аналізуючи множини $\{tr \in T : dist(tr, l) < r\}$ місць та $\{ma \in M : dist(ma, l) < r\}$, що лежать в радіусі r навколо місця l. Функція $dist$ позначає географічну відстань, T – загальну множину зупинок громадського транспорту, M – загальну множину маршрутів. EasyWay API за запитом з заданими координатами та радіусом передає відповідь у JSON форматі з якого ми можемо здобути потрібну інформацію.

Кількість зупинок (*stops density*) оцінює, наскільки успішність місця залежить від кількості зупинок громадського транспорту в заданій області. Формально:

$$stops\ density\ = \ |\{tr \in T\ :\ dist\,(tr,\ l)\ < r\}|$$

Позначимо число зупинок в радіусі r як $N_{tr}(l,r)$. Інтуїтивно зрозуміло, що більше зупинок громадського транспорту тим більша ймовірність можливості візиту до громадського закладу.

Наявність поблизу важливих транспортних вузлів та їх кількість напряму впливає на кількість відвідувачів. Множину таких місць назвемо Im. До таких важливих вузлів належать, наприклад, Міжнародний аеропорт «Київ» імені Ігоря Сікорського та залізнична станція Київ-Пасажирський. Формально:

$$important\ nodes\ = \ |\{tr \in Im\ :\ dist\,(tr,\ l)\ < r\}| \qquad (2.4)$$

У формулі (2.4) зрозуміло, що $Im \subset T$.

Також аналогічно будемо аналізувати і кількість маршрутів (*transport density*). Цей показник буде рівний значенню потужності множини маршрутів, що мають зупинки в досліджуваній області. Формально:

$$transport\ density\ = \ |\{ma \in M\ :\ dist\,(ma,\ l)\ < r\}|$$

Ентропія транспорту (*transport entropy*) покликана оцінити вплив різноманітності видів громадського транспорту в досліджуваній області на популярність місця. Аналогічно з ентропією розташування застосовуємо ентропію з теорії інформації за Клодом Шенноном до частоти видів транспорту в області[10]. Позначимо число транспорту виду δ як $N_{tr_\delta}(l,r)$.

$$transport\ entropy\ =\ -\ \text{??}\ \sum_{\delta \in V} \frac{N_{tr_\delta}(l,r)}{N_{tr}(l,r)} \times log\ \frac{N_{tr_\delta}(l,r)}{N_{tr}(l,r)} \qquad (2.5)$$

У формулі (2.5) V – множина всіх можливих видів громадського транспорту.

В цілому, територія з високими значеннями транспортної ентропії, як очікується, буде привабливою для ширшого кола користувачів, тоді як найменш ентропійні райони означатимуть, що в користувачів є не багато альтернатив для пересування по місту, використовуючи громадський транспорт.

2.2.4 Принципи побудови регресорів для даних з OSMnx

Висуваємо гіпотезу, про те, що будова міста важлива при рекомендації по розташуванню громадського закладу. Для цього застосуємо мережевий аналіз. Даний підхід побудований на основі теорії графів, що в свою чергу є відгалуженням дискретної математики.

Граф – це абстрактне представлення набору елементів і зв'язків між ними (Trudeau, 1994). Кількість вершин у графі (називається степенем графа) зазвичай представлено як n, а число ребер – m. Дві вершини є суміжними, якщо їх зв'язує ребро. Два ребра суміжні, якщо вони з’єднані одною і тою ж вершиною. Ребра неорієнтованого графа вказують взаємно в обох напрямках, але орієнтований граф або орграф має конкретний напрямок ребра. Петля – це ребро, яке з'єднує один вузол з собою. Графи також можуть мати паралельні ребра між однаковими двома вузлами. Такі графи називаються мультиграфами

Шлях – це впорядкована послідовність ребер, яка з'єднує деякі впорядковані послідовності вершин. Два шляхи є внутрішньо непересічними, якщо вони не мають спільних вершин, крім кінцевих точок. Ребра зваженого графа мають ваговий атрибут для визначення величини або якості зв’язку між двома вершинами. Відстань між двома вершинами – це кількість ребер у шляху між ними, тоді як зважена відстань – це сума вагових атрибутів ребер на шляху.

Хоча граф є абстрактним математичним представленням елементів і зв'язків між ними, ми розглядаємо мережу як граф реального світу. Мережі успадковують термінологію теорії графів. Відомі приклади включають соціальні мережі (де вузли – люди, а ребра – їхні міжособистісні відносини) та Інтернет (де вузли – це веб-сторінки, а ребра – гіперпосилання, які вказують один на інший). Так можна представити і вуличні мережі в якості графу, представляючи перехрестя як вершини, а сегменти вулиць як ребра. Аналізуючий такий граф можна віднайти багато закономірностей, що можуть бути корисними для рекомендації щодо оптимального розташування громадського закладу.

У цій роботі побудуємо такі ознаки графу як щільність вулиць, щільність перехресть, середня заокругленість вулиць та центральність цільової точки в обумовленому регіоні аналізуючи граф вуличної мережі в околі заданого радіусу r від заданої цільової точки.

Щільність вулиць (*street density*) будемо шукати як відношення суми довжин усіх вулиць графу, що ділиться на загальну площу, що охоплює граф у квадратних кілометрах. Формально:

$$street\ density = \frac{\sum_{m \in M} weight(m)}{area} \quad (2.6)$$

У формулі (2.6) M – множина всіх ребер, $weight(m)$ – функція, що повертає вагу ребра $m \in M$, $area$ – площа, яку покриває граф у квадратних кілометрах.

Щільність перехресть (*intersection density*) будемо шукати як відношення кількості усіх перехресть графу, що ділиться на загальну площу, що охоплює граф у квадратних кілометрах. Формально:

$$intersection\ density = \frac{n}{area}$$

Середню заокругленість вулиць (*average circuity*) знаходимо як загальна довжина вулиць поділена на суму довжин великих дуг, тобто фактичних відстаней між вершинами [12]. Де довжина великої дуги формально знаходиться як:

$$\zeta_{ab} = r \cdot arccos(sin(\Phi_a)sin(\Phi_b) + cos(\Phi_a)cos(\Phi_b)cos(|\lambda_a - \lambda_b|)) \quad (2.7)$$

У формулі (2.7) r – радіус Землі приблизно 6371 км, а Φ_a, Φ_b, λ_a, λ_b представляють географічну широту і довготу двох точок a і b у радіанах. Відстань вздовж великої дуги ζ являє собою найкоротшу відстань вздовж криволінійної поверхні Землі і є більш точною, ніж евклідова відстань. Тепер формально запишемо формулу для визначення середньої заокругленості:

$$average\ circuity\ =\ \frac{1}{m}\sum_{a,b\in M,\,a\neq b}\frac{weight(m)}{\zeta_{ab}}$$

Такий показник допомагає зрозуміти чи наявні короткі шляхи у графі, та наскільки складно людині подолати певну дистанцію в заданому районі.

Центральність цільової точки (*betweenness centrality*) вказує на важливість даної вершини у мережі вулиць. Центральність оцінює кількість найкоротших шляхів, які проходять через кожний вузол. Даний функціонал заданий в бібліотеці OSMnx. Також можемо знайти і густину забудованості (*building density*), яка знаходиться як відношення забудованої площі до загальної площі місцевості. Даний функціонал також реалізований в OSMnx.

2.2.5 Аналіз отриманих даних

Таким чином отримали набір 15 потенційних регресорів: щільність, ентропія розташування, конкурентоспроможність, популярність, щільність переміщень, величина вхідного потоку, кількість зупинок, наявність поблизу важливих транспортних вузлів, кількість маршрутів, ентропія транспорту, щільність вулиць, щільність перехресть, середня заокругленість вулиць, центральність цільової точки, густину забудованості. Будемо використовувати дану інформацію для передбачення популярності громадського закладу в заданій точці. За допомогою програмного середовища Jupyter notebook на мові програмування Python, реалізуємо функціонал для знаходження даних регресорів та знайдемо значення для 250 наявних кав’ярень у місті Києві.

Побудуємо деякі графіки з метою проаналізувати наявні дані. Швидкий спосіб отримати уявлення про розподіл кожного регресора – це подивитися на гістограми. Гістограми групують дані у групи, що належать конкретним інтервалам, що не перетинаються, та надають підрахунок кількості спостережень у кожній групі. З графіків можна швидко емпірично оцінити розподіл ознаки, а також побачити можливі викиди. З рисунку 2.3 можна зробити висновок, що шкала кожного з регресорів є досить різною, майже всі регресори мають скошеність як наприклад ентропія розташування чи кількість зупинок громадського транспорту. Також легко помітити, що багато регресорів мають домінуючі групи значень як наприклад конкурентоспроможність: значення цієї величини лежить у проміжку від -1 до 0, однак більшість кав'ярень мають значення все ж близьке до 0. Можна зробити висновок, що дані потребують певного препроцесингу перед їх використанням в алгоритмах передбачення.

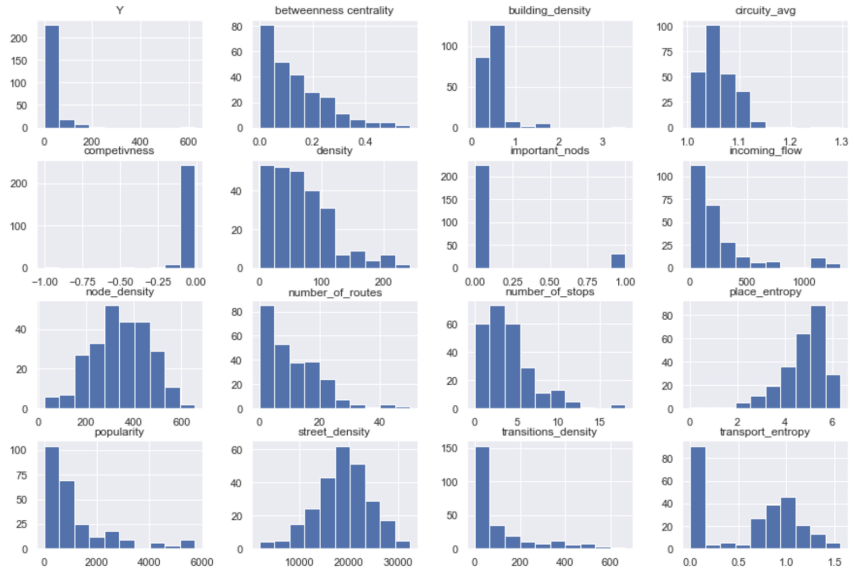


Рисунок 2.3 – Гістограми розподілу кожного з атрибутів вибірки

Діаграма розмаху або коробковий графік – засіб візуалізації в описовій статистиці даних через їх квантилі. Коробковий графік може також мати лінії, які виходять вертикально з коробки, вони вказують величину мінливості. Дані, що являються викидами, можуть бути нанесені у вигляді точок. Коробка відображає квантилі: нижня та верхня сторони коробки – це 25-й процентиль і 75-й процентиль, а смужка всередині коробки – медіана, кінці ліній, що виходять з коробки – 1-й процентиль та 99-й процентиль даних. На рисунку 2.4 представлені діаграми розмаху для кожного з атрибутів вибірки. За ними ми можемо оцінити наявність та кількість викидів, які в, наприклад, кількості зупинок громадського транспорту майже відсутні, і яких досить багато в ентропії розташування. Можна також оцінити діапазон в якому змінюється регресор.

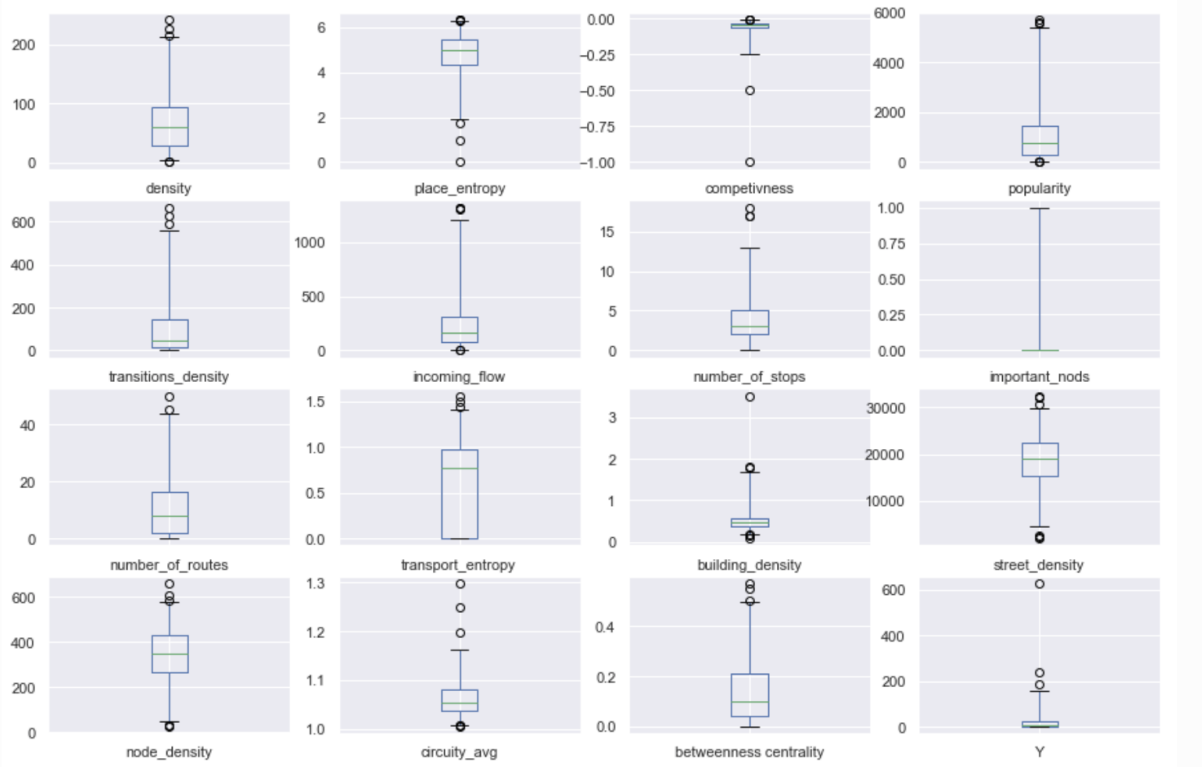


Рисунок 2.4 – Діаграми розмаху кожного з атрибутів для вибірки кав’ярень

Також було побудовано теплову карту (англ. heatmap) по матриці взаємних кореляцій між регресорами, що представлена на рисунку 2.5. За таким графіком ми можемо оцінити наскільки корельовані регресори з цільовим значенням та між собою. Можемо бачити, що регресори або слабо корельовані, або некорельовані з цільовим значенням. Також можна помітити, що існують блоки ознак, що сильно корелюють між собою. Такою групою є наприклад: популярність, щільність переміщень, та вхідний потік. Також сильну кореляцію мають між собою щільність вулиць та щільність перехресть. Інші ознаки мають здебільшого додатні невеликі кореляції. Однак наявність сильно корельованих ознак створює проблему мультиколінеарності, яку потрібно вирішити перед тим як застосовувати лінійні моделі, або ж використовувати нелінійні моделі.

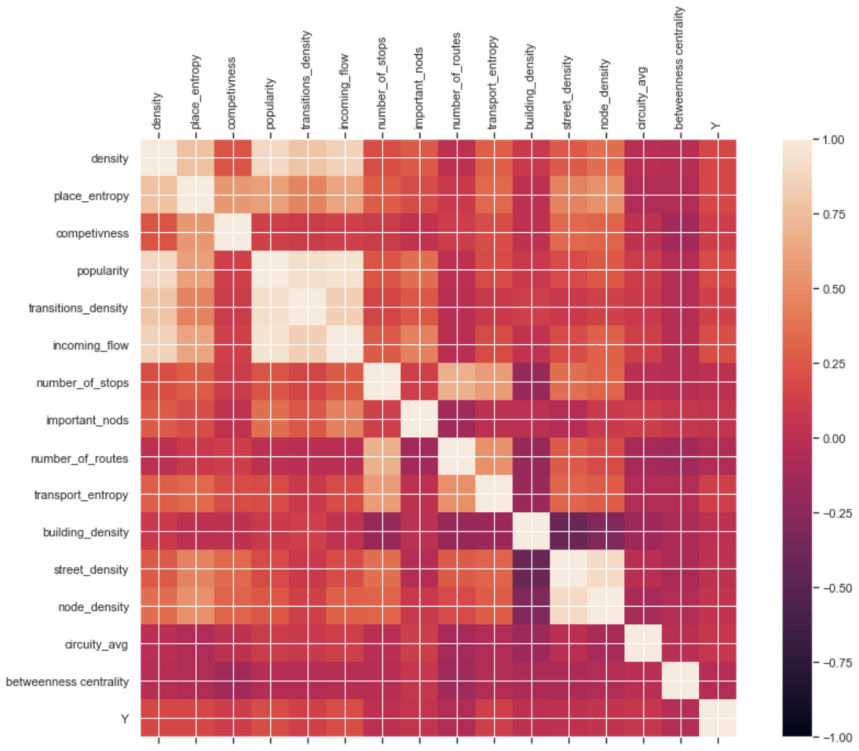


Рисунок 2.5 – Теплова карта матриці взаємних кореляцій між регресорами

2.3 Машинне навчання

Машинне навчання – спеціальний розділ методів штучного інтелекту, що спеціалізується на проектуванні та створенні (комп'ютерних) систем, здатних «навчатись» на основі даних. Такі системи розпізнають певні закономірності і після процесу навчання можуть узагальнювати знайдені закономірності для нових даних та здійснювати прогнози [13].

Вчені використовують різні алгоритми машинного навчання, щоб виявити закономірності у великих даних, і використовувати таку інформацію на благо людей. На високому рівні такі алгоритми можна класифікувати на дві групи, виходячи з того, як вони "навчаються": з учителем (англ. supervised learning) та без вчителя (англ. unsupervised learning).

Більшість машинного навчання на практиці використовує навчання з учителем. Такий підхід полягає в тому, що за наявності вхідних змінних X і вихідної змінної Y, використовується алгоритм для вивчення функції відображення від входу до виходу $Y = f(X)$. Мета полягає в тому, щоб наблизити функцію відображення настільки добре, що при наявності нових вхідних даних X_{new} можна передбачити вихідні змінні Y_{new} для цих даних. До таких алгоритмів належать лінійна регресія, дерева рішень, ліси рішень, градієнтний бустинг та інші. Навчання з учителем вимагає, щоб дані, які використовуються в алгоритмі, були вже позначені правильними відповідями [14]. У нашому випадку нам потрібно розглядати саме навчання з учителем, а точніше моделі регресії, адже вихідна змінна – кількість чекінів у громадському закладі, належить множині дійсних чисел $\mathbb{R}$ та є необмеженою.

2.3.1 Лінійні моделі в задачах регресії

Метою задачі лінійної регресії є знайти коефіцієнти лінійної залежності. Це один з найпростіших алгоритмів машинного навчання з учителем. Введемо деякі необхідні позначення:

– X – простір об'єктів,

– Y – простір відповідей,

– $x = (x^1, ..., x^d)$ – ознаки об’єкта,

– $X = (x_i, y_i)^l_{i=1}$ – навчальна вибірка,

– $a(x)$ – алгоритм, модель,

– $Q(a, X)$ – функціонал помилки алгоритму a на вибірці X.

Навчання – це процес знаходження такого алгоритму за якого мінімізується помилка: $a(x) = argmin_{a \in A}(Q(a, X))$. Простір відповідей у задачі регресії $Y = R$. Щоб навчитися вирішувати завдання регресії, необхідно задати: функціонал помилки Q – спосіб вимірювання того, добре чи погано працює алгоритм на конкретній вибірці, сімейство алгоритмів A – безліч алгоритмів, з яких вибирається найкращий, метод навчання, за яким вибирається найкращий алгоритм з сімейства алгоритмів.

Лінійний алгоритм в задачах регресії виглядає наступним чином:

$$a(x) = w_o + \sum_{j=1}^{d} w_j x^j \qquad (2.8)$$

У формулі (2.8) w_0 – це вільний коефіцієнт, x^j – ознаки об’єкта, а w_j – їх ваги. Якщо додати ще одну фіктивну ознаку, яка на кожному

об'єкті приймає значення 1, то лінійний алгоритм можна буде записати у більш простій формі:

$$a(x) = \sum_{j=1}^{d+1} w_j x^j = (w, x) \tag{2.9}$$

У формулі (2.9) позначення (w, x) означає скалярний добуток двох векторів w та x.

У якості міри помилки не може бути вибрано просте відхилення від прогнозу $Q(a, y) = a(x) - y$, так як в цьому випадку мінімум функціоналу не досягатиме при правильній відповіді $a(x) = y$. Найпростіший спосіб це виправити є взяти модуль відхилення: $|a(x) - y|$.

Але функція модуля не є гладкою тому для її оптимізації незручно використовувати градієнтні методи. Тому в якості міри помилки зазвичай обирається квадрат відхилення: $(a(x) - y)^2$. Функціонал помилки, іменований середньоквадратичної помилкою алгоритму, задається наступним чином:

$$Q(a, x) = \frac{1}{l} \sum_{i=1}^{l} (a(x_i) - y_i)^2$$

У випадку лінійної моделі такий функціонал можна переписати у вигляді функції (оскільки тепер Q залежить від вектора, а не від функції) помилок: $Q(w, x) = \frac{1}{l} \sum_{i=1}^{l} ((w, x_i) - y_i)^2$.

Для навчання моделі лінійної регресії, тобто налаштування її параметрів будемо розв’язувати таку задачу:

$$Q(w,x) = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2 \to min_w$$

Слід нагадати, що в число ознак входить також константа, що дорівнює 1 для всіх об'єктів з вибірки, що дозволяє виключити константну складову в останньому співвідношенні.

Перепишемо раніше згадані співвідношення в матричної формі. Матриця «об'єкти-ознаки» X складена з признакових описів усіх об'єктів з навчальної вибірки: $X = (x_{11}, ...x_{1d}, ..., x_{l1}, ...x_{ld})$. Таким чином, в елементі ij матриці X записано значення j-ї ознаки для i-го об'єкта навчальної вибірки. Також знадобиться вектор відповідей y, який складено з істинних відповідей для всіх об'єктів: $y = (y_1 ... y_l)$. У цьому випадку середньоквадратична помилка може бути переписана в матричному вигляді:

$$Q(w,X) = \frac{1}{l}\|Xw - y\|^2 \to min_w$$

Досить просто знайти аналітичний розв'язок задачі мінімізації: $w^* = (X^T X)^{-1} X^T y$. Але для знаходження такого розв'язку необхідно знаходити обернену матрицю. Така операція вимагає d^3 операцій у разі наявності d ознак, що створює обчислювальну складність. Також чисельний спосіб знаходження оберненої матриці не може бути застосований в деяких випадках, наприклад коли матриця погано обумовлена.

Інший спосіб знайти рішення – використовувати чисельні методи оптимізації. Нескладно показати, що середньоквадратична помилка – це

опукла і гладка функція. Опуклість гарантує існування лише одного мінімуму, а гладкість – існування вектора градієнта в кожній точці. Це дозволяє використовувати метод градієнтного спуску.

При використанні методу градієнтного спуску необхідно вказати початкове наближення. Найпростіший спосіб це зробити – визначити всі ваги рівними нулю: $w^0 = 0$. На кожній наступній ітерації $t = 1, 2, 3, ...,$ від наближення, отриманого на попередній ітерації w^{t-1}, віднімається вектор градієнта у відповідній точці w^{t-1}, помножений на деякий коефіцієнт η_t, що називається кроком: $w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, X)$. Зупинити ітерації потрібно тоді, коли настає збіжність. Збіжність можна визначати по-різному. В даному випадку визначаємо збіжність наступним чином: ітерації слід завершити, якщо різниця двох послідовних наближень не дуже велика: $\|w^t - w^{t-1}\| < \varepsilon$ [15].

2.3.2 Модифіковані лінійні моделі в задачах регресії

Регресія Ridge – це метод, який використовується, коли у вхідних даних спостерігається мультиколінеарність, тобто незалежні змінні сильно корельовані. Додаючи ступінь зміщення до оцінок регресії ми таким чином враховуємо значення середнього квадрата помилок. Метод варто використовувати, якщо: сильно обумовлена $X^T X$, сильно розрізняються власні значення або деякі з них близькі до нуля, в матриці X є майже лінійно залежні стовпці. Такий метод також відомий як регресія з L_2 регуляризацією. Ridge вирішує проблему мультиколінеарності через параметр λ. Формально в даному методі ми мінімізуємо такий функціонал помилки:

$$Q(w,X) = \frac{1}{l}||Xw - y||^2 + \lambda||w||^2 \to min_{w \in R^d}$$

Таким чином враховуючи не тільки похибку, а й пеналізуючи за великі коефіцієнти w.

Подібно до регресії Ridge, Lasso також пеналізує абсолютне значення коефіцієнтів регресії. Крім того, він здатний знижувати нестабільність і підвищувати точність моделей лінійної регресії. Lasso відрізняється від Ridge тим, що вона використовує абсолютні значення у штрафній функції, а не квадрати. Це призводить до того, що деякі оцінки параметрів виявляються рівними нулю. Чим більший штраф застосовується, тим більше оцінки зменшуються до абсолютного нуля. Формально в даному методі ми мінімізуємо такий функціонал помилки:

$$Q(w,X) = \frac{1}{l}||Xw - y||^2 + \lambda||w||_{l_1} \to min_{w \in R^d}$$

ElasticNet є гібридним методом Lasso і Ridge регресій. Він використовує L_1, L_2 регуляризації. ElasticNet корисна, коли існує декілька корельованих ознак. Lasso працює так, що вибере одну з них навмання, а ElasticNet обирає обидва. Формально в даному методі ми мінімізуємо такий функціонал помилки, який є гібридом двох попередніх:

$$Q(w,X) = \frac{1}{l}||Xw - y||^2 + \lambda||w||_{l_1} + \lambda||w||^2 \to min_{w \in R^d}$$

2.3.3 Нелінійні моделі в задачах регресії

У випадку взаємозалежних факторів та складних залежностей на практиці часто застосовуються алгоритми, що будуються по принципу дерева рішення. Також важливою перевагою є легка інтерпретація результату роботи алгоритму, так як когнітивний процес прийняття рішень у людей є досить схожий. Дерева рішень – це сімейство алгоритмів, яке дуже сильно відрізняється від лінійних моделей, але в той же час грає важливу роль в машинному навчанні.

Дерева рішення зазвичай представляють собою бінарні дерева, де в кожній внутрішній вершині записана умова, а в кожному листі дерева – прогноз. Варто зазначити, що не обов'язково вирішальне дерево повинно бути бінарним. Умови у внутрішніх вершинах вибираються вкрай простими. Найпоширеніший варіант – це перевірити, чи лежить значення деякої ознаки x_j лівіше, ніж заданий поріг t: $[x^j \le t]$.

Прогноз в листі є дійсним числом, якщо вирішується задача регресії. Нехай вирішується задача регресії з однією ознакою, за якою потрібно відновити значення цільової змінної. Не дуже глибоке дерево відновлює залежність приблизно так як показано на рисунку 2.6:

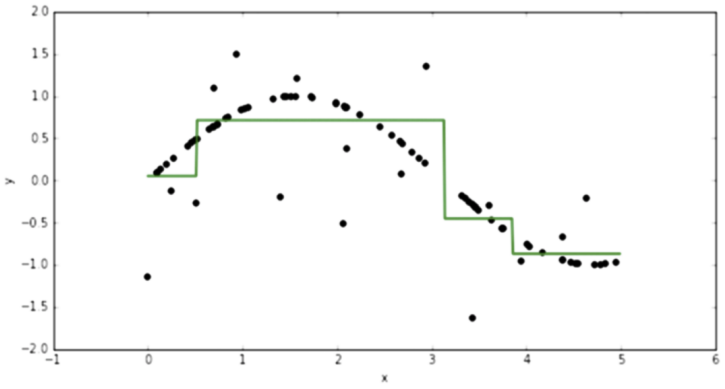


Рисунок 2.6 – Використання вирішальних дерев в задачах регресії

Відновлена залежність буде кусочно-постійна. При збільшенні глибини дерева можна майже ідеально вивчити вхідні дані, і таким чином перевчитися. Якщо дерево підігнати під викиди то його якість вже буде не такою хорошою.

У машинному навчанні застосовується жадібний спосіб побудови дерева рішень від кореня до листя. Спочатку вибирається корінь, який розбиває вибірку на дві підвибірки. Потім розбивається кожен з нащадків цього кореня і так далі. Дерево розгалужується до тих пір, поки цього не буде достатньо. Нехай в вершину m потрапило безліч X_m об'єктів з навчальної вибірки. Параметри умови $[x^j \le t]$ будуть обрані так, щоб мінімізувати критерій помилки $Q(X_m, j, t)$:

$$Q(X_m, j, t) \to min_{j,t}$$

Параметри j і t можна підбирати перебором. Існує кінцеве число ознак, а з усіх можливих значень порога t. Можна розглядати тільки ті, при яких виходять різні розбиття. Можна показати, що таких значень параметра t стільки, скільки різних значень ознаки x^j на навчальній вибірці [15]. Після того, як параметри були обрані, безліч X_m об'єктів з навчальної вибірки розбивається на дві підмножини: $X_l = \{x \in X_m | [x^j \le t]\}$, $X_r = \{x \in X_m | [x^j > t]\}$, кожна з яких відповідає своїй дочірній вершині. Запропоновану процедуру можна продовжити для кожної з дочірніх вершин: в цьому випадку дерево буде все більше і більше заглиблюватися. Такий процес рано чи пізно повинен зупинитися, і чергова дочірня вершина буде визначена листком, а не розділена навпіл. Цей момент визначається критерієм зупинки. Існує багато різних варіантів критерію зупинки, наприклад:

– Якщо в вершину потрапив тільки один об'єкт навчальної вибірки або всі об'єкти належать одному класу (в задачах класифікації), далі розбивати не має сенсу.

– Можна також зупиняти розбиття, якщо глибина дерева досягла певного значення [15].

Якщо якась вершина була оголошена листом, потрібно визначити прогноз, який буде міститися в даному листі. У цей лист потрапила деяка підвибірка X_m вихідної навчальної вибірки і потрібно вибрати такий прогноз, який буде оптимальний для даної підвибірки. У задачі регресії, якщо функціонал – середньоквадратична помилка, оптимально давати усереднену відповідь по цій підвибірці:

$$a_m = \frac{1}{|X_m|} \sum_{i \in X_m} y_i$$

Вибір оптимального розбиття при побудові вирішального дерева відбувається за критерієм інформативності. Критерій помилки записується формально визначений так:

$$Q(X_m, j, t) = \frac{|X_l|}{|X_m|} H(X_l) + \frac{|X_r|}{|X_m|} H(X_r) \to min_{j,t}$$

Можемо бачити, що вираз складається з двох доданків, кожен з яких відповідає своєму листу. Функція $H(X)$ називається критерієм інформативності. Найменше значення критерію відповідає найменшому розкиду відповідей в X. У випадку регресії розкид відповідей – це дисперсія, тому критерій інформативності в задачах регресії записується в такий спосіб [15]:

$$H(X) = \frac{1}{|X|}\sum_{i\in X}(y_i - \overline{y}(X))^2, \text{ де } \overline{y} = \frac{1}{|X|}\sum_{i\in X} y_i$$

2.3.4 Композиція дерев. Випадковий ліс

Дерева рішень занадто легко перевчаються під навчальну вибірку і виходять непридатними для побудови прогнозів. Однак такі алгоритми дуже добре підходять для об'єднання в композиції і побудови одного неперенавченного алгоритму на основі великої кількості вирішальних дерев. Композиція – це об'єднання N алгоритмів $b_1(x), ..., b_N(x)$ в один. Ідея полягає в тому, щоб навчити алгоритми $b_1(x), ..., b_N(x)$, а потім усереднити отримані від них відповіді: $a(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x)$. Цей вираз безпосередньо і буде відповіддю в задачі регресії [15].

Щоб побудувати композицію, потрібно спочатку навчити N базових алгоритмів, причому їх не можна навчати на всій навчальній вибірці, так як в цьому випадку вони виходять однаковими, і в їх усередненні не буде ніякого сенсу. У таких випадках використовують рандомізацію, тобто навчання базових алгоритмів на різних підвибірках навчальної вибірки, що значно підвищує різноманітність базових алгоритмів.

Так званий бутстрап – один з популярних підходів до побудови підвибірок. Він полягає в тому, що з навчальної вибірки довжини l вибирають з поверненням l об'єктів. При цьому нова вибірка також матиме розмір l, але деякі об'єкти в ній будуть повторяться. Можна показати, що в бутстрапірованной вибірці буде міститися в середньому 63% унікальних об'єктів вихідної вибірки [16].

Щоб побудувати випадковий ліс з N дерев рішень, необхідно:

1) Побудувати за допомогою бутстрапа N випадкових підвибірок;

2) Кожна підвибірка $\hat{X}_n$ використовується як навчальна вибірка для побудови відповідного дерева рішень $b_n(x)$, причому: дерево будується так, щоб отримати складні і перенавчені вирішальні дерева;

3) Побудовані дерева об'єднуються в композицію. У задачі регресії це формально записується так: $a(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x)$.

У процесі побудови кожного дерева на етапі вибору оптимальної ознаки, за якою буде відбуватися розбиття, ця ознака шукається не серед усієї множини ознак, а серед випадкової підмножини розміру q. Випадкова підмножина розміру q вибирається заново кожний раз, коли необхідно розбити чергову вершину [15].

2.3.5 Градієнтний бустинг. XGBoost

Випадковий ліс – композиція глибоких дерев, які будуються незалежно одне від одного. Однак навчання глибоких дерев вимагає дуже багато обчислювальних ресурсів, особливо при застосуванні великих вибірок або великої кількості ознак.

Якщо обмежити глибину дерев у випадковому лісі, то вони вже не можуть визначити складні закономірності в даних. Ще одна проблема є в тому, що процес навчання дерев є не цілеспрямленим: кожне наступне дерево в композиціях не залежить від попередніх. Через це для вирішення складних завдань необхідна велика кількість дерев [15].

Вирішити ці недоліки можна з допомогою бустинга. Бустинг – це підхід до створення композицій, у яких базові алгоритми будуються

послідовно та кожен наступний алгоритм будується таким чином, щоб виправити помилки вже побудованої композиції.

Завдяки тому, що композиціонування за таким принципом є цілеспрямленим, у якості базових алгоритмів можна використовувати нескладні моделі, наприклад неглибокі дерева.

Покажемо це на конкретному прикладі задачі регресії. Використовуємо функціонал середньоквадратичної помилки: $Q(w,x) = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2$. Для початку достатньо навчити перший простий алгоритм: $b_1(x) = argmin_b \frac{1}{l}\sum_{i=1}^{l}(b(x_i) - y_i)^2$. Даний алгоритм і буде першим в побудові композиції. Другий алгоритм вже буде будуватися за принципом мінімізації помилки алгоритма $b_1(x_i) + b_2(x_i)$.

$$b_2(x) = argmin_b \frac{1}{l}\sum_{i=1}^{l}(b_1(x_i) + b(x_i) - y_i)^2 =$$

$$= argmin_b \frac{1}{l}\sum_{i=1}^{l}(b(x_i) - (y_i - b_1(x_i)))^2$$

Таким чином алгоритм $b_2(x)$ буде покращувати роботу алгоритму $b_1(x)$. Аналогічно для композиції $b_1(x_i) + ... + b_N(x_i)$ продовжуємо таким чином: $b_N(x) = argmin_b \frac{1}{l}\sum_{i=1}^{l}(b(x_i) - (y_i - \sum_{n=1}^{N-1} b_n(x_i)))^2$. Виконуємо аж поки помилка не буде нас влаштовувати.

У нашому дослідженні будемо використовувати функціонал градієнтного бустингу бібліотеки XGBoost, який є відкритим для використання. У якості базових алгоритмів там використовуються прості дерева рішень. Він є дещо модифікованим, зокрема там наявна

регуляризація, що штрафує за кількість листів у деревах, вимагаючи таким чином, щоб алгоритми були простими [17].

2.4 Висновки до розділу 2

У даному розділі розглянуто основні метрики якості для задачі рекомендації. Визначено особливості кожної з метрик. Розглянуто різні підходи до побудови моделей машинного навчання та визначено їхні особливості. Розглянуто принципи обробки даних з різних джерел та методи побудови регресорів. Обґрунтовано важливість використання кожної з ознак для побудови моделей надання рекомендації щодо геопозиціонування.

РОЗДІЛ 3 ПРОВЕДЕНА ЕКСПЕРИМЕНТІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Умови проведення експериментів

Побудова та проведення експериментів складає найважливішу частину роботи. Від даного процесу залежить коректність висновків та вибір майбутньої моделі для роботи програми. Також в даному розділі проаналізовано результати роботи різних алгоритмів та проінтерпритовано отримані результати.

3.1.1 Препроцесинг даних

У загальному процесі побудови математичної моделі, попередня обробка даних відіграє вирішальну роль. Один з перших кроків стосується нормалізації даних. Цей крок є дуже важливим при роботі з ознаками, що мають різні одиниці виміру і шкали. Наприклад, деякі математичні моделі, побудовані на даних, використовують евклідову відстань. Тому всі параметри повинні мати однакову шкалу для можливості справедливого порівняння.

Існує два основних, добре відомих способів для масштабування даних. Нормалізація, яка масштабує всі числові змінні в діапазоні [0,1]. Формально таких функціонал записується формулою:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

З іншого боку, також використовується стандартизація набору даних. Мета такого перетворення – перетворити дані таким чином, щоб вони мали нульове середнє і одиничну дисперсію. Реалізується це таким функціоналом:

$$x_{new} = \frac{x-\mu}{\sigma}$$

Обидві ці методи мають свої недоліки. Якщо у наборі даних існують відхилення, нормалізація даних, безсумнівно, буде масштабувати дані до дуже малого інтервалу. При використанні стандартизації перетворені дані не обмежені на відміну від нормалізації.

У нашому досліджені ми також спробуємо комбінований метод препроцисингу. Пояснюється, що значення, отримані в результаті стандартизації, є безрозмірними та знаходяться на проміжку $\left[\frac{x_{min}-\bar{x}}{\sigma}, \frac{x_{max}-\bar{x}}{\sigma}\right]$ переважно в околі нуля, де $\bar{x}$ – вибіркове середнє, σ – вибіркове середньоквадратичне відхилення. Для використання такого перетворення, автор рекомендує застосовувати додаткове перетворення. Формально такий ланцюг перетворень визначається так:

$$x \to \frac{x-\bar{x}}{\sigma} \to \frac{1}{1+e^{-(\frac{x-\bar{x}}{\sigma})}}$$

Таке перетворення, окрім належності інтервалу [0,1], гарантує і більш рівномірний розподіл значень [18].

3.1.2 Поділ даних для експерименту

У машинному навчанні є декілька підходів до того як поділити дані для проведення експерименту. Перший найпростіший метод це просто поділити дані на три частини. Одна з них буде використовуватися для тренування моделі – тренувальна вибірка. Друга – для перевірки роботи моделі та підбору гіпер параметрів. Третя – для перевірки моделі на

умовно реальних даних, що є незалежними від попередніх двох частин, які використовувалися при тренуванні. Продемонструємо такий експеримент на рисунку 3.1.

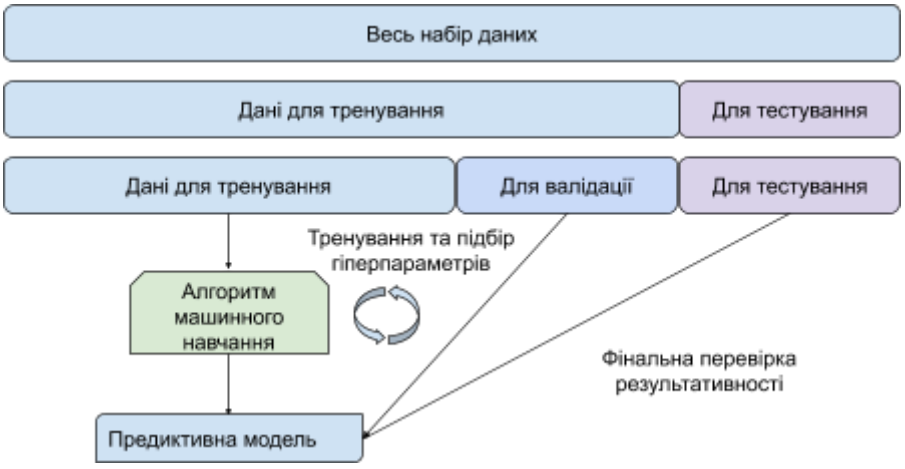


Рисунок 3.1 – Принцип поділу даних на три частини

Важливим моментом є те, у якій пропорції здійснювати поділ даних. У нашій роботі, виходячи з логічних міркувань, будемо використовувати 80:10:10. При наявності величезних даних рекомендують використовувати 98:1:1, проте це не наш випадок [19].

Ще одним, методом є крос валідація. Ідея полягає в тому, щоб поділити дані на декілька однакових за розміром частин, для прикладу припустимо, на п’ять рівних частин. Далі вчимо модель на всіх частинах за винятком однієї, на якій ми тестуємо результати роботи моделі. Далі виконуємо таку ж операцію відкладаючи інші частини. Продемонструємо такий метод на рисунку 3.2.

Далі отримавши результати кожного з експериментів, можна знайти середнє значення, щоб оцінити в загальному результативність моделі та також за дисперсією можна оцінити наскільки модель є стабільною, тобто показує однаковий результат на різних наборах даних.

У рамках нашого дослідження зосередилися саме на першому підході, саме через брак даних та легкість інтерпритації результату.

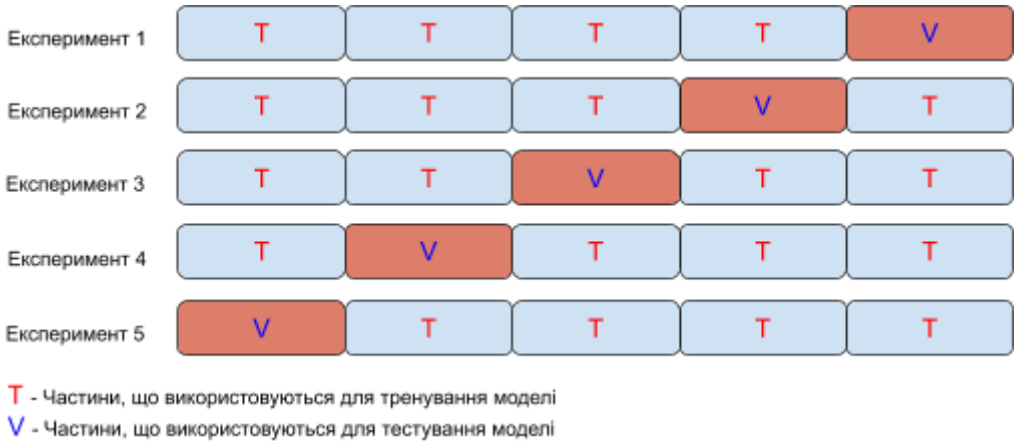


Рисунок 3.2 – Принцип методу кросвалідації

3.1.3 Умови проведення експерименту

Дизайн експерименту є важливим процесом для будь-якого вченого, інженера або статистика, який планує провести експериментальний аналіз. Це завдання стало критично важливим в епоху швидкого розвитку науки про дані та пов'язаного з нею статистичного моделювання та машинного навчання.

У своїй найпростішій формі науковий експеримент спрямований на прогнозування результату шляхом внесення зміни передумов, який представлений однією або кількома незалежними змінними, які також називаються "вхідними змінними" або ознаками. Більш незалежні змінні, як правило, передбачають зміну однієї або декількох залежних змінних, які також називаються “вихідними змінними” або цільовим значенням. Основними завданнями в проектуванні експерименту є забезпечення його адекватності, надійності та повторюваності.

У нашому завданні модель представляє собою декілька етапів підготовки і обробки даних. На першому етапі відбувається агрегація даних з глобального набору даних. Далі отримавши конкретну вибірку відбувається препроцесінг даних. У нашому випадку є три альтернативи виконання цього етапу: стандартизація, нормалізація та модифікований метод. Далі відбувається етап поділу даних для того, щоб потім можна було перевірити результативність. У нашому випадку ми будемо ділити дані на три частини в пропорції 80:10:10, де перша буде використана для навчання, друга для валідації та підбору гіперпараметрів, третя – для фінальної перевірки на незалежних даних. Важливо, щоб три частини мали приблизно однакові розподіли. У випадку коли маємо справу з геопросторовими даними, потрібно, щоб заклади з кожної з частин були рівномірно розповсюджені у просторі. Перевіряти це будемо візуалізуючи результати поділу. Після коректного поділу даних відбувається моделювання. Під час цієї частини експерименту за допомогою однієї з семи розглянутих раніше математичних моделей буде відбуватися навчання і, що також дуже важливо, оптимізація гіперпараметрів. В кінці відбувається фінальна перевірка моделі на тестовій вибірці. Мета є порівняння алгоритмів та вибір найкращого для нашої задачі. Також порівнювати ми будемо з елементарним алгоритмом, який обирає випадковий порядок.

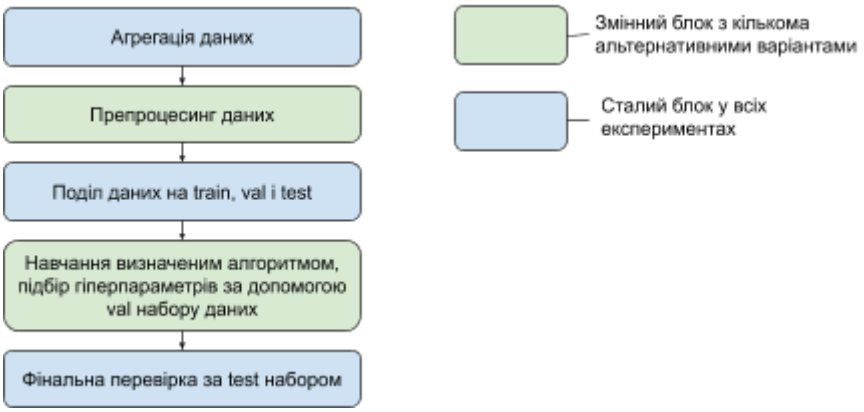


Рисунок 3.3 – Процес проведення експерименту

3.2 Проведення експериментів

На попередніх етапах роботи, було зібрано дані та отримано інформацію про 250 кав'ярні. Попередньо зробивши один із варіантів препроцесингу, зробимо поділ даних на три частини та провізуалізуємо результати на рисунку 3.4. Варто зазначити, що поділ будемо робити випадковим чином.

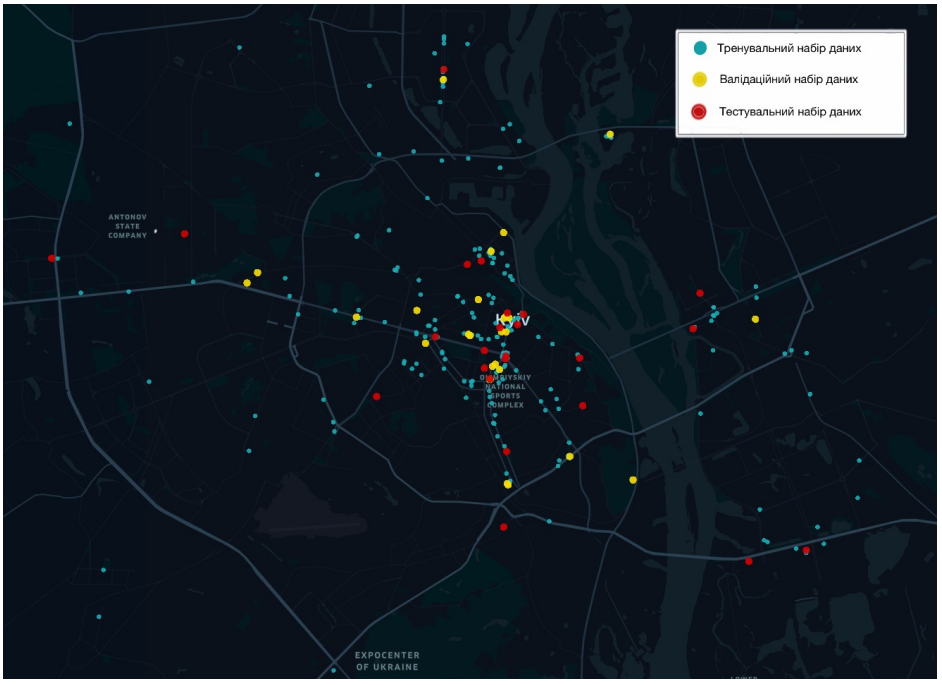


Рисунок 3.4 – Розподіл даних на три частини

Отримали набір даних для тренування, що містить 200 елементів, валідаційний та тестовий набір, що містять по 25 елементів кожен.

Далі, використовуючи тренувальну вибірку навчаємо модель та підбираємо найкращі параметри методом перебору по сітці. При цьому перевіряємо результативність використовуючи валідаційний набір даних. Далі, знайшовши модель, яка показує найкращі результати на валідаційному наборі для конкретного алгоритму, перевіряємо роботу програми на відкладеній тестовій вибірці і робимо висновок про її придатність для подальшого використання. Для того, щоб модель була придатною, її результати на тестовому наборі повинні не сильно відрізнятися від значень на валідації. Якщо така різниця велика, то має місце перенавчання. Якщо така тенденція зберігається для всіх алгоритмів без виключення – це може означати, що тестова і валідаційна вибірки сильно між собою відрізняються. Хоча в ідеальних умовах це неприпустимо, проте в рамках нашого експерименту, так як у нас є реальна інформація про малу кількість закладів, при тому, що робота йде з геопросторовими даними, ми припускаємо існування такої різниці. Також, так як метрики ранжування є метриками відносними, тому основним показником для оцінки моделі є її порівняння з базовим алгоритмов, що реалізує найпростіші міркування.

Наведемо результати для категорії кав’ярень у таблиці 3.1 та провізуалізуємо їх на рисунку 3.5. Важливо є те, що оцінку проводимо відразу за трьома різними метриками, які кожна по-своєму характеризують алгоритми.

Таблиця 3.1 – Результати експерименту з розділенням даних на три частини

Algorithm	**NDCG@10**	**AP@10**	**Precision@10**
Baseline	0.238	0.091	0.3
Decision Tree	0.696	0.523	0.6
Random Forest	0.782	0.647	0.7
SGD	0.494	0.291	0.4
Ridge	0.597	0.394	0.5
Lasso	0.593	0.388	0.5
ElasticNet	0.520	0.319	0.4
XGBRegressor	0.684018	0.495	0.6

У якості базового, найпростішого алгоритму, було обрано проста випадкова перестановка. Зроблено це з припущення, що з наданих завідомо придатних для громадського закладу позицій, рішення буде зроблене випадковим чином. Також випадково було здійснено і поділ даних, для того, щоб експеримент був максимально наближеним до реальних умов.

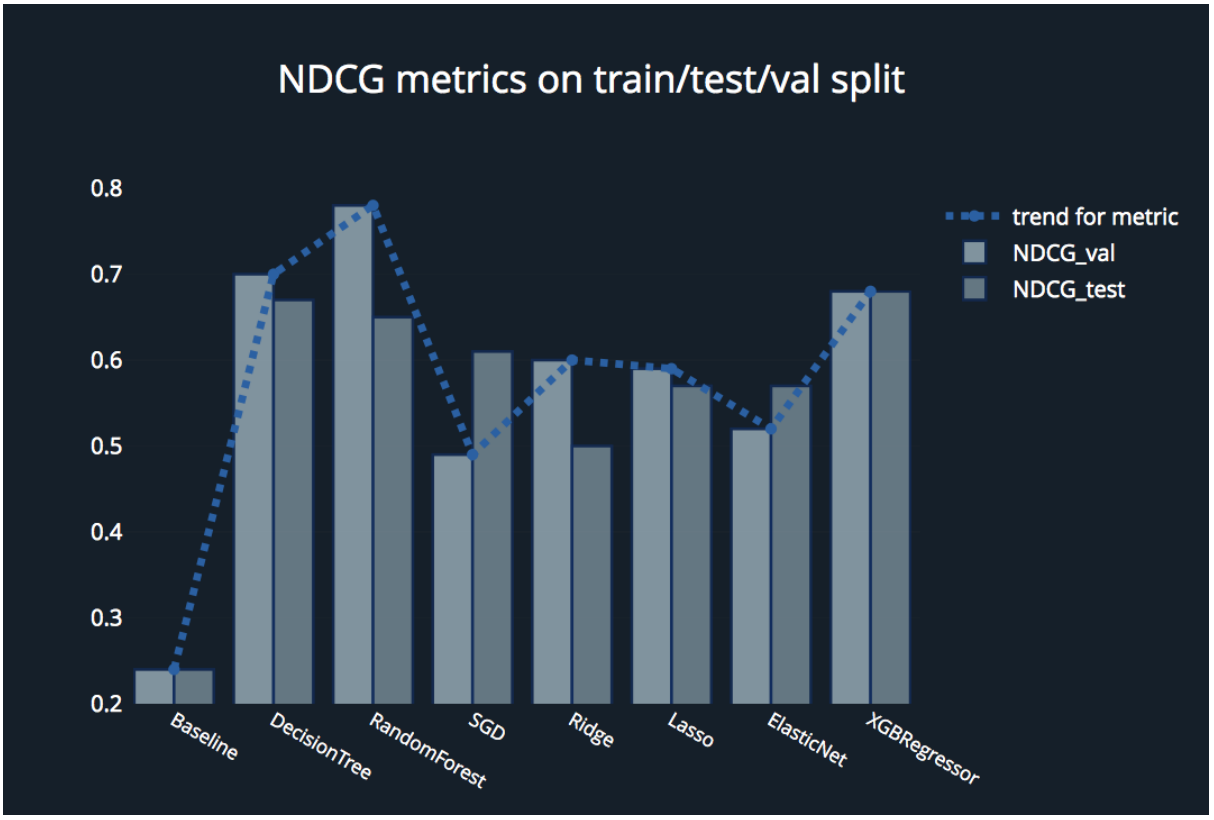


Рисунок 3.5 – Результати експерименту з розділенням даних на три частини для метрики NDCG@10

З результатів експерименту ми можемо бачити, що найкраще показують себе алгоритми, побудовані на принципі дерева прийняття рішень. А саме, найкращим алгоритмом на валідації у такому експерименті виявився випадковий ліс (англ. Random Forest). Однак видно, що існує проблема перенавчання, зокрема найбільше на випадковому лісі. Однак, схожий за принципом роботи XGBRegressor, показує мінімальну різницю в результатах на тестовому та валідаційних наборах, при цьому демонструючи досить високий результат, що значно перевершує базовий алгоритм. Беручи до уваги те, що важливим показником рекомендаційних систем є їхня надійність, рекомендованим для використання є саме бустингова модель.

Також було проведено експеримент з кросвалідацією. Результати експериментів наведено в таблиці 3.2 та на рисунку 3.6.

Таблиця 3.2 – Результати експерименту з кросвалідацією

Algorithm	**NDCG@10**	**AP@10**	**Precision@10**
Baseline	0.237584	0.091071	0.3
Decision Tree	0.556922	0.523333	0.6
RandomForest	0.554524	0.423889	0.6
SGD	0.495899	0.113333	0.3
Ridge	0.526075	0.370833	0.5
Lasso	0.564061	0.375000	0.5
ElasticNet	0.550480	0.281667	0.4
XGBRegressor	0.574686	0.281667	0.5

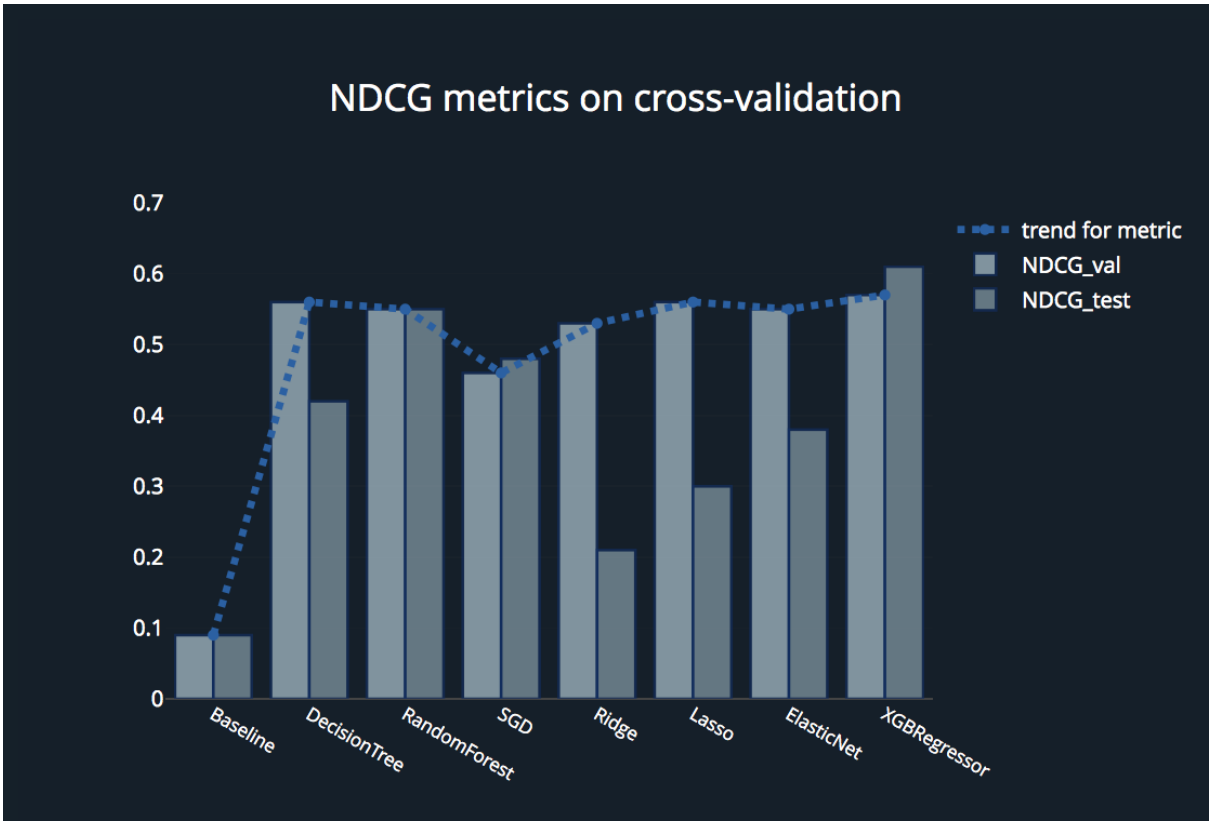


Рисунок 3.6 – Результати експерименту з кросвалідацією для метрики NDCG@10

Використання такого підходу значно зменшує ймовірність перенавчання під певну конкретну вибірку, так як оцінювання проводиться одночасно на декількох наборах даних. За кросвалідацією підтверджуються результати проведені у попередньому експерименті. А саме, найкращий результат мають алгоритми, що базуються на композиції дерев прийняття рішень. Проте можемо бачити, що тепер у випадку випадкового лісу результат на валідації значно гірший, однак майже співпадає з результатом на тестовій вибірці. Це означає, що отриманий результат є валідним, а модель надійною. Результати лінійних моделей навіть з оптимальною попередньою обробкою та різними варіантами

регуляризації не дозволяють досягти результатів отриманих нелінійними моделями.

Тому для практичного вирішення такого завдання краще використовувати нелінійні моделі, такі як, наприклад, випадковий ліс або нейронні мережі. Такі наші спостереження повністю співпадають з результатами роботи британських вчених, які здійснювали подібні дослідження раніше [7].

3.3 Демонстрація та інтерпретація результатів

У проектах, результатом яких є рекомендація або консультація, важливим етапом є демонстрація результату. Висновок, має бути ефективно донесений замовнику та також має бути зрозумілим та логічним. Тому, також було здійснено розробку інструменту для демонстрації результатів дослідження та їх інтерпретації.

3.3.1 Розробка графічного інтерфейсу

Для ефективної демонстрації всіх етапів дослідження та результатів було розроблено графічний інтерфейс. Зроблено це засобами відкритої бібліотеки flask на мові програмування Python з використанням HTML, CSS, JS вкраплень. Графічні матеріали були підготовлені з використанням бібліотек plotly та kepler.gl.

Інтерфейс розрахований для демонстрації результатів для різнотипних категорій закладів. Вибір здійснюється використовуючи інтерфейс зображений на рисунку 3.7.

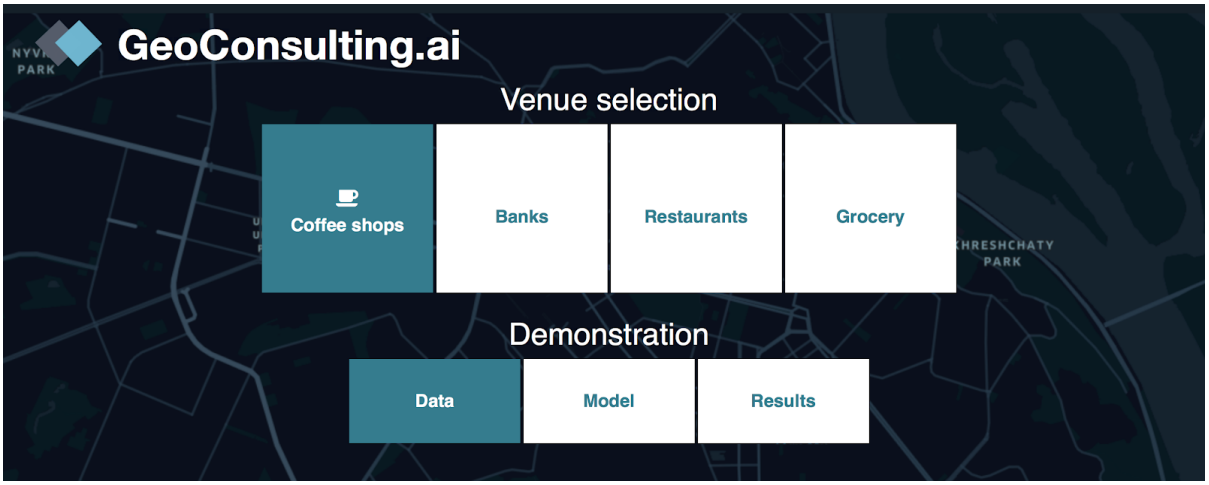


Рисунок 3.7 – Графічний інтерфейс для демонстрації результатів

Для кожної категорії реалізовано можливість візуалізації загальної інформації про наявні дані, наприклад дані про наявні заклади в місті Києві на рисунку 3.8, дані про географічні переміщення людей в розрізі часу на рисунку 3.9, геопросторовий розподіл наявних закладів на рисунку 3.10. результативність моделей для різного роду експериментів та картографічне відображення рекомендацій на рисунку 3.11.

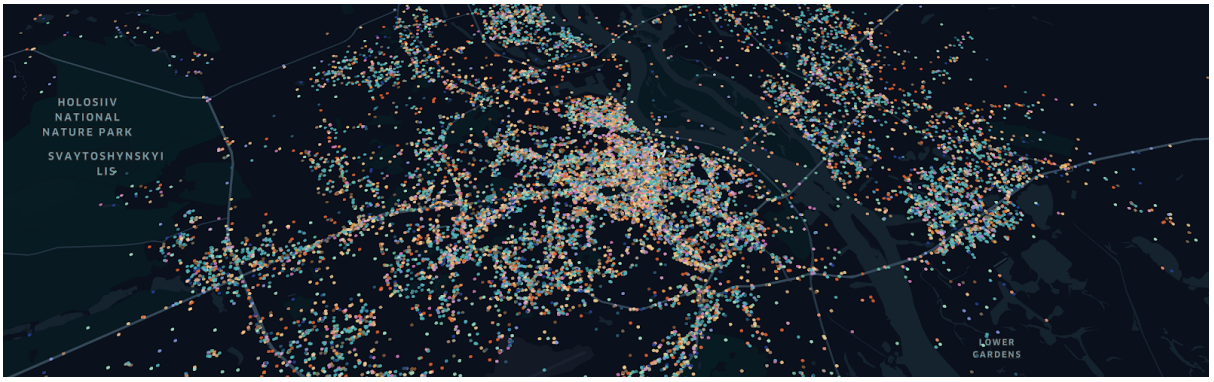


Рисунок 3.8 – Наявні заклади в місті Києві

Рисунок 3.9 – Дані про географічні переміщення людей в розрізі часу

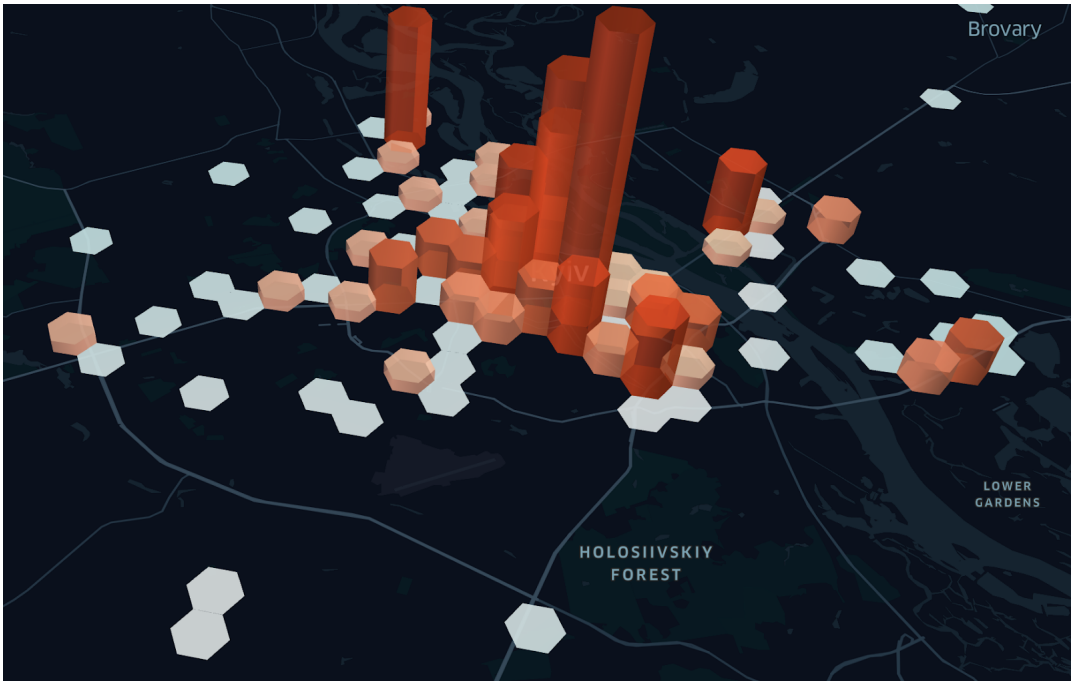


Рисунок 3.10 – Геопросторовий розподіл наявних закладів

Рисунок 3.11 – Результативність рекомендаційних моделей

За рисунком 3.8 та рисунком 3.9 можна наочно оцінити те, що ми маємо справу з великою кількістю інформації та те, що ця інформація коректна та логічна. За рисунком 3.10 бачимо, що найбільша кількість кав'ярень розташована у центрі міста, також є дві області поза центром, де спостерігається скупчення такого роду закладів. Це можна пояснити наявністю там популярних торгово-розважальних центрів. За допомогою рисунку 3.11 можемо зрозуміти, які заклади використовувалися для тренування, та оцінити наскільки рекомендації, що представлені зеленим кольором співпали з фактичними найпопулярнішими місцями, які представлені червоним кольором. Бачимо, що модель правильно визначила сім з десяти ка'ярень. Також важливим моментом є аналіз того, де ми помиляємося. Спостерігається тенденція того, що місця, які є помилково

відрекомендовані географічно знаходяться дуже близько до тих, що є фактично найкращими, що може в свою чергу сприяти помилці моделі.

3.3.2 Інтерпретація результатів

Найпростішим методом інтерпретації результатів є пройти етапи того, як модель зробила таке рішення. У лінійних моделях таке дослідження зводиться до перевірки коефіцієнтів регресії. Так, якщо ми привели всі ознаки до однакової шкали, то значення коефіцієнтів регресії при кожній з ознак будуть відповідати її конкретному внеску у висновок моделі.

З моделями побудованими по принципу випадкових лісів, все трохи складніше. Випадковий ліс будують багато індивідуальних дерев рішень під час навчання. Прогнози з усіх дерев об'єднуються, щоб зробити остаточний прогноз – середнє арифметичне по кінцевому листку у випадку регресії. Оскільки вони використовують набір результатів для прийняття остаточного рішення, їх називають методами ансамблю.

Важливість ознаки обчислюється за зменшенням суми за вузлами, де розбиття відбулося за заданою ознакою, нормуючи на кількість елементів, що брали участь у кожному з розбиттів. Чим вище значення, тим важливішою є функція.

У нашому дослідженні ми визначили важливість вузлів моделі випадкового лісу та встановили, що найважливішими ознаками у роботі моделі є конкурентоспроможність, щільність транспорту, а також вхідний потік та центральність цільової. Результати наведені на рисунку 3.12. Для інтерпритації результату, побудовано теплову карту ознаки на яку

накладено запропоновані точки та перевірено правдоподібність рекомендації. Результат продемонстровано на рисунку 2.13.

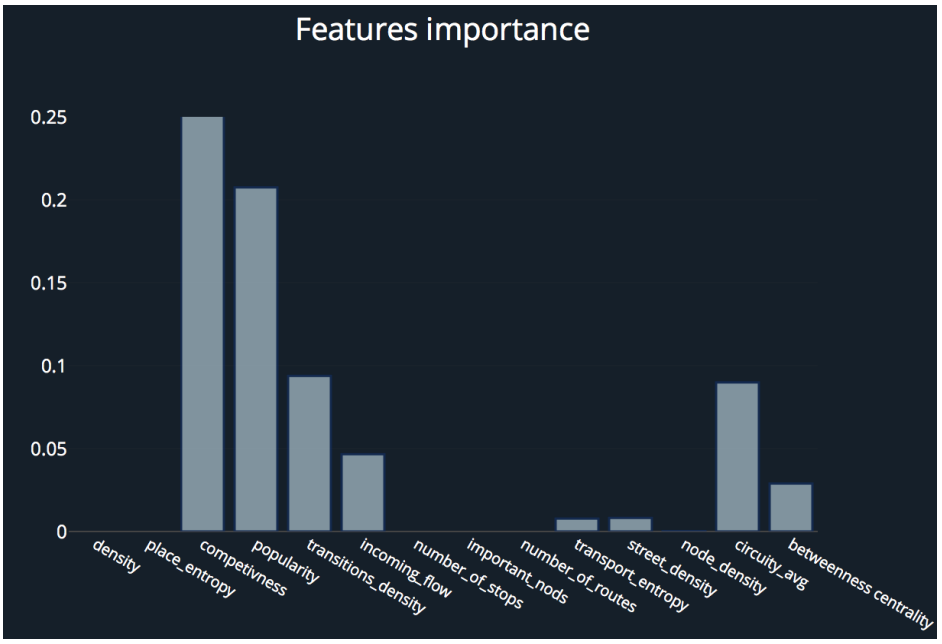


Рисунок 3.12 – Важливість ознак для найкращої моделі випадкового лісу

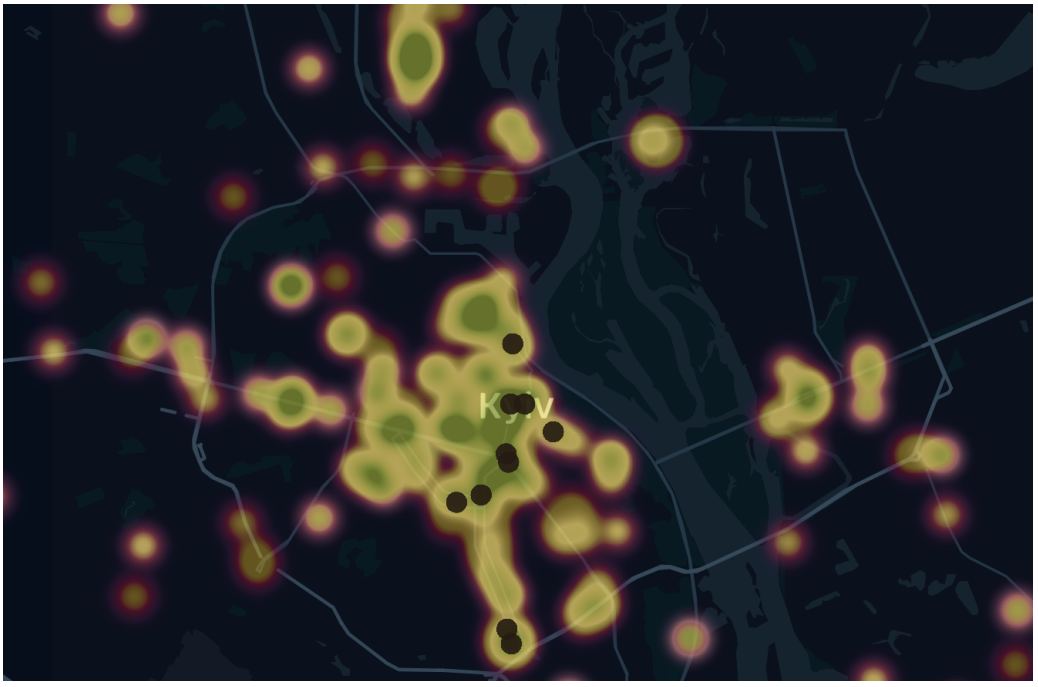


Рисунок 3.13 – Теплова карта ознаки конкурентоспроможності та вхідного потоку для інтерпритації результату рекомендації

3.4 Висновки до розділу 3

У даному розділі визначено умови проведення експериментів. Проведено експерименти, виявлено те, що моделі машинного навчання придатні для задачі рекомендації. Визначено, що найкраще в даній задачі підходять нелінійні моделі побудовані з використанням дерев прийняття рішень. Проведено експеримент з реальними кав’ярнями міста Києва та за заданим набором даних визначено найкраще місце розташування для десяти нових кав’ярень. Проінтерпритовано результат та віднайдено фактори, що мають найбільший вплив на результат роботи програми. Також було розроблено графічний інтерфейс та картографічні візуалізації для ефективної демонстрації та інтерпретації результатів.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСТНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Створення програмного продукту вимагає великих фінансових та трудових затрат. Тому у даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для рекомендації найкращого розміщення громадських закладів, що дозволяє обрати найбільш оптимальний варіант його створення та підтримки. Даний продукт розроблений на мові програмування Python в середовищі Jupyter notebook в якості самостійної інтерфейсу для аналітика. Програмний продукт в першу чергу призначено для визначення найкращих точок для розміщення, базуючись на моделях машинного навчання.

Далі наведено аналіз альтернатив реалізації програми з метою вибору оптимальної, з огляду на економічні фактори, так і на характеристики продукту, що впливають на результативність роботи і на його сумісність з апаратним забезпеченням. Для цього використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це методологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії.

Фактично цей метод працює за таким алгоритмом:

1) Визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат;

2) Для кожної функції визначаються повні річні витрати й кількість робочих годин;

3) Для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат;

4) Після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки. Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на сучасних комп’ютерах з доступом до Інтернету;

– повинен забезпечувати швидку обробку реальних геопросторових даних та побудову моделей для рекомендації;

– забезпечувати якісні рекомендації геопозицій;

– забезпечувати зручність, простоту та якість інтерпретації результатів;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який визначає рекомендації щодо геопозиціонування громадських закладів, вибираючи

місця з заданого списку, та використовуючи моделі машинного навчання. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір типу моделі машинного навчання;

F_3 – вибір способу візуалізації.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування R.

Функція F_2:

а) Лінійна;

б) Нелінійна.

Функція F_3:

а) Plotly;

б) Kepler.gl.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи на рисунку 4.1. На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 4.1).

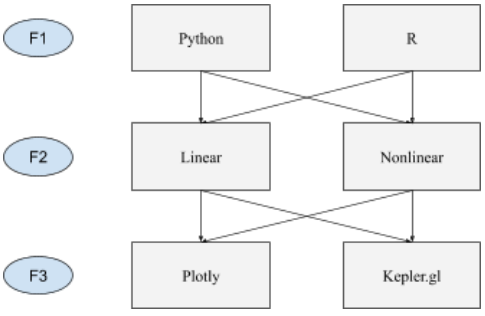


Рисунок 4.1 Теплова карта ознаки конкурентоспроможності та вхідного потоку для інтерпритації результату рекомендації

Рисунок 4.1 Теплова карта ознаки конкурентоспроможності та вхідного потоку для інтерпритації результату рекомендації

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Низька вартість, багато загальнодоступних модулів, відносна легкість проведення експериментів	Низька швидкодія
	Б	Популярність в наукових колах, наявність загальнодоступних бібліотек	Низький рівень підтримки
F2	А	Легкість у тренуванні та інтерпретації	Вимоги до даних, точність.
	Б	Легкість інтерпретації, можливість рішення комплексних проблем, робота з різними даними	Проблема перенавчання
F3	*А*	Загальнодоступність, універсальність	Складність імплементації

	Б	Зручний інтерфейс, та легкість побудови графіків, інтерактивність	Обчислювальна складність

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1*:*

Оскільки для даного продукту критична надійність і підтримуваність то в подальшому будемо розглядати тільки варіант а).

Функція F2:

Оскільки для даного продукту важливою є точність та універсальність, відкидаємо варіант а).

Функція F3:

Оскільки кожен варіант має свої переваги, в подальшому будемо розглядати обидва.

Таким чином, будемо розглядати такий варіант реалізації ПП:

1) F1а – F2б – F3а;

2) F1а – F2б – F3б.

3.2 Обґрунтування системи параметрів програмного продукту

Визначивши основні вимоги щодо функцій програмного продукту, визначають основні параметри виробу, що використовуватимуться для розрахунку коефіцієнта технічного рівня.

Введемо наступні параметри для характеристики програмного продукту:

1) X1 – об’єм пам’яті для збереження даних;

2) X2 – час обробки даних;

3) X3 – точність розв’язку;

4) X4 – потенційний об’єм програмного коду.

X1 – відображає об’єм пам’яті на обчислювальній машині, необхідний для збереження даних та програмного коду.

X2 – відображає час, який витрачається на виконання обробки і агрегації даних, побудови моделей та формування рекомендації.

X3 – відображає необхідну точність роботи алгоритма.

X4 – показує розмір програмного коду, який необхідно створити.

За заданими основними функціями, що повинен реалізувати продукт, бажаних характеристик та вимог до продукту, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня. Будемо розглядати три типи варіантів значення параметрів. Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Об’єм пам’яті для збереження даних	X1	Гб	9	3	1

Час обробки даних	X2	години	5	2	0.5
Точність розв’язку	X3	доля одиниці	10E-3	10E-4	10E-5
Потенційний об’єм програмного коду	X4	рядків коду	2000	1300	800

За даними таблиці 4.2 будуються графічні характеристики параметрів представлені на рисунках 4.2 – 4.5.

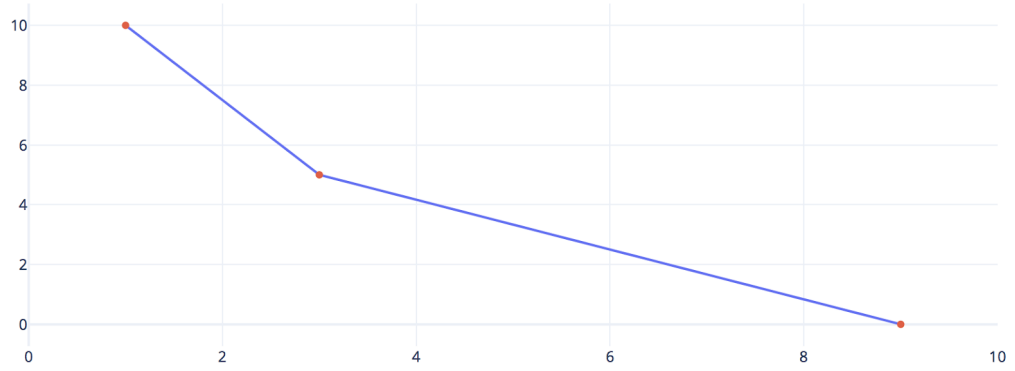


Рисунок 4.2 – X1, об’єм пам’яті для збереження даних

Рисунок 4.3 – X2, час обробки даних

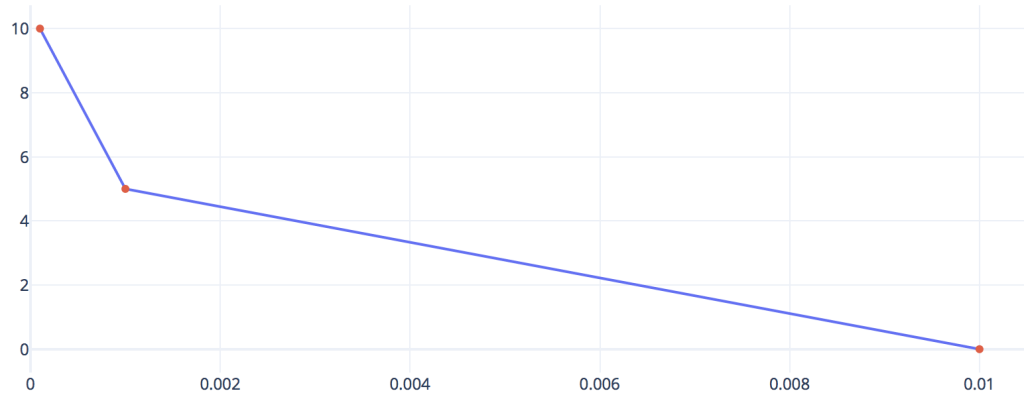


Рисунок 4.4 – X3, точність розв’язку

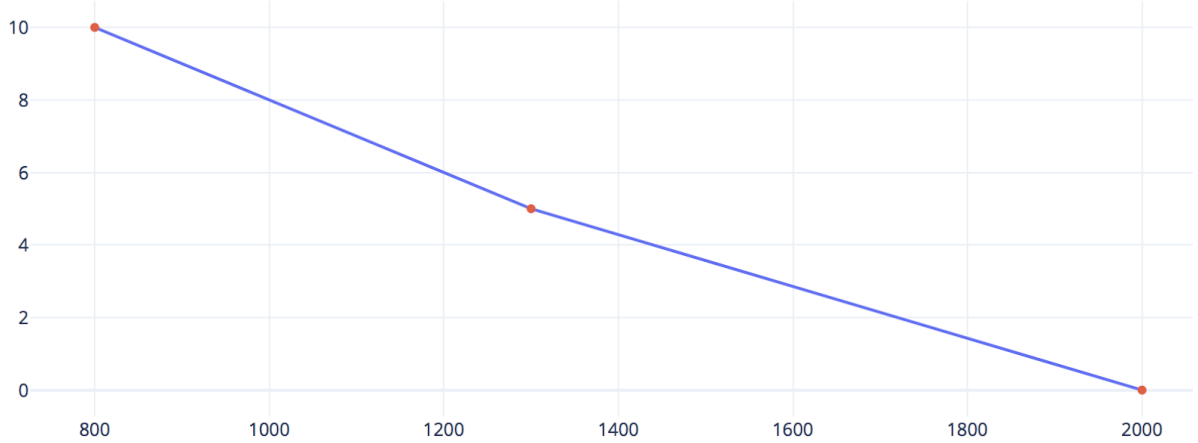


Рисунок 4.5 – X4, Потенційний об’єм програмного коду

4.3 Аналіз експертного оцінювання параметрів

Після детального обговорення, дослідження та аналізу, кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при рекомендації найкращого географічного розташування громадського закладу.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості. Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Познач. парам.	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Об’єм пам’яті для збереження даних	Гб	1	2	1	1	1	2	1	9	-8,5	72,25
X2	Час обробки даних	години	2	1	2	2	3	1	2	13	-4,5	20,25
X3	Точність розв’язку	доля одиниці	3	4	3	4	2	4	3	23	5,5	30,25
X4	Потенційний об’єм програмного коду	кількість рядків коду	4	3	4	3	4	3	4	25	7,5	56.25

	Разом		10	10	10	10	10	10	10	70	0	179

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_j^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 ,$$

де N – кількість експертів, n – кількість параметрів.

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{n} \Delta_i^{\,2} = 179$$

Обчислимо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2} , \; W = \frac{12 \cdot 179}{7^2 (4^3 - 4)} = 0,73 > W_k = 0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, який дорівнює 0,67.

Скориставшись результатами ранжування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	>	<	<	<	>	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	<	<	<	>	<	<	<	0,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	<	>	<	>	<	>	<	=	1

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається так:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \left|\left|a_{ij}\right|\right|$. Для кожного параметра зробимо розрахунок вагомості $K_{\text{ві}}$ за формулою (4.1):

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{ де } b_i = \sum_{i=1}^{N} a_{ij} \qquad (4.1)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються так:

$$K_{\text{ві}} = \frac{b'_i}{\sum_{i=1}^{n} b'_i},$$

$$b'_i = \sum_{i=1}^{N} a_{ij} b_j$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	$b^1{}_i$	$K^1{}_{\text{ві}}$	$b_i{}^2$	$K^2{}_{\text{ві}}$
X1	1,0	0,5	0,5	0,5	2,5	0,156	9.25	0,155	34,38	0,156
X2	1,5	1,0	0,5	0,5	3,5	0,218	12.25	0,206	45,12	0,205
X3	1,5	1,5	1,0	1,0	5	0,313	19	0,319	70,25	0,319
X4	1,5	1,5	1,0	1,0	5	0,313	19	0,319	70,25	0,319
Всього:					16	1	59.5	1	220	1

4.4 Аналіз рівня якості варіантів реалізації функцій

Визначено рівень якості кожного варіанту виконання основних функцій. Абсолютні значення параметра X1 (об’єм пам’яті для збереження даних), X3 (точність розв’язку), X4 (потенційний об’єм програмного коду) відповідають вимогам до даного ПП. Абсолютні значення параметра X2 (час обробки даних) обрані з наступних міркувань: для варіанта а) X2 = 1 година, тому що такий варіант підходить для клієнтів, що вимагають швидкого рішення; для варіанта б) X4 = 5 годин – клієнти, для яких час не є критичним.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так:

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j} \ ,$$

де n – кількість параметрів;

$K_{вi}$ – коефіцієнт вагомості i-го параметра;

B_i – оцінка i-го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	7	2	0,156	0,312
F2	Б	10Е-4	5	0,319	1,595
F3	А	1	9	0,205	1,845
F4	Б	5	1	0,205	0,205

За даними з таблиці 4.6 за формулою (4.2) визначаємо рівень якості кожного з варіантів:

$$K_K = K_{TY}[F_{ik}] + K_{TY}[F_{2k}] + ... + K_{TY}[F_{zk}], \quad (4.2)$$

$$K_{K1} = 0,312 + 1,595 + 1,845 = 3,752,$$

$$K_{K2} = 0,312 + 1,595 + 0,205 = 2,112$$

Отже, найкращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1) Розробка проекту програмного продукту;
2) Розробка графічної оболонки для демонстрації;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється так:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.3)$$

У формулі (4.3) T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру з ступенем 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: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний одиниці: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.6. Тоді загальна трудомісткість програмування першого завдання дорівнює: $Т_1 = 90 \cdot 1.7 \cdot 0.7 = 107,1$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань. Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р = 28$ людино-днів, $К_П = 0.7$, $К_{СК} = 1$, $К_{СТ} = 0,8$:

$$Т_2 = 28 \cdot 0,7 \cdot 0,8 = 15,68 \text{ людино-днів}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (107,1 + 15,68 + 4,8 + 15,68) \cdot 8 = 1146,08 \text{ людино-годин}$$
$$Т_{II} = (107,1 + 15,68 + 6,91 + 15,68) \cdot 8 = 1162,96 \text{ людино-годин}$$

Найбільш високу трудомісткість має варіант II.

В розробці бере участь два програмісти з окладом 5000 грн. і один фахівець з містобудування з окладом 11000 грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{Т_m \cdot t} \text{ грн.} \qquad (4.4)$$

У формулі (4.4) М – місячний оклад працівників; t – кількість робочих днів тиждень; T_m – кількість робочих годин в день.

$$C_{ч} = (5000 + 5000 + 11000)/(3 \cdot 21 \cdot 8) = 41,67 \text{ грн}$$

Тоді, розраховуємо заробітну плату за формулою

$$C_{зп} = C_{ч} \cdot T_{i} \cdot K_{д} \qquad (4.5)$$

У формулі 4.5 $C_{ч}$ – величина погодинної оплати праці програміста; T_{i} – трудомісткість відповідного завдання; $K_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників становить:

$$C_{зп1} = 41,67 \cdot 1146,08 \cdot 1,2 = 57308,58 \text{ грн.},$$

$$C_{зп2} = 41,67 \cdot 1162,96 \cdot 1,2 = 58152,65 \text{ грн.}$$

Відрахування на соціальний внесок становить 36,77%:

$$C_{від1} = C_{зп1} \cdot 0,22 = 57308,58 \cdot 0,22 = 21072,36 \text{ грн.},$$

$$C_{від2} = C_{зп2} \cdot 0,22 = 58152,65 \cdot 0,22 = 21382,73 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. ($C_{М}$)

Так як одна ЕОМ обслуговує одного працівника з окладом 11000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{Г} = 12 \cdot М \cdot K_{З} = 12 \cdot 11000 \cdot 0,2 = 12000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_{Г} \cdot (1 + K_{З}) = 12000 \cdot (1 + 0.2) = 14400 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД}= С_{ЗП} \cdot 0.3677 = 14400 \cdot 0,3677 = 5294.88 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 25000 = 7187,5 \text{ грн.},$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$ – річна норма амортизації; $Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 25000 \cdot 0.05 = 1437,5 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0,9 = 1706,4 \text{ год},$$

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t – кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,22 \cdot 0,78 \cdot 2,7515 = 805,69 \text{ грн.},$$

де N_C – середньо-споживча потужність приладу; $K_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо так:

$$С_Н = Ц_{ПР} \cdot 0.67 = 25000 \cdot 0,67 = 16750 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н,$$

$$С_{ЕКС} = 14400 + 5294.88 + 7187,5 + 1437,5 + 805,69 + 16750 = 45875,57 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 45875,570 / 1706,4 = 26,884 \text{ грн/год}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу:

$$С_М = С_{М\text{-}Г} \cdot Т,$$

$$С_{М1} = 26,884 \cdot 1146,08 = 30811,21 \text{ грн.},$$

$$С_{М2} = 26,884 \cdot 1162,96 = 31265,02$$

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

$$С_{н1} = 0,67 \cdot 57308,58 = 38396,75 \text{ грн.},$$

$$С_{н2} = 0,67 \cdot 58152,65 = 38962,27 \text{ грн.}$$

Отже, вартість розробки ПП становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н},$$

$$С_{ПП} = 57308,58 + 21072,36 + 30811,21 + 38396,75 = 147588,9 \text{ грн.},$$

$$С_{ПП} = 58152,65 + 21382,73 + 31265,02 + 38962,27 = 149762,67 \text{ грн.}$$

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{тер_j} = \frac{К_{К_j}}{С_{Ф_j}},$$

$$К_{ТЕР1} = 3,752 / 147588,9 = 2,5 \cdot 10^{-5},$$

$$К_{ТЕР2} = 2,112 / 149762,67 = 1,4 \cdot 10^{-5}$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 2,5 \cdot 10^{-5}$.

4.7 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних

оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР1} = 2,5 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– Мова програмування – Python;

– Тип моделі машинного навчання – нелінійна;

– Засіб візуалізації – Plotly.

Даний варіант виконання програмного комплексу дає користувачу можливість отримувати хороші рекомендації щодо геопозиціонування стратегії повернення до середнього значення з достатньою точністю, уможливлює велику кількість експериментів під час розробки та їх ефективну візуалізацію.

ВИСНОВКИ

Дана дипломна робота присвячена методам аналізу геоданих, побудови моделей машинного навчання з метою надання рекомендацій для розміщення громадських закладів у місті Києві.

У роботі було розглянуто основні методи роботи з точковими геоданими та способи виокремлення з них корисної інформації. Зокрема було розглянуто дані з соціальної мережі Foursquare, дані про вуличні мережі OSMnx та дані про громадський транспорт у місті Києві з EasyWay API. Розглянуто основні підходи до агрегації таких даних.

У роботі було розглянуто інноваційний підхід щодо використання геоданих не тільки для оглядового аналізу, а й для моделювання складних рекомендаційних систем. Було розглянуто та визначено структуру рекомендаційної системи, основні алгоритми та метрики. Проведено експерименти з конкретною групою громадських закладів – кав'ярнями. Знайдено основні закономірності. Побудована рекомендаційна система та підтверджена її ефективність. Проведена візуалізація наявних даних та інтерпретація отриманих результатів

Було розглянуто основні моделі лінійної та нелінійної регресії. Визначено умови та проведено експерименти з метою порівняти результативність різних моделей та перевірити їх придатність для задачі рекомендації геопозиції.

У подальших розробках можливе використання спеціалізованих моделей призначених для ранжування та нейронних мереж. Також можливе розширення переліку характерних ознак громадських закладів.

СПИСОК ЛІТЕРАТУРИ

1. Як Big Data від Київстар допомагає Правекс Банку знаходити клієнтів. URL: https://hub.kyivstar.ua/ua/kak-big-data-ot-kievstar-pomogaet-praveks-banku-nahodit-klientov (дата звернення: 01.06.2019).

2. Dingqi Y. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. *The Web Conference (WWW'19)*. 2019. URL: https://exascale.info/assets/pdf/yang2019www.pdf (Last accessed: 01.06.2019).

3. Boeing G. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*. 2017. №65. С. 126–139.

4. Boeing G. Understanding Cities through Networks and Flows. *Berkeley Planning Journal*. 2017. №28. С. 118–123.

5. Метрики качества ранжирования. *HABR*. URL: https://m.habr.com/ru/company/econtenta/blog/303458/ (дата звернення: 01.06.2019).

6. Jensen P. Network-based predictions of retail store commercial categories and optimal location. *Physical Review*. 2006. №74.

7. Karamshuk D. Geo-Spotting: Mining Online Location-based Services for Optimal Retail Store Placement. *ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013. №19. С. 793–801.

8. Krittika D. The Role of Urban Mobility in Retail Business Survival. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2018. №2. С. 1–22.

9. Mehaffy M. Urban nuclei and the geometry of streets: The ’emergent neighborhoods’ model. *Urban Design International*. 2010. №15. С. 22–46.

10. Shannon C. A. Mathematical Theory of Communication. *The Bell System Technical Journal*. 1948. №27. С. 379–423, 623–656.

11. Цілі сталого розвитку 2016-2030. URL: http://www.un.org.ua/ua/tsili-rozvytku-tysiacholittia/tsili-staloho-rozvytku (дата звернення: 01.06.2019).

12. Boeing G. The Morphology and Circuity of Walkable and Drivable Street Networks. *The Mathematics of Urban Morphology*. 2019. DOI: 10.31235/osf.io/edj2s

13. Bishop C. Pattern Recognition and Machine Learning. Berlin: Springer-Verlag, 2006. 738 p.

14. Regression and Classification | Supervised Machine Learning. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/ (дата звернення: 01.06.2019).

15. MIPT Machine Learning online specialization. *Coursera online learning platform*. URL: https://www.coursera.org/specializations/machine-learning-data-analysis. (дата звернення: 01.06.2019).

16. Tibshirani R. Improvements on Cross-Validation: The 632+ Bootstrap Method. *Journal of the American Statistical Association*. 1997. №438. С. 548–560.

17. Tianqi C. XGBoost: A Scalable Tree Boosting System. *KDD '16 Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. №22. С. 785–794.

18. Снитюк В. Є. Прогнозирование. Модели, методы, алгоритмы: учебное пособие. Київ: Маклаут, 2008. С. 364.

19. Andrew Ng Neural Networks Specialization. *Coursera learning platform*. URL: https://www.coursera.org/learn/deep-neural-network/home/welcome (дата звернення: 01.06.2019).

ДОДАТОК А ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Методи аналізу геоданих для надання рекомендацій щодо розміщення громадських закладів у Києві

Виконав:
студент групи КА-51
Трохимович М.А.

Керівник:
к.ф.-м.н., доцент
Каніовська І. Ю.

Актуальність дослідження

- Стрімко розвивається напрям “розумне місто” (англ. smart city). Впровадження єдиних електронних квитків на громадський транспорт, соціальні мережі, відцифрування інформації сприяють накопиченню потужної бази знань.
- Аналіз геоданих та побудова математичних моделей, потенційно може покращити доступність товарів та послуг для людей, та збільшити вигоду для бізнесу.

- **Об'єкт дослідження**
  - Data mining, алгоритми машинного навчання, побудова складних рекомендаційних моделей
- **Предмет дослідження**
  - Точкові геодані та математичні моделі побудови рекомендацій
- **Мета дослідження**
  - Дослідити можливість надання рекомендацій, щодо розташування громадських закладів у місті Києві. Побудувати модель для рекомендації геопозиціонування конкретної категорії громадських закладів

Постановка задачі

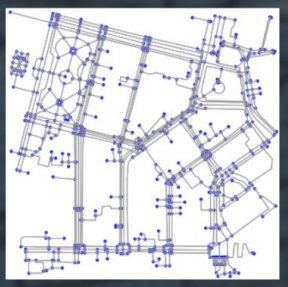
- Визначити показник успішності громадського закладу
- Проаналізувати дані з різних джерел, спроектувати та видобути необхідні атрибути конкретних локацій
- Побудувати модель для відновлення функціональної залежності успішності громадських закладів від здобутих географічних ознак
- За допомогою моделі знайти цільові значення для нових локацій. Проранжувавши ці значення, дати рекомендацію щодо найкращих місць для розміщення конкретного типу громадських закладів.
- Оцінити якість роботи рекомендаційної системи

Дані та здобуті ознаки

➢ Foursquare

- ❑ Показник успішності – кількість чекінів
- ❑ Density – кількість закладів навколо заданого місця
- ❑ Place entropy - просторова гетерогенність області
- ❑ Competitiveness - конкурентоспроможність області
- ❑ Popularity - загальна кількість чекінів у досліджуваній області
- ❑ Transition Density - мобільність між місцями всередині області
- ❑ Incoming Flow - зовнішній потік людей до заданої області.

➢ EasyWay API

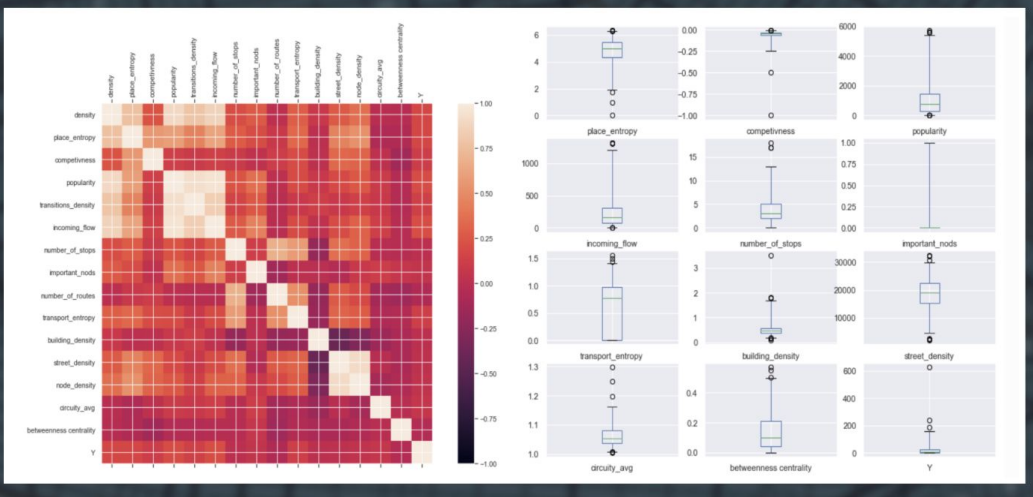


- ❑ stops density – кількість зупинок громадського транспорту
- ❑ important nodes – наявність важливих транспортних вузлів
- ❑ transport density - кількість маршрутів громадського транспорту
- ❑ transport entropy - різноманітність видів громадського транспорту

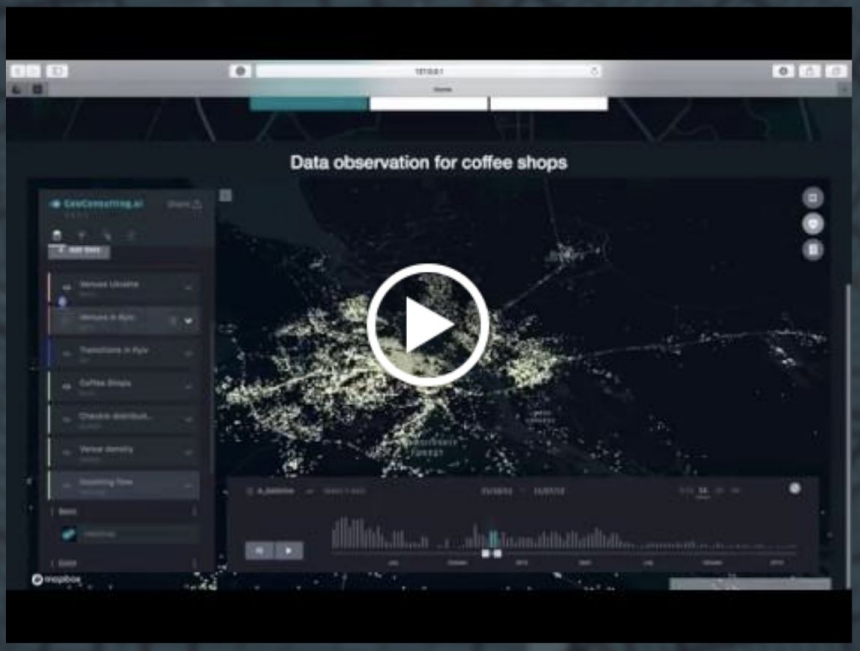
➢ OSMnx API

- ❑ street density – щільність вулиць
- ❑ intersection density – щільність перехресть
- ❑ average circuity - середня заокругленість вулиць
- ❑ betweenness centrality - важливість даної вершини у мережі

Огляд здобутих ознак



Демонстрація в графічному інтерфейсі

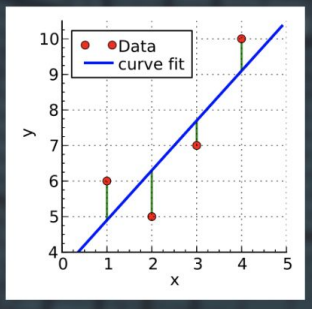


Відновлення функціональної залежності

❑ Лінійні моделі

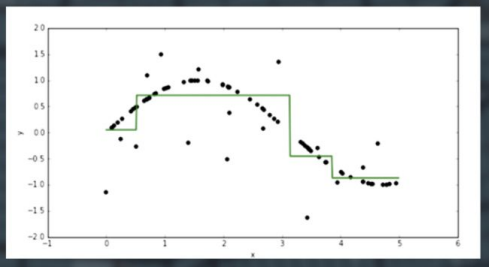
Лінійна регресія:
- Стохастичний градієнтний спуск
- Ridge (L2 регуляризація)
- Lasso (L1 регуляризація)
- ElasticNet (L2+L1 регуляризація)

$$Q(w,X) = \frac{1}{l}||Xw - y||^2 + \lambda||w||_{l_1} + \lambda||w||^2 \rightarrow min_{w \in R^d}$$



★ Нелінійні моделі

Decision tree based algorithms:
- Decision Tree
- Random Forest
- XGBoost Regression



Decision Tree and Random Forest

- Навчання моделі:
  - Існує вершина m у якій Xm об'єктів з навчальної вибірки.
  - Відбувається її поділ на $X_l = \{x \in X_m | \left[x^j \leq t\right]\}, X_r = \{x \in X_m | \left[x^j > t\right]\}$

    При цьому параметри $\left[x^j \leq t\right]$ обираються з таких міркувань:

$$Q\left(X_m, j, t\right) = \frac{|X_l|}{|X_m|} H\left(X_l\right) + \frac{|X_r|}{|X_m|} H\left(X_r\right) \to \min_{j,t}$$

$$H(X) = \frac{1}{|X|} \sum_{i \in X} \left(y_i - \overline{y}(X)\right)^2, \text{ ae } \overline{y} = \frac{1}{|X|} \sum_{i \in X} y_i$$

- Можливі критерії зупинки:
  - У вершині тільки один об'єкт навчальної вибірки
  - Глибина дерева досягла певного значення
  - Кінцева відповідь моделі в листі m: $a_m = \frac{1}{|X_m|} \sum_{i \in X_m} y_i$
- Random Forest це сукупність дерев, відповіді яких комбінуються.

Метрики

- Precision at K

$$p@K = \frac{\sum_{k=1}^{K} r^{true}\left(\pi^{-1}(k)\right)}{K} = \frac{\text{number of relevant}}{K}$$

- Average precision at K

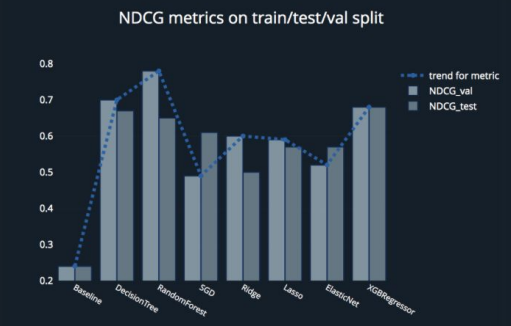
$$ap@K = \frac{1}{K} \sum_{k=1}^{K} r^{true}\left(\pi^{-1}(k)\right) \cdot p@k$$

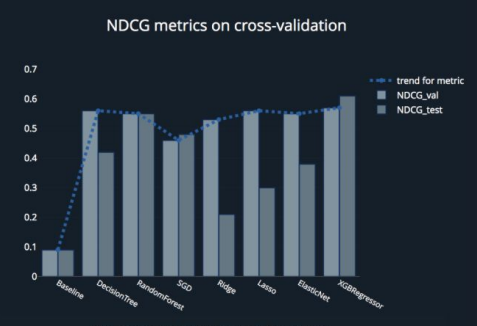
- Normalized discounted cumulative gain (nDCG)

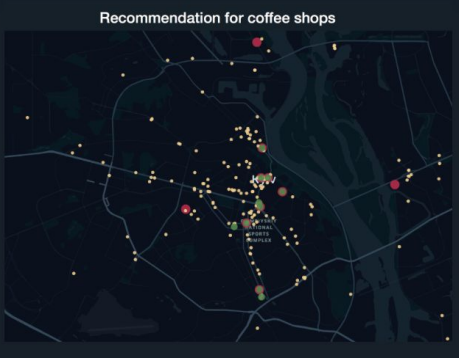
$$DCG@K = \sum_{k=1}^{K} \frac{2^{r^{true}\left(\pi^{-1}(k)\right)} - 1}{\log_2(k+1)}$$

$$nDCG@K = \frac{DCG@K}{IDCG@K}$$

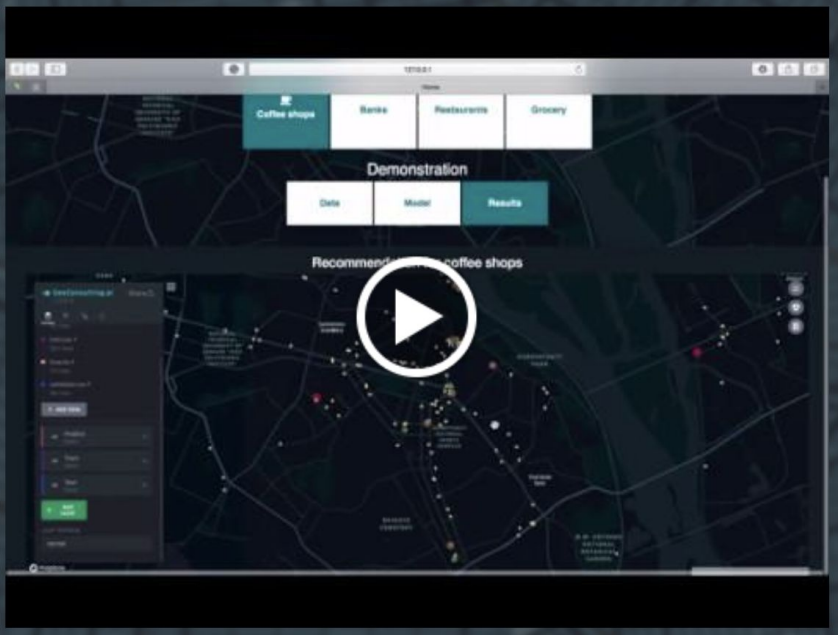
Результати експериментів з кав'ярнями







Демонстрація роботи моделей



Практичне застосування

- **Рекомендація щодо розміщення**
  - Визначення найкращого географічного положення нового громадського закладу
- **Оцінка ефективності роботи бізнесу**
  - Знаючи характеристики конкретної локації можна визначити теоретичний показник її успішності. Порівнявши його з емпіричним показником можна оцінити ефективність організації роботи громадського закладу

Висновки

- Проаналізували дані з різних джерел, спроектували та видобули необхідні атрибути конкретних локацій, знайшли закономірності в даних.
- Побудували моделі для відновлення функціональної залежності успішності громадських закладів від здобутих географічних ознак.
- За допомогою моделі знайшли цільові значення для нових локацій. Проранжувавши ці значення, дали рекомендацію щодо найкращих розташувань кав'ярень у місті Києві.
- Спроектували рекомендаційну систему та програму для візуалізації, демонстрації та інтерпретації результатів дослідження.

Подальші дослідження

- Використання нейронних мереж для відновлення функціональної залежності.
- Використання більшої кількості джерел даних.
- Порівняння ефективності роботи у різних містах та на різних типах закладів.

Дякую за увагу!

ДОДАТОК Б КОД ПРОГРАМНОГО ПРОДУКТУ

```
#/usr/bin/env python
# coding: utf-8
# # Extracting big data
# ## Check ins data
# In[1]:
import pandas as pd
from tqdm import tqdm
PATH = 'raw_Checkins_anonymized.txt'
chunksize = 1000000
traintypes = {
'user':'str',
'ven':'str',
'time':'str',
'offset':'float'}
cols = list(traintypes.keys())
df_list = [] # list to hold the batch dataframe
for df_chunk in tqdm(pd.read_csv(PATH, names=cols, dtype=traintypes, chunksize=chunksize, sep='\t')):
    # Can process each chunk of dataframe here
    # clean_data(), feature_engineer(),fit()
    # Alternatively, append the chunk to list and merge all
    df_list.append(df_chunk)
# Merge all dataframes into one dataframe
X = pd.concat(df_list)
# Delete the dataframe list to release memory
del df_list
del df_chunk
# In[2]:
X.head()
# ## Venues data
# In[3]:
import pandas as pd
from tqdm import tqdm
PATH = 'raw_POIs.txt'
chunksize = 500000
traintypes = {
'ven':'str',
'lat':'float',
'lon':'float',
'cat':'str',
'country':'str'}
cols = list(traintypes.keys())
df_list = [] # list to hold the batch dataframe
```

```
for df_chunk in tqdm(pd.read_csv(PATH, names=cols, dtype=traintypes, chunksize=chunksize, sep='\t')):
    # Can process each chunk of dataframe here
    # clean_data(), feature_engineer(),fit()
    # Alternatively, append the chunk to list and merge all
    df_list.append(df_chunk)
# Merge all dataframes into one dataframe
Y = pd.concat(df_list)
# Delete the dataframe list to release memory
del df_list
del df_chunk
# ## Observing global venues data
# In[5]:
Y_UA = Y[Y.country=='UA']
# In[12]:
Y_UA.to_csv('venues_in_Ukraine.csv', index=False)
# In[6]:
Y_kiev = Y_UA[Y_UA.lat>50.31][Y_UA.lat<50.57][Y_UA.lon>30.24][Y_UA.lon<30.81]
# In[13]:
Y_kiev.to_csv('venues_in_Kyiv.csv', index=False)
# In[1]:
import pandas as pd
Y_kiev = pd.read_csv('venues_in_Kyiv.csv')
# ## Observing global check ins data
# In[7]:
a = Y_kiev.ven.unique()
# In[8]:
X_kiev = X[X.ven.isin(a)]
# ## Observing OSMNx and Easyway API
# In[2]:
import plotly.plotly as py
import plotly.graph_objs as go
import requests
latt = 50.439546
lng = 30.517015
r = requests.get('https://api.easyway.info/?login=trokhymovych.mykola&password=mw6GWvwhc29Kwmz&city=kyiv&function=stops.GetStopsNearPointWithRoutes&lat='+str(latt)+'&lng='+str(lng)+'&r=300')
data = r.json()
lat = []
lon = []
for i in data['stop']:
    lat.append(i['lat'])
    lon.append(i['lng'])
```

```
mapbox_access_token =
"pk.eyJ1IjoidmFsZW50eW4xOTk3IiwiYSI6ImNqNHlubm03cjFpc3EzM21nbW1rdGhvNmwifQ.xu
Q0fbWNXjOi3y03MYbqAQ"
data = [
    go.Scattermapbox(
        lat=lat,
        lon=lon,
        mode='markers',
        name = 'Public transport stops',
        marker=go.scattermapbox.Marker(
            size=9
        ),
    ),
    go.Scattermapbox(
        lat=[latt],
        lon=[lng],
        mode='markers',
        name = 'Cafe',
        marker=go.scattermapbox.Marker(
            size=13
        ),
    )
]
layout = go.Layout(
    autosize=True,
    hovermode='closest',
    mapbox=go.layout.Mapbox(
        accesstoken=mapbox_access_token,
        bearing=0,
        center=go.layout.mapbox.Center(
            lat=latt,
            lon=lng
        ),
        pitch=0,
        zoom=14.5
    ),
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='Multiple Mapbox')
# In[19]:
import osmnx as ox
get_ipython().run_line_magic('matplotlib', 'inline')
ox.config(log_console=True, use_cache=True)
ox.__version__
# create a network around some (lat, lon) point and plot it
location_point = (latt, lng)
```

```
G = ox.graph_from_point(location_point, distance=500, simplify=True, network_type='walk')
fig, ax = ox.plot_graph(G, node_color='b', node_zorder=1)
# # Extracting transitions data from check ins data
# In[ ]:
X_kiev.time = pd.to_datetime(X_kiev.time)
# In[ ]:
# from tqdm import tqdm_notebook as tqdm
# data3 = X_kiev
# users = data3.user.unique()
# transitions = pd.DataFrame(columns = ['A_id', 'B_id', 'A_datetime', 'B_datetime'])
# for i in tqdm(range(len(users))):
#     user_1 = data3[data3.user==users[i]]
#     user_1 = user_1.drop_duplicates(keep='first')
#     for i in range(len(user_1.time)-1):
#         if list(user_1.time)[i+1] - list(user_1.time)[i]<pd.Timedelta('3 hours'):
#             t = pd.DataFrame({'A_id':list(user_1[user_1.time==list(user_1.time)[i]].ven)[:1],
#                               'B_id':list(user_1[user_1.time==list(user_1.time)[i+1]].ven)[:1],
#                               'A_datetime':list(user_1.time)[i],
#                               'B_datetime':list(user_1.time)[i+1]})
#             transitions = pd.concat([transitions,t], axis=0)
# transitions
# In[ ]:
# transitions.to_csv('transitions_all.csv', index= False)
# In[25]:
transitions = pd.read_csv('transitions_all.csv')
# In[28]:
transitions = transitions.merge(Y_kiev, left_on='A_id', right_on='ven').merge(Y_kiev, left_on='B_id',
right_on='ven')[['A_id', 'B_id', 'A_datetime', 'B_datetime', 'lat_x', 'lon_x', 'lat_y', 'lon_y']]
# In[30]:
transitions.to_csv('transitions_all_Kyiv.csv', index= False)
# In[36]:
Y_kiev.head()
coffee_id = list(Y_kiev[Y_kiev.cat=='Coffee Shop'].ven.unique())
# In[39]:
from geopy import distance
A = list(transitions.A_id)
B = list(transitions.B_id)
dist = []
for a,b in tqdm(zip(A,B)):
    coord_a = (list(Y_kiev[Y_kiev.ven==a].lon)[0], list(Y_kiev[Y_kiev.ven==a].lat)[0])
    coord_b = (list(Y_kiev[Y_kiev.ven==b].lon)[0], list(Y_kiev[Y_kiev.ven==b].lat)[0])
    dist.append(distance.distance(coord_a,coord_b).m)
transitions['distance'] = dist
# In[3]:
# transitions.to_csv('transitions_all_Kyiv.csv', index= False)
transitions = pd.read_csv('transitions_all_Kyiv.csv')
```

```
# In[4]:
coffee_id = list(Y_kiev[Y_kiev.cat=='Coffee Shop'].ven.unique())
bank_id = list(Y_kiev[Y_kiev.cat=='Bank'].ven.unique())
rest_id = list(Y_kiev[Y_kiev.cat=='Restaurant'].ven.unique())
grocery_id = list(Y_kiev[Y_kiev.cat=='Grocery Store'].ven.unique())
# In[57]:
import numpy as np
dist = list(transitions[transitions.B_id.isin(grocery_id)]['distance'])
# dist = list(transitions['distance'])
dist = [i for i in dist if i > 0]
dist = np.sort(dist)
# In[58]:
import numpy as np
import statsmodels.api as sm # recommended import according to the docs
import matplotlib.pyplot as plt
sample = dist
ecdf = sm.distributions.ECDF(sample)
fig = plt.figure(figsize=(12,8))
ax = fig.gca()
ax.set_xticks(np.arange(0, 10000, 500))
ax.set_yticks(np.arange(0, 1., 0.1))
x = np.linspace(min(sample), 10000,num=1000)
y = ecdf(x)
plt.step(x, y, label='ECDF')
plt.plot(x,np.ones(len(x))*0.5,linestyle='--',color='r', label='50th percentile (Q2)')
# plt.plot(x,np.ones(len(x))*0.8,linestyle='--',color='r',)
plt.title('Емпірична функція розподілу відстаней')
plt.xlim(0, 6000)
plt.ylim(0, 1)
plt.legend()
plt.xlabel("distance")
plt.ylabel("ECDF");
plt.grid()
plt.show()
# # Building features for future regression analysis
# 1) щільність
# 2) ентропія розташування,
# 3) конкурентоспроможність,
# 4) популярність,
# 5) щільність переміщень,
# 6) величина вхідного потоку,
# 7) кількість зупинок,
# 8) наявність поблизу важливих транспортних вузлів,
# 9) кількість маршрутів,
# 10) ентропія транспорту,
# 11) щільність вулиць,
```

```
# 12) щільність перехресть,
# 13) середня заокругленість вулиць,
# 14) центральність цільової точки,
# 15) густину забудованості
# In[12]:
from math import pi
import networkx as nx
import osmnx as ox
import requests
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt
ox.config(use_cache=True, log_console=True)
def get_OSMnx(x,y,rad):

    p = (x,y)

    G = ox.graph_from_point(p, distance=rad, network_type='walk')
    G_proj = ox.project_graph(G)
    nodes_proj = ox.graph_to_gdfs(G_proj, edges=False)
    are = nodes_proj.unary_union.convex_hull.area

    try:
        gdf = ox.buildings_from_point(p, distance = rad)
        gdf_proj = ox.project_gdf(gdf)
        areas = gdf_proj.area
        building_density = sum(areas)/are
    except:
        building_density = np.nan



    # graph analysis
    stats = ox.basic_stats(G, area=500*500)
    temp = ox.extended_stats(G, ecc=True, bc=True, cc=True)
    for key, value in stats.items():
        temp[key] = value

    street_density = temp['street_density_km']#Total street length divided by area in square kilometers
    node_density = temp['intersection_density_km']#n divided by area in square kilometers
      circuity_avg = temp['circuity_avg'] #Total edge length divided by sum of great circle distances between the nodes incident to each edge Proportion of edges that have a single incident node (i.e., the edge links nodes u and v, and u = v)

#     # point analysys
    origin_node = ox.get_nearest_node(G, p)
```

```
    b_c = temp['betweenness_centrality'][origin_node]    #For each node, the fraction of all shortest paths that pass through the node

    return building_density, street_density, node_density, circuity_avg, b_c
# In[13]:
from geopy.distance import vincenty
from geopy import distance
import requests
from math import log2
import warnings
warnings.filterwarnings("ignore")
from tqdm import tqdm_notebook as tqdm
# Important transport nodes
r = requests.get('https://api.easyway.info/?login=trokhymovych.mykola&password=mw6GWvwhc29Kwmz&city=kyiv&function=cities.GetPlacesList')
important = r.json()['place']
transitions = transitions[transitions.distance>0]
my_cat = 'Restaurant' ################################################
def get_transport_data(lat,lng):

    r = requests.get('https://api.easyway.info/?login=trokhymovych.mykola&password=mw6GWvwhc29Kwmz&city=kyiv&function=stops.GetStopsNearPointWithRoutes&lat='+str(lat)+'&lng='+str(lng)+'&r=200')
    data = r.json()

    # Number of stops
    number_of_stops = len(data['stop'])

    # preparation
    temp = {}
    for t in ['bus','train','trol','tram','metro','boat']:
        temp[t] = set()
    for stop in data['stop']:
        for route in stop['routes']['route']:
            type_ = route['@attributes']['type']
            number_ = route['title']
            temp[type_].add(number_)

    # Number of routes
    number_of_routes = 0
    for r in temp.keys():
        number_of_routes+=len(temp[r])

    #Transport Entropy
```

```
    entropy = 0
    for r in temp.keys():
        if len(temp[r])!=0:
            entropy+=(len(temp[r])/number_of_routes)*np.log2(len(temp[r])/number_of_routes)
    entropy = -entropy

    #Number of important transport nods
    important_nods = 0
    for el in important:
        p1 = (el['lat'], el['lng'])
        p2 = (lat,lng)
        if (vincenty(p1, p2).meters < 400):
            important_nods+=1
    return number_of_stops, important_nods, number_of_routes, entropy
# getting places around
def get_places_around(data, lat,lon):
    # circle area
    radius = 200
    center_coordinates = (lat,lon)
    places_ids = []
    for l,ll in zip(data.lat,data.lon):
        dist = distance.distance(center_coordinates, (l, ll)).m
        if dist <= 200:
            places_ids.append(list(data[data.lon==ll][data.lat==l].ven)[0])
    return data[data.ven.isin(places_ids)]
# In[16]:
# mining_banks
from tqdm import tqdm_notebook as tqdm
import numpy as np

X_kiev = pd.read_csv('X_kiev.csv')


feature_names = ['density', 'place_entropy', 'competivness', 'popularity',
                 'transitions_density','incoming_flow', 'number_of_stops', 'important_nods',
                 'number_of_routes', 'transport_entropy', 'building_density',
                 'street_density', 'node_density', 'circuity_avg', 'betweenness centrality']
density1 = []
place_entropy1 = []
competivness1 = []
popularity1 = []
t_density1 = []
incoming_flow1 = []
number_of_stops1 = []
important_nods1 = []
number_of_routes1 = []
```

```
transport_entropy1 = []
building_density1 = []
street_density1 = []
node_density1 = []
circuity_avg1 = []
b_c1 = []
for index, venue in tqdm(Y_kiev[Y_kiev.ven.isin(rest_id)].iterrows()):
############################

    # getting dataframe of neibours
    temp = get_places_around(Y_kiev, venue['lat'], venue['lon'])

    # 1) density
    density = len(temp)
    density1.append(density)

    # 2) place entropy
    temp_cat = set(temp.cat)
    p_entropy = 0
    for c in temp_cat:
        dd = temp[temp.cat==c]
        p_entropy -= (len(dd) / density * log2(len(dd) / density))
    place_entropy1.append(p_entropy)

    # 3) Competivness
    competivness = -len(temp[temp.cat==my_cat])/density
    competivness1.append(competivness)

    # 4) Popularity
    aa = X_kiev[X_kiev.ven.isin(temp.ven)]
    popularity = len(aa)
    popularity1.append(popularity)

    # 5) Transition Density
    f1 = transitions[transitions.A_id.isin(temp.ven)]
    f1 = f1[f1.B_id.isin(temp.ven)]
    t_density = len(f1)
    t_density1.append(t_density)

    # 6) Incoming Flow
    f1 = transitions[~transitions.A_id.isin(temp.ven)]
    f1 = f1[f1.B_id.isin(temp.ven)]
    incoming_flow = len(f1)
    incoming_flow1.append(incoming_flow)

    # 7,8,9,10
```

```
    number_of_stops, important_nods, number_of_routes, entropy = get_transport_data(venue['lat'],
venue['lon'])
  number_of_stops1.append(number_of_stops)
  important_nods1.append(important_nods)
  number_of_routes1.append(number_of_routes)
  transport_entropy1.append(entropy)

  #11,12,13,14,15
  try:
      building_density, street_density, node_density, circuity_avg, b_c = get_OSMnx(venue['lat'],
venue['lon'], 200)
    building_density1.append(building_density)
    street_density1.append(street_density)
    node_density1.append(node_density)
    circuity_avg1.append(circuity_avg)
    b_c1.append(b_c)
  except:
    building_density1.append(np.nan)
    street_density1.append(np.nan)
    node_density1.append(np.nan)
    circuity_avg1.append(np.nan)
    b_c1.append(np.nan)
# In[17]:
data = pd.DataFrame({'density':density1,
                'place_entropy':place_entropy1,
                'competivness':competivness1,
                'popularity':popularity1,
                'transitions_density':t_density1,
                'incoming_flow':incoming_flow1,
                'number_of_stops':number_of_stops1,
                'important_nods':important_nods1,
                'number_of_routes':number_of_routes1,
                'transport_entropy':transport_entropy1,
                'building_density':building_density1,
                'street_density':street_density1,
                'node_density':node_density1,
                'circuity_avg':circuity_avg1,
                'betweenness centrality':b_c1})
# In[18]:
data['ven'] = list(Y_kiev[Y_kiev.ven.isin(rest_id)].ven)
# adding venue checkins
data['Y'] = list(X_kiev[X_kiev.ven.isin(rest_id)].groupby('ven').count().user)
data = data.merge(Y_kiev, on='ven')
data = data[list(data.columns)[:19]]
data.to_csv('restorant_dataset.csv', index = False)
# # Preprocesing is fineshed!!!!
```

```
import pandas as pd
X_kiev = pd.read_csv('X_kiev.csv')
data = pd.read_csv('coffee_dataset.csv')
feature_names = ['density', 'place_entropy', 'competivness', 'popularity',
            'transitions_density','incoming_flow', 'number_of_stops', 'important_nods',
            'number_of_routes', 'transport_entropy', #'building_density',
            'street_density', 'node_density', 'circuity_avg', 'betweenness centrality']
import pandas as pd
Y_UA = pd.read_csv('venues_ua.csv')
Y_kiev = Y_UA[Y_UA.lat>50.31][Y_UA.lat<50.57][Y_UA.lon>30.24][Y_UA.lon<30.81]
feature_names = ['density', 'place_entropy', 'competivness', 'popularity',
            'transitions_density','incoming_flow', 'number_of_stops', 'important_nods',
            'number_of_routes', 'transport_entropy', #'building_density',
            'street_density', 'node_density', 'circuity_avg', 'betweenness centrality']
X = data
X.fillna(X.mean(), inplace=True)
y = data.Y
other = list(X.columns)
for i in feature_names:
   other.remove(i)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=2)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=1./9., random_state=2)

X_train.merge(Y_kiev, on='ven').to_csv('train.csv', index=False)
X_val.merge(Y_kiev, on='ven').to_csv('validation.csv', index=False)
X_test.merge(Y_kiev, on='ven').to_csv('test.csv', index=False)

def standartize(df):
   df2 = df[other]
   df = df[feature_names]
   df = (df-df.mean())/df.std()
   return pd.concat([df,df2], axis=1)

def normalize(df):
   df2 = df[other]
   df = df[feature_names]
   df = (df-df.min())/(df.max()-df.min())
   return pd.concat([df,df2], axis=1)

def mixpreprocessing(df):
   df2 = df[other]
   df =  1./(1+np.exp(-standartize(df)[feature_names]))
   return pd.concat([df,df2], axis=1)

def precision_at_k(true, predict, k=10):
```

```
    tt = pd.DataFrame({'id':np.arange(len(predict)), 'true':true, 'pred':predict})
    true = tt.sort_values(by='true', ascending=False).id.values[:k]
    predict = tt.sort_values(by='pred', ascending=False).id.values[:k]
    return len(set(true).intersection(set(predict)))/float(k)

def AP(true, predict, k=10):

    tt = pd.DataFrame({'id':np.arange(len(predict)), 'true':true, 'pred':predict})
    true = tt.sort_values(by='true', ascending=False).id.values[:k]
    predict = tt.sort_values(by='pred', ascending=False).id.values[:k]
    precisions = np.zeros(k)
    tru_predicted = 0.

    for i in range(k):
        if predict[i] in true:
            tru_predicted += 1
            precisions[i] = tru_predicted/(i+1)
    return sum(precisions)/k

def NDCG_implicit(true, predict, k=10):

    tt = pd.DataFrame({'id':np.arange(len(predict)), 'true':true, 'pred':predict})
    true = tt.sort_values(by='true', ascending=False).id.values[:k]
    predict = tt.sort_values(by='pred', ascending=False).id.values[:k]

    idcg = 0.
    for i in range(k):
        idcg += 1./np.log2(i+2)

    dcg = 0.
    for i in range(k):
        if predict[i] in true:
            dcg += 1./np.log2(i+2)
    return dcg/idcg

from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
import xgboost as xgb

from copy import deepcopy
def baseline(x):
```

```
    pred = deepcopy(x)
    np.random.seed(10)
    np.random.shuffle(pred)
    return pred
base_res1 = NDCG_implicit(list(y_val), baseline(list(y_val)), k=10)
base_res2 = AP(list(y_val), baseline(list(y_val)), k=10)
base_res3 = precision_at_k(list(y_val), baseline(list(y_val)), k=10)

print(base_res1,base_res2,base_res3)

from copy import deepcopy

base_res_2 = AP(list(y_test), baseline(list(y_test)), k=10)
base_res_2

def train_algo(model, grid, metric):

    if model in set([Ridge, Lasso, SGDRegressor, ElasticNet]):
        X_train2 = mixpreprocessing(X_train)
        X_val2 = mixpreprocessing(X_val)
        X_test2 = mixpreprocessing(X_test)

    # finding the best model and parameters
    params_matrix = [grid[k] for k in grid.keys()]
    params_combinations = list(itertools.product(*params_matrix))
    all_combinations = []
    for pc in params_combinations:
        params = {}
        for k, i in zip(grid.keys(), range(0, len(pc))):
            params[k] = pc[i]
        all_combinations.append(params)


    best_res = 0
    best_params = all_combinations[0]
    res=0
    best_model=0
    temp = []
    error = 0
    for c in tqdm(all_combinations):
        try:
            model = model.set_params(**c)
            model.fit(X_train[feature_names], y_train)
            predict = model.predict(X_val[feature_names])
            met = metric(list(y_val), predict, k=10)
            temp.append(met)
```

```
            # conditions
            if(met>best_res):
                best_res = met
                best_params = c
                best_model = model
                ap1= AP(list(y_val), predict, k=10)
                precision_at_k1= precision_at_k(list(y_val), predict, k=10)
        except:
            error+=1
    print('Number of errors occured '+str(error))

    # testing results on testset
    predict = best_model.predict(X_test[feature_names])
    best_res_test = metric(y_test, predict, k=10)
    print('result on validation is '+str(best_res))
    print('result on test is '+str(best_res_test))
    diff = round(abs(best_res-best_res_test)/best_res*100,2)
    print('difference is '+str(round(diff))+'%')

    ap2= AP(list(y_test), predict, k=10)
    precision_at_k2= precision_at_k(list(y_test), predict, k=10)




    return best_res,best_res_test, ap1,ap2,precision_at_k1,precision_at_k2, best_params, best_model,
diff

import itertools
from tqdm import tqdm_notebook as tqdm
import warnings
warnings.filterwarnings('ignore')

dtr_grid = {
                'max_depth': [x for x in range(1,8)],
                'min_samples_split': [x for x in range(2,10)],
                'min_samples_leaf': [x for x in range(1,10)]
               }
rfr_grid = {
                'max_depth': [x for x in range(1,8)],
                'min_samples_split': [x for x in range(2,10)],
                'min_samples_leaf': [x for x in range(1,10)],
                  'random_state': [2]
               }
lir_grid = {
                'penalty': ['none'],
```

```
                'max_iter': [x*100 for x in range(10,100,10)],
                'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive']
                }
rr_grid = {
                'alpha': [x*0.1 for x in range(5,15)],
                'max_iter': [x*100 for x in range(10,100,10)],
                'solver' : ['auto','svd','cholesky','lsqr','sparse_cg','sag', 'saga']
                }
lar_grid = {
                'alpha': [x*0.1 for x in range(5,15)],
                'max_iter': [x*100 for x in range(10,100,10)]
                }
enr_grid = {
                'alpha': [x*0.1 for x in range(5,15)],
                'max_iter': [x*100 for x in range(10,100,10)],
                'l1_ratio' : [x*0.1 for x in range(1,9)]
                }
xgb_grid = {
                'max_depth': [x for x in range(2,4)],
                'learning_rate': [x*0.001 for x in range(0,100,7)],
                'n_estimators': [x*10 for x in range(5,15,2)],
                'reg_alpha': [x*0.05 for x in range(3,15,4)],
                'reg_lambda': [x*0.05 for x in range(3,14,4)],
                  'random_state': [2]
                }

grids = [dtr_grid, rfr_grid , lir_grid, rr_grid, lar_grid, enr_grid, xgb_grid]
models = [DecisionTreeRegressor(), RandomForestRegressor(), SGDRegressor(), Ridge(), Lasso(),
ElasticNet(), XGBRegressor()]
names = ['Baseline', 'DecisionTreeRegressor', 'RandomForestRegressor', 'SGDRegressor',
       'Ridge', 'Lasso', 'ElasticNet', 'XGBRegressor']

best_res,best_res2,best_res3,best_res_test,best_res_test2,best_res_test3, best_params, best_model,
diff = [base_res1],[base_res2],[base_res3],[base_res1],[base_res2],[base_res3],[],[],[0]
for m,g in zip(models, grids):
   i1,i2,i12, i22,i13,i23, i3, i4, i6 = train_algo(m, g, NDCG_implicit)
   best_res.append(i1)
   best_res_test.append(i2)
   best_res2.append(i12)
   best_res_test2.append(i22)
   best_res3.append(i13)
   best_res_test3.append(i23)
   best_params.append(i3)
   best_model.append(i4)
   diff.append(i6)
```

```
names = ['Baseline', 'DecisionTree', 'RandomForest', 'SGD',
        'Ridge', 'Lasso', 'ElasticNet', 'XGBRegressor']

result_df = pd.DataFrame({'Algorithm': names, 'NDCG_val': best_res, 'NDCG_test': best_res_test,'AP_val': best_res2, 'AP_test': best_res_test2,'Precision_val': best_res3, 'Precision_test': best_res_test3,'diff_in_%':diff})
result_df
```