ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра механіки, автоматизації та інформаційних технологій

(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття рівня вищої освіти «магістр»

(рівень вищої освіти)

на тему

“Інформаційна технологія аналізу добробуту та потреб громадян віртуальної країни з метою прийняття відповідних рішень влади”

“Information technology for analyzing the well-being and needs of citizens of a virtual country for the purpose of making relevant decisions of the authorities”

Виконала: студентка денної форми навчання

спеціальності 126 – Інформаційні системи та технології

(шифр і назва напряму підготовки, спеціальності)

Освітня програма «Інформаційні системи та технології»

(назва освітньої програми)

Нуждіна Марина Ігорівна

(прізвище, ім’я, по-батькові)

Керівник док. тех. наук, проф. Волков В. Е.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент канд. фіз.-мат. наук, доц. Рачинська А. Л.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2024 р.

Завідувач кафедри

_____ _Алла РАЧИНСЬКА_

(підпис) (ім’я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «___» _____ 2024 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Володимир ВИЧУЖАНІН_

(підпис) (ім’я, прізвище)

Одеса - 2024

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Інформаційна технологія аналізу добробуту та потреб громадян віртуальної країни з метою прийняття відповідних рішень влади».

Мета дослідження – удосконалення та оптимізація процесу аналізу даних з перепису населення з метою прийняття відповідних рішень влади щодо добробуту громадян віртуальної країни за рахунок зниження впливу людського фактору шляхом розробки відповідної інформаційної технології.

Об'єкт дослідження – процеси та методи аналізу потреб громадян віртуальної країни та підтримки прийняття відповідних рішень влади.

Предмет роботи – оригінальна інформаційна технологія, яка використовується для аналізу добробуту та потреб громадян віртуальної країни та підтримки прийняття відповідних рішень влади.

Розроблено інформаційну технологію, яка поєднує кластеризацію даних з перепису населення з використанням методу кластеризації категоріальних даних K-modes для виявлення основних соціальних груп, які мають схожі потреби, та прогнозування майбутніх потреб громадян на основі поточних та попередніх даних про громадян з використанням моделі лінійної регресії.

В результаті виконання кваліфікаційної роботи було розроблено 4 проекти. Розробка проводилася у СУБД PostgreSQL із використанням декларативної мови програмування SQL, IDE “Visual Studio” із використанням мови програмування високого рівня C# та веб-інтерактивного обчислювального середовища Jupyter Notebook із використанням мови програмування високого рівня Python.

Кваліфікаційну роботу виконано на 110 сторінках. Робота містить 15 рисунків та 8 додатків.

ANNOTATION

The qualification work develops the topic "Information technology for analyzing the well-being and needs of citizens of a virtual country for the purpose of making relevant decisions of the authorities".

The purpose of the research is to improve and optimize the process of analyzing census data for the purpose of making relevant decisions of the authorities regarding the well-being of citizens of a virtual country by reducing the influence of the human factor through the development of appropriate information technology.

The object of the research is the processes and methods of analyzing the needs of citizens of a virtual country and supporting the adoption of appropriate decisions by the authorities.

The subject of the work is an original information technology used to analyze the well-being and needs of citizens of a virtual country and supporting the adoption of appropriate decisions by the authorities.

An information technology has been developed that combines the clustering of census data with the use of the K-modes categorical data clustering method to identify the main social groups that have similar needs and predicting the future needs of citizens based on current and previous data about citizens using a linear regression model.

As a result of the qualification work, 4 projects were developed. The development was carried out in the PostgreSQL DBMS using the declarative programming language SQL, the IDE “Visual Studio” using the high-level programming language C# and the web-interactive computing environment Jupyter Notebook using the high-level programming language Python.

The qualification work was completed on 110 pages. The work contains 15 figures and 8 appendices.

ЗМІСТ

Стор.

ВСТУП

Перепис населення спрямований на отримання та аналіз об'єктивної інформації щодо змін у соціально-економічному житті та устрої держави з часу попереднього перепису.

Мета такого заходу, в першу чергу, полягає в створенні інформаційної бази демографічних та соціально-економічних даних віртуальної країни, включаючи чисельність її населення, його розподіл за статевою ознакою та віком, національним, мовним та сімейним складом, іншим громадянством, рівнем освіти, джерелами засобів існування, зайнятістю, міграційною активністю та житловими умовами як в країні загалом, так і в її адміністративно-територіальних одиницях.

Дані переписів населення є підґрунтям для численних політико-економічних та соціально-економічних досліджень.

Жодне змістовне дослідження не може бути проведеним без опори на інформацію, отриману в ході перепису. На основі даних, зібраних під час перепису населення, приймається велика кількість важливих рішень з приводу формування інфраструктури країни. [1]

Дані перепису використовуються для прогнозування демографічних та соціально-економічних тенденцій розвитку країни, що дозволяє розробляти державні програми та стратегії розвитку в таких сферах, як освіта, охорона здоров'я, зайнятість, соціальний захист, житлова політика тощо.

Цей процес є трудомістким і відповідальним, тому він вимагає ретельного аналізу з боку держави.

Метою дослідження є удосконалення та оптимізація процесу аналізу даних з перепису населення з метою прийняття відповідних рішень влади щодо добробуту громадян віртуальної країни за рахунок зниження впливу людського фактору шляхом розробки відповідної інформаційної технології.

Об'єктом роботи є саме процеси та методи аналізу потреб громадян віртуальної країни та підтримки прийняття відповідних рішень влади.

Предметом роботи є оригінальна інформаційна технологія, яка використовується для аналізу добробуту та потреб громадян віртуальної країни та підтримки прийняття відповідних рішень влади.

Для досягнення мети необхідно вирішити наступні задачі:

1) проаналізувати предметну область добробуту та потреб громадян віртуальної країни;
2) дослідити наявні методи прийняття рішень;
3) згенерувати віртуальну інформацію з перепису населення за допомогою окремого програмного додатку;
4) дослідити наявні методи аналізу даних;
5) обрати метод аналізу даних для виявлення основних соціальних груп, які мають схожі потреби;
6) дослідити наявні методи прогнозування;
7) обрати метод для прогнозування майбутніх потреб громадян на основі поточних даних про громадян;
8) реалізувати обрані методи у вигляді інформаційної технології аналізу добробуту та потреб громадян віртуальної країни з метою прийняття відповідних рішень влади;
9) провести аналіз ефективності розробленої інформаційної технології.

1 ОГЛЯД НАЯВНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ

1.1 Підходи до обробки інформації

Ефективне планування міської інфраструктури є ключовим фактором забезпечення сталого міського розвитку та покращення якості життя мешканців міст.

В умовах посилення урбанізації та соціально-економічних змін органи місцевого самоврядування стикаються з необхідністю приймати рішення на основі комплексного аналізу даних про населення.

Для прискорення процесу аналізу даних з перепису населення та прийняття рішень влади щодо добробуту громадян віртуальної країни розробляється інформаційна технологія аналізу добробуту та потреб громадян віртуальної країни.

Для реалізації даної інформаційної технології можна застосовувати декілька різних підходів.

1.1.1 Базовий підхід

Перший підхід до реалізації інформаційної технології заснований на аналізі поточного добробуту громадян. У даному випадку не прогнозуються майбутні потреби населення і не відбувається сегментації населення для визначення окремих потреб кожного сегменту населення.

У даному підході використовуються загальні статистичні дані про населення міста: чисельність населення, загальна кількість іноземців, загальна кількість безробітних і т. д.

Специфічна потреби різних соціальних груп, таких як молодь, пенсіонери, сім'ї з дітьми, люди з інвалідністю не враховуються.

Оскільки цей підхід не вимагає складного аналізу або прогнозування, він є найпростішим у плані реалізації, і прийняття рішень владою може відбуватися швидше.

Однак, ігнорування потреб різних соціальних груп може призвести до того, що ухвалені рішення не відповідатимуть реальним запитам населення, а відсутність прогнозування майбутніх потреб громадян загрожує виникненням проблем, яких можна було б уникнути.

Такий підхід може бути корисним для швидкого реагування на поточні проблеми суспільства, але він має суттєві обмеження.

Для довгострокового планування та ефективного розвитку міст доцільніше використовувати більш комплексні підходи.

1.1.2 Аналіз потреб без прогнозування майбутніх вимог

Другий підхід базується на аналізі поточних потреб та проблем різних соціальних груп населення без прогнозування майбутніх потреб громадян.

У даному випадку сегментація використовується для того, щоб зосередитися на вирішенні поточних проблем окремих груп громадян.

Сегментація населення за відбувається за різними ознаками (вік, стать, сімейний стан, наявність роботи) або комбінаціями ознак для того, щоб проаналізувати добробут певних соціальних груп.

Основою для прийняття рішень владою є поточні дані, отримані з перепису населення.

Оскільки аналіз відбувається на основі поточних даних про громадян, зі сторони влади забезпечується прийняття більш актуальних рішень, а реагування на наявні проблеми є швидким.

Вирішення конкретних проблем різних груп населення сприяє підвищенню якості життя громадян в короткостроковій перспективі.

Однак без прогнозування майбутніх потреб населення країна може виявитися неготовою до змін, таких як демографічний ріст або економічні кризи.

Також без прогнозування інформаційна технологія може пропонувати обмежений набір рішень або заходів для кожного кластера громадян.

Цей підхід добре підходить для швидкого реагування на поточні потреби, але може потребувати доповнення іншими методами.

1.1.3 Прогнозування майбутніх вимог без урахування потреб

Третій підхід фокусується на прогнозуванні майбутніх потреб населення, але без детального аналізу поточних потреб.

Цей підхід використовує прогнозні моделі, які допомагають визначити майбутні потреби міста.

Як і в першому підході, використовуються загальні статистичні дані про населення міста, але з акцентом на прогнозування майбутніх змін.

Прогнозування допомагає підготуватися до майбутніх демографічних та економічних змін та допомагає оптимально розподіляти ресурси, що може знизити витрати у довгостроковій перспективі.

Однак, якщо ігнорувати поточні специфічні потреби різних соціальних груп, окремі категорії населення можуть виявитися незадоволеними.

Також, інформаційна технологія працюватиме із загальним набором даних, не враховуючи можливі відмінності та специфіку різних груп громадян. Це може призвести до упущення важливих відмінностей між різними групами населення, що може знизити точність прогнозів.

Цей підхід зосереджений на довгостроковому плануванні та підготовці до майбутніх змін, що є його основною перевагою, проте відсутність врахування поточних потреб і проблем різних соціальних груп може призвести до того, що частина населення залишиться незадоволеною або відчуватиме нестачу необхідних послуг.

Таким чином, даний підхід не є оптимальним для ефективного вирішення проблем у містах.

1.1.4 Комплексний підхід

Четвертий підхід є найбільш комплексним і поєднує детальний аналіз поточних потреб та проблем різних соціальних груп населення з прогнозуванням майбутніх вимог до інфраструктури.

Цей підхід спрямований на створення стратегії розвитку, яка враховує як поточні, так і майбутні потреби міста.

Основні характеристики:

1) проводиться детальний аналіз потреб та проблем різних категорій населення (молодь, пенсіонери, люди з інвалідністю, бездомні тощо);
2) використовуються методи прогнозування для визначення майбутніх потреб в інфраструктурі, враховуючи демографічні, економічні та соціальні тенденції;
3) розробляються довгострокові плани розвитку інфраструктури, які враховують як поточні, так і прогнозовані потреби міста.

Переваги підходу:

1) сегментація дозволяє виділити схожі групи споживачів або об'єктів, а потім окремо прогнозувати для кожної групи. Це покращить точність прогнозів, оскільки моделі будуть більш адаптовані до конкретних характеристик кожної групи;
2) комбінування сегментації та прогнозування може дати більш гнучку систему, яка може пристосовуватися до різноманітних ситуацій і змінюватися з часом. Це допоможе забезпечити більш ефективне прийняття рішень у реальному часі;
3) сегментація може допомогти виявити складні взаємозв'язки між змінними та групами об'єктів. Поєднання цієї інформації з прогнозуванням може допомогти краще зрозуміти та врахувати ці залежності при прийнятті рішень;

4) комбінування сегментації та прогнозування може допомогти уникнути помилкових висновків, оскільки аналіз виконується на рівні кожного кластера, а не на загальному рівні всіх даних.

Недоліки підходу:

1) цей підхід вимагає значних ресурсів для збору даних, аналізу та прогнозування;
2) незважаючи на використання прогнозування, майбутні зміни можуть бути непередбачуваними, що вимагає постійного моніторингу та коригування планів.

Цей підхід є найбільш ефективним для створення стійкої та адаптивної міської інфраструктури, яка враховує як поточні, так і майбутні потреби різних соціальних груп населення.

Реалізація даного підходу вимагає значних ресурсів та постійного моніторингу для досягнення найкращих результатів.

1.2 Методи, які можна застосувати для визначення потреб та проблем різних соціальних груп

Розділення населення на групи сприяє ефективності управління, дозволяючи приймати більш обґрунтовані управлінські рішення.

Розглянемо методи, які можна застосувати для визначення потреб та проблем різних соціальних груп.

1.2.1 Класифікація чи кластеризація

Класифікація та кластеризація – це два основні підходи до аналізу даних у галузі машинного навчання та штучного інтелекту, які використовуються для розпізнавання закономірностей та групування даних.

Методи класифікації використовують навчені моделі для присвоєння кожного спостереження певному класу чи категорії.

Мета класифікації полягає у тому, щоб навчитися визначати, до якої категорії належить новий, невідомий об'єкт на основі його характеристик.

Коли дані не мають вже призначених класів або мають неоднорідний розподіл, класифікація стає складною.

У таких ситуаціях ***кластеризація*** дозволяє групувати схожі екземпляри разом без попередньої інформації про класи.

Мета кластеризації – виявити внутрішню структуру даних і розділити їх на кластери так, щоб об'єкти в одному кластері були максимально схожими один на одного і максимально відрізнялися від об'єктів в інших кластерах.

Таким чином, класифікація використовується, коли присутні мічені дані. Це означає, що кожен приклад у наборі даних має відповідну мітку або категорію.

Кластеризація використовується, коли немає мічених даних і необхідно зрозуміти структури даних, знайти подібності між об’єктами або виявити приховані патерни (див. рис. 1.2.1.1). [2]

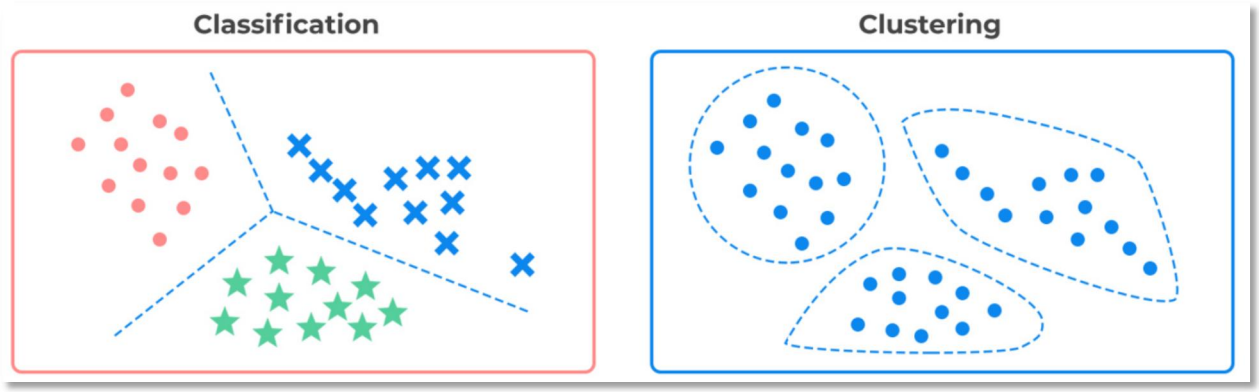

Рисунок 1.2.1.1 – Різниця між класифікацією та кластеризацією

Отже, оскільки у даному випадку немає конкретних категорій для нових об'єктів і ми хочемо виявити природні групи або структури в даних, використовувати кластеризацію буде доцільніше.

1.2.2 Алгоритми кластеризації

K-modes — це метод кластеризації, який використовується для групування даних з категоріальними (якісними) ознаками.

Мета цього методу — об’єднати об’єкти в групи (кластери) так, щоб об’єкти всередині одного кластера були максимально схожими, а об’єкти з різних кластерів відрізнялися.

K-modes підходить для категоріальних даних: наприклад, для даних про стать, національність, професію або освіту, які не можна порівнювати за математичними формулами.

У **K-modes** кластер описується не середнім значенням, а модою, тобто найпоширенішим значенням категоріальних ознак у кластері.

Для вимірювання "відстані" між двома об’єктами використовується кількість відмінностей між їх категоріальними ознаками (метрика Геммінга).

Як працює **K-modes:**

1) алгоритм починає з випадкового вибору декількох об’єктів, які будуть початковими центрами кластерів;
2) кожен об’єкт даних порівнюється з центрами кластерів, і його відносять до того кластеру, де "відстань" до центру мінімальна;
3) центр кожного кластеру оновлюється шляхом знаходження моди для кожної ознаки;
4) кроки 2 і 3 повторюються, доки не буде досягнуто стабільності (об'єкти перестануть переходити між кластерами). [3]

K-medoids — це метод кластеризації, який групує дані в кілька кластерів, обираючи як представника кожного кластера один із реальних об'єктів (названий медоїдом), чиї відмінності з усіма іншими об’єктами в кластері мінімальні.

Алгоритм підходить як для числових, так і для категоріальних даних, оскільки він працює з фактичними об’єктами, а не з їх середніми значеннями.

K-medoids працює з будь-якою мірою відстані, що робить його більш гнучким для різних типів даних.

Як працює **K-medoids:**

1) спочатку випадково обирається k об'єктів як медоїди;
2) кожен об'єкт даних відноситься до кластера, медоїдом якого є найближчий об'єкт (згідно з вибраною метрикою відстані);
3) для кожного кластера обирається новий медоїд — об’єкт, який мінімізує суму відстаней до інших об’єктів у цьому кластері;
4) кроки 2 і 3 повторюються, поки медоїди не перестануть змінюватися (або поки зміни не стануть дуже малими).

Основним недоліком **K-medoids** є те, що він не підходить для кластеризації несферичних (довільної форми) груп об’єктів та може отримати різні результати для різних прогонів на одному наборі даних, оскільки перші k медоїдів вибираються випадковим чином. [4]

Агломеративна ієрархічна кластеризація — це метод кластеризації, що послідовно об'єднує об'єкти у все більші групи (кластери) на основі їх схожості. Він належить до ієрархічних методів кластеризації, оскільки створює структуру кластерів у вигляді дерева (дендрограми).

Як працює агломеративна кластеризація:

1) кожен об'єкт на початку є окремим кластером;
2) визначається відстань або схожість між кожною парою кластерів;
3) два найближчі кластери (з найменшою відстанню) об'єднуються в один;
4) після об'єднання кластерів відстані між новим кластером і всіма іншими кластерами обчислюються знову;
5) процес об'єднання та оновлення повторюється, поки всі об'єкти не будуть об'єднані в один кластер або поки не буде досягнуто заданої кількості кластерів.

Способи визначення відстані/подібності між кластерами:

1) мінімальна відстань: знаходиться мінімальна відстань між будь-якими двома точками кластера;

2) максимальна відстань: знаходиться максимальна відстань між будь-якими двома точками кластера;
3) середнє значення групи: знаходиться середня відстань між кожними двома точками кластерів;
4) метод Уорда: подібність двох кластерів заснована на збільшенні квадратичної помилки, коли два кластери об’єднуються. [5]

K-modes є найкращим вибором для кластеризації категоріальних даних, таких як стать, професія чи освіта. На відміну від інших методів, він використовує просту й зрозумілу метрику Геммінга для оцінки схожості, а центром кластера визначає моду — найпоширеніше значення ознаки. Це робить алгоритм швидким і зручним.

Порівняно з агломеративною ієрархічною кластеризацією, **K-modes** значно ефективніший для великих наборів даних.

Таким чином, для етапу кластеризації інформаційної технології використовуватиметься **K-modes**.

1.3 Методи, які можна застосувати для прогнозування майбутніх вимог до інфраструктури

Завдяки прогнозуванню майбутніх потреб громадян можна забезпечити своєчасне надання необхідних послуг і покращити якість життя громадян.

Розглянемо методи, які можна застосувати для виконання даної задачі.

1.3.1 Машинне навчання, статистичні моделі чи експертні оцінки

Три основні категорії методів прогнозування — це машинне навчання, статистичні моделі та експертні оцінки.

Методи машинного навчання використовують алгоритми, які дозволяють комп'ютерам навчатися на основі даних і автоматично виявляти закономірності, що дозволяє робити точні та масштабовані прогнози.

Даний метод видає високу точність прогнозів завдяки здатності моделювати складні нелінійні залежності.

Немає необхідності обробляти дані вручну: обробка великих обсягів даних та виявлення закономірностей відбувається автоматично.

Моделі можуть бути регулярно оновлювані новими даними для повищення точності прогнозів.

З недоліків є факт, що деякі моделі можуть бути важкими для інтерпретації та вимагати значних обчислювальних ресурсів для навчання та прогнозування.

Статистичні моделі базуються на математичних моделях і теорії ймовірності, надаючи чіткі й інтерпретовані результати.

Статистичні моделі простіше інтерпретувати та пояснити, ніж складні моделі машинного навчання.

На відміну від машинного навчання, статистичні моделі можуть працювати з меншими наборами даних та зазвичай менше схильні до перенавчання.

Їх можна швидко обчислити навіть для великих наборів даних.

Однак, багато статистичних моделей роблять припущення про лінійність або нормальність розподілу даних, що може бути обмежувальним фактором.

Статистичні моделі також показують меншу точність у порівнянні з методами машинного навчання для складних задач.

Зазвичай, статистичні моделі вимагають більше ручної роботи для підготовки даних та моделювання.

Присутня також проблема з обробкою різноманітних типів даних: методи машинного навчання є більш гнучкими у цьому плані.

Експертні оцінки спираються на знання і досвід людей-експертів. Використовуючи даний метод, існує можливість враховувати контекстуальні фактори і робити прогнози в ситуаціях, де автоматичні моделі можуть бути недостатньо ефективними.

Експертні оцінки можуть адаптуватися до унікальних та нестандартних ситуацій, а у випадках, де дані недостатні або неповні, для прийняття рішень можуть використовуватися інтуїтивні судження.

Однак, рішення можуть бути суб'єктивними, менш адаптивними до нових даних або змін у середовищі та залежати від особистих упереджень експертів.

У даному випадку важко обробляти велику кількість прогнозів.

Також різні експерти можуть давати різні прогнози на основі однакових даних, що може призводити до непослідовності рішень. [6]

У даному випадку найкращим вибором для прогнозування майбутніх вимог до інфраструктури є використання машинного навчання.

Машинне навчання дозволяє автоматично виявляти складні патерни та залежності у даних про громадян.

Цей підхід забезпечує високу точність прогнозів завдяки здатності моделей адаптуватися до змін у даних та середовищі.

Крім того, машинне навчання дозволяє автоматизувати процес аналізу та прийняття рішень, що є критичним для ефективного управління інфраструктурою у різних містах країни.

1.3.2 Алгоритми машинного навчання

Лінійна регресія — це техніка машинного навчання, що застосовується для прогнозування значень, які варіюються в безперервному діапазоні.

Лінійна регресія встановлює зв'язок між вхідною і вихідною змінними, що можна виразити у вигляді прямої лінії.

У цьому процесі лінійна регресія аналізує набір даних з відомими значеннями і знаходить найкращу лінію, що описує ці точки. Ця лінія, відома як «лінія регресії», використовується для прогнозування значення вихідної змінної на основі вхідної.

Алгоритми ***Gradient Boosting*** використовують ансамблевий метод, створюючи серію "слабких" моделей, які поступово покращуються через кілька ітерацій для формування сильної прогнозної моделі.

Кожен новий етап спрямований на виправлення помилок попередніх моделей, що дозволяє значно зменшити їх похибки та отримати точніший результат.

К-найближчий сусід (KNN) — це метод класифікації та прогнозування, який визначає результат на основі близькості нової точки до інших точок даних.

Наприклад, якщо для класифікації використовується параметр K=5, алгоритм аналізує 5 найближчих сусідів і на основі більшості з них визначає клас для нової точки. [7]

У даному випадку лінійна регресія ідеальна для прогнозування кількості людей у кластері, тому що лінійна регресія ідеально підходить для прогнозування числових значень, таких як кількість осіб.

Лінійна регресія проста, швидка та дозволяє чітко інтерпретувати результати.

1.4 Опис інформаційної технології

Для підтримки прийняття рішень з розвитку інфраструктури віртуальної країни розробляється інформаційна технологія, яка враховуватиме результати аналізу даних та прогнози, які отримані на основі моделі машинного навчання.

Ця технологія буде здатна визначати потреби, які існують у громадян у кожному місті залежно від характеристик населення.

Компоненти інформаційної технології:

1) збір даних про громадян: збір даних відбувається у застосунку, призначеному спеціально для збору інформації про громадян, яка

включає особисті характеристики (вік, стать, освіта), економічний статус (професія, наявність роботи), сімейний стан і т. д.;

2) первинна обробка даних: після збору дані проходять через етап первинної обробки, де вони перетворюються у відповідний формат для подальшого аналізу;
3) виділення соціальних груп громадян: для ідентифікації соціальних груп громадян використовується алгоритм **K-modes**;
4) побудова моделі прогнозування: для прогнозування майбутніх вимог до інфраструктури використовується модель лінійної регресії;
5) графічне представлення результатів: отримані результати, включаючи кластери громадян і прогнозовані вимоги до інфраструктури, відображаються у вигляді графіків для зручного сприйняття управлінцями;
6) формування звітів: інформаційна технологія формує звіти з аналізом добробуту та потреб громадян для кожного міста віртуальної країни. Важливо зазначити, що зазначена технологія не прийматиме рішення самостійно, а лише надаватиме рекомендації, допомагаючи особам, які приймають рішення, робити остаточні висновки.

2 ЕТАП ЗБОРУ ДАНИХ З ПЕРЕПИСУ НАСЕЛЕННЯ

2.1 Перелік запитань, за якими здійснюється збирання основних первинних (персональних) даних

Для того, щоб зібрати дані з перепису населення та проводити аналіз, спочатку необхідно визначитися з даними, які будуть необхідні для проведення аналізу. Було прийнято рішення базуватися на Програмі Всеукраїнського перепису населення 2023 року.

Перелік запитань, за якими здійснюється збирання основних первинних (персональних) даних:

1) число повних років;

2) стать;

3) місце проживання;

4) сімейний (шлюбний) стан (для осіб віком 15 років і старше) (ніколи не перебував(ла) у шлюбі, перебуваю в зареєстрованому шлюбі, перебуваю в незареєстрованому шлюбі, удівець, удова, розлучений(а) (зареєстроване розірвання шлюбу), розійшовся(лася) (незареєстроване розірвання шлюбу));

5) етнічне походження (за самовизначенням респондента);

6) рідна мова;

7) володіння українською мовою;

8) громадянство (громадянство України, громадянство іншої держави (зазначте державу), особа без громадянства, громадянство не визначене);

9) рівень освіти (для осіб віком 10 років і старше) (третій (освітньо-науковий/освітньо-творчий) рівень вищої освіти, другий (магістерський) рівень вищої освіти, перший (бакалаврський) рівень вищої освіти (базова вища), початковий рівень (короткий цикл) вищої освіти (неповна вища, середня спеціальна), фахова передвища, професійна (професійно-технічна), профільна середня (повна загальна середня), базова середня (неповна середня), початкова (початкова загальна), не маю освіти);

10) уміння читати і писати;

11) отримання освіти та тип закладу освіти, у якому навчається респондент (для осіб віком 6 років і старше) (заклад вищої освіти, заклад фахової передвищої освіти, заклад професійної (професійно-технічної) освіти, заклад загальної середньої освіти, інший заклад освіти);

12) відвідування дитиною дошкільного закладу освіти (для осіб віком до восьми років);

13) наявність роботи впродовж тижня, який передував даті перепису населення, з метою отримання оплати або доходу (прибутку) у грошовому або натуральному вигляді;

14) професія;

15) місцезнаходження основної роботи;

16) причина, через яку респондент не працює в населеному пункті, де проживає (не знайшов(ла) роботу в цьому населеному пункті, знайшов(ла) роботу з більш високою заробітною платою, інші соціально-побутові вигоди, інші причини);

17) пошук респондентом оплачуваної роботи;

18) наявність проблем із ходінням, зором, слухом, когнітивними здібностями, доглядом за собою, спілкуванням (розподіленими за категорією відповіді: ”Ні” - не відчуваю труднощів, "Так"- відчуваю деякі труднощі, "Так”- відчуваю великі труднощі, "Взагалі не можу цього зробити") (за самооцінкою*)*;

19) тип будівлі та житлового приміщення (квартира у дво- або багатоквартирному будинку, квартира в нежитловій будівлі, одноквартирний будинок (індивідуальний), частина одноквартирного будинку (індивідуального), кімната (частина кімнати) у гуртожитку, кімната (частина кімнати) у готелі, установа із рухомим складом населення, інституційна установа, інше житлове приміщення, житлового приміщення немає (безпритульний));

20) доступ домогосподарства до Інтернету (у тому числі до рухомого/мобільного Інтернету);

21) розмір загальної площі житлового приміщення (у цілих кв. м);

22) тип володіння житловим приміщенням (приватна власність членів домогосподарства, державна, комунальна, приватна власність юридичних осіб, оренда (найм) у окремих громадян, не знаю). [8]

2.2 Програмні засоби. PostgreSQL

PostgreSQL — це потужна об’єктно-реляційна система управління базами даних із відкритим вихідним кодом, яка поєднує в собі розширені можливості SQL із додатковими функціями для безпечного зберігання та масштабування навіть найскладніших наборів даних.

Ця СУБД є популярною завдяки перевіреній архітектурі, високій надійності, збереженню цілісності даних та розширюваним функціональним можливостям.

Важливу роль у її популярності відіграє активна спільнота, яка постійно вдосконалює систему, впроваджуючи ефективні та інноваційні рішення.

PostgreSQL має широкий набір функцій, які роблять її незамінною для різних завдань.

Вона допомагає розробникам створювати високопродуктивні програми, адміністраторам — підтримувати цілісність даних і створювати відмовостійкі середовища, а бізнесам — керувати великими та малими наборами даних із максимальною ефективністю.

Окрім того, що PostgreSQL є безкоштовною та має відкритий вихідний код, вона відзначається високим рівнем розширюваності. Користувачі можуть створювати власні типи даних, додавати функції, а також писати код із використанням різних мов програмування без необхідності перекомпілювати базу даних.

Ця гнучкість робить PostgreSQL ідеальним вибором для проєктів будь-якого масштабу. [9]

2.3 Програмні засоби. C#

C# — це універсальна, об’єктно-орієнтована мова програмування високого рівня, яка працює як на платформах із відкритим кодом, так і на архітектурі Microsoft Windows .NET.

Популярність C# зумовлена низкою переваг, зокрема зручністю для створення різноманітних додатків завдяки можливості забезпечувати інтерактивне середовище для користувачів.

C# також вирізняється відносною легкістю у вивченні, що робить її доступнішою для новачків у програмуванні.

Функції C# сприяють швидшій розробці, а простота й ефективність коду зменшують час, необхідний для налагодження.

C# можна використовувати для розробки різних типів програм, програмного забезпечення й платформ.

Сильні сторони C# включають велику й активну спільноту розробників по всьому світу, які створюють численні ресурси для підтримки та обміну знаннями.

C# також має високу масштабованість, що спрощує підтримку та розширення проєктів, а також інтеграцію з іншими технологіями, такими як хмарні сервіси Microsoft Azure.

Ці особливості роблять C# однією з найпопулярніших мов програмування для різноманітних завдань. [10]

2.4 Проектування бази даних

На основі переліку питань, за якими здійснюється збирання основних первинних (персональних) даних, можна визначитися зі структурою бази даних.

База даних міститиме дві таблиці: одна з них відповідатиме за питання, які відносяться до домогосподарства (**`Household`**), інша за питання, які

відносяться безпосередньо до людини, яка проживає у даному домогосподарстві (**`Citizen`**).

Всі запити на створення таблиць бази даних наведено у додатку А.2.

Одним із засобів графічного представлення предметної області є діаграма «сутність-зв'язок» (ER – діаграма). Для її побудови використаємо програмний застосунок DBeaver. ER-діаграма приведена на рисунку 3.2.1.

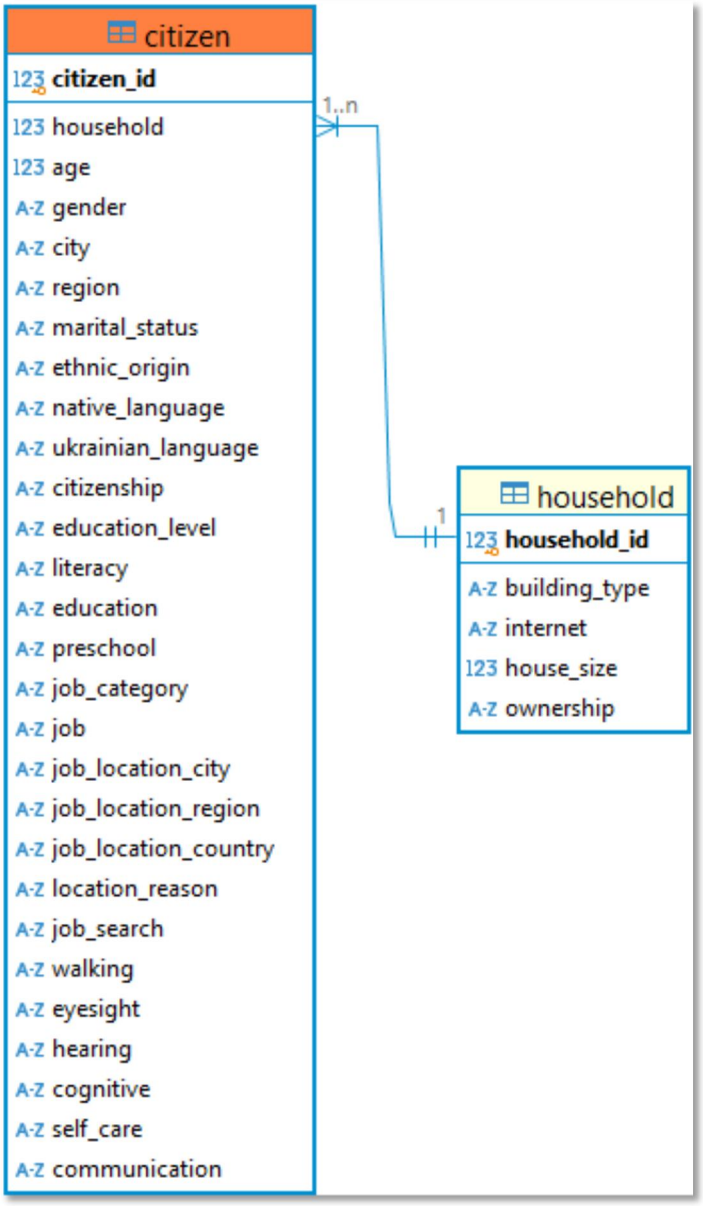

Рисунок 2.4.1 – ER-діаграма бази даних

Людина має належати до домогосподарства, тому таблиця, яка відповідає питанням, які відносяться до самої людини, має містити

первинний ключ домогосподарства, до якого належить людина, у якості зовнішнього ключа.

Ці таблиці мають зв’язок один-до-багатьох (до кожного домогосподарства може належати скільки завгодно людей, але кожна людина може належати тільки до одного домогосподарства).

Поля таблиці **`Household`**, які мають обов’язково бути заповнені: “Тип будівлі та житлового приміщення”, “Доступ домогосподарства до Інтернету”.

Поля таблиці **`Household`**, які можуть приймати значення null: “Розмір загальної площі житлового приміщення” та “Тип володіння житловим приміщенням”.

Поля таблиці **`Citizen`**, які мають обов’язково бути заповнені: “id домогосподарства”, “Число повних років”, “Стать”, “Місто проживання”, “Область проживання”, “Сімейний (шлюбний) стан”, “Етнічне походження”, “Рідна мова”, “Володіння українською мовою”, “Громадянство”, “Рівень освіти”, “Уміння читати і писати”, “Отримання освіти та тип закладу освіти, у якому навчається респондент”, “Наявність проблем із ходінням”, “Наявність проблем із зором”, “Наявність проблем із слухом”, “Наявність проблем із когнітивними здібностями”, “Наявність проблем із доглядом за собою”, “Наявність проблем із спілкуванням”.

Поля таблиці **`Citizen`**, які можуть приймати значення null: “Відвідування дитиною дошкільного закладу освіти”, “Професія”, “Місцезнаходження основної роботи”, “Причина, через яку респондент не працює в населеному пункті, де проживає”, “Пошук респондентом оплачуваної роботи”.

Для даної бази даних буде корисним обмежити значення, які можуть бути введені в певному стовпці таблиці.

У тих випадках, де поле може приймати лише якесь певне значення, були визначені домени, більшість з яких обмежують значення, яке може приймати поле, таким чином, щоб був доступний вибір із кількох певних значень.

Усі запити на створення доменів бази даних наведено у додатку А.1.

Розглянемо в якості прикладу один з доменів, **marital_status_d**. Даний домен дозволяє обмежити значення, які може приймає поле “Сімейний (шлюбний) стан”, таким чином, щоб були доступні лише значення 'ніколи не перебував(ла) у шлюбі', 'перебуваю в зареєстрованому шлюбі', 'удівець', 'удова', 'перебуваю в незареєстрованому шлюбі', 'розлучений(а) (зареєстроване розірвання шлюбу)', 'розійшовся(лася) (незареєстроване розірвання шлюбу)' та 'особа віком молодше 15 років'.

Окрім доменів, для деяких полей використовуються обмеження. Наприклад, “Розмір загальної площі житлового приміщення” має бути більшим за 0, тому влаштовується перевірка, щоб вказаний розмір житлової площі був додатною величиною:

```
constraint size_check check (house_size > 0)
```

Деякі поля можуть приймати значення “Так” або “Ні”: “Доступ домогосподарства до Інтернету”, ”Уміння читати і писати”, “Відвідування дитиною дошкільного закладу освіти”, “Наявність роботи впродовж тижня, який передував даті перепису населення”, “Пошук респондентом оплачуваної роботи”. У такому випадку використовується домен **yes_no**.

Для того, щоб уникнути помилок з боку користувача, було створено функції та тригери, які обмежують відповіді на питання для певних громадян (наприклад, громадяни старше за 8 років не матимуть можливості відповісти на питання щодо відвідування дитиною дошкільного закладу освіти).

Усі запити на створення тригерів і функцій бази даних наведено у додатку А.3.

Таким чином, маємо базу даних, яка зберігатиме зібрану інформацію про громадян.

2.5 Генерація віртуальної інформації з перепису населення за допомогою окремого програмного додатку

Спроектуємо програмний додаток у середовищі розробки Visual Studio 2022 Community, націлений на генерацію випадкових значень, які відповідають полям таблиць бази даних. У якості мови програмування використовуватиметься C#.

Весь код генерації даних надано у додатку Б.

Для генерації значень з певного переліку (по аналогії з доменами у PostgreSQL), будемо користуватися списками.

Таким чином, домен, який мав такий вигляд у PostgreSQL:

```
create domain citizenship_d as varchar
check(value in ('громадянство України', 'громадянство іншої держави', 'особа без громадянства', 'громадянство не визначене'));
```

матиме такий вигляд у C#:

```
static List<string> citizenship = new List<string> { "громадянство іншої держави", "особа без громадянства", "громадянство не визначене", "громадянство України" };
```

В основній функції програми використовується цикл `for()`, кожна ітерація якого генеруватиме значення для таблиці **`Household`**.

Кількість ітерацій обмежена лише продуктивністю машини, на якій запускається програма.

Коли всі значення будуть заповнені, вони будуть записані до спеціально створеного текстового файлу:

```
writer.WriteLine($"{household}|{random_building_type}|{random_internet}|{random_house_size}|{random_ownership}");
```

Далі генерується випадкова кількість жителів даного домогосподарства (1-8), після чого будуть заповнені дані кожного громадянина.

З більшою ймовірністю громадяни житимуть у великих містах та областях (наприклад, для Київської області та Києва буде згенеровано більше жителів, ніж для Одеської області та Ізмаїлу).

Для того, щоб ознаки не мали однаковий частотний розподіл у всіх містах, кожне значення генерується з різною ймовірністю відповідно до того, в якому місті проживає громадянин, який в нього рівень освіти, яка в нього професія й т. д.

Запустимо програму та подивимося на результуючий текстовий файл, який містить приблизно 400000 рядків:

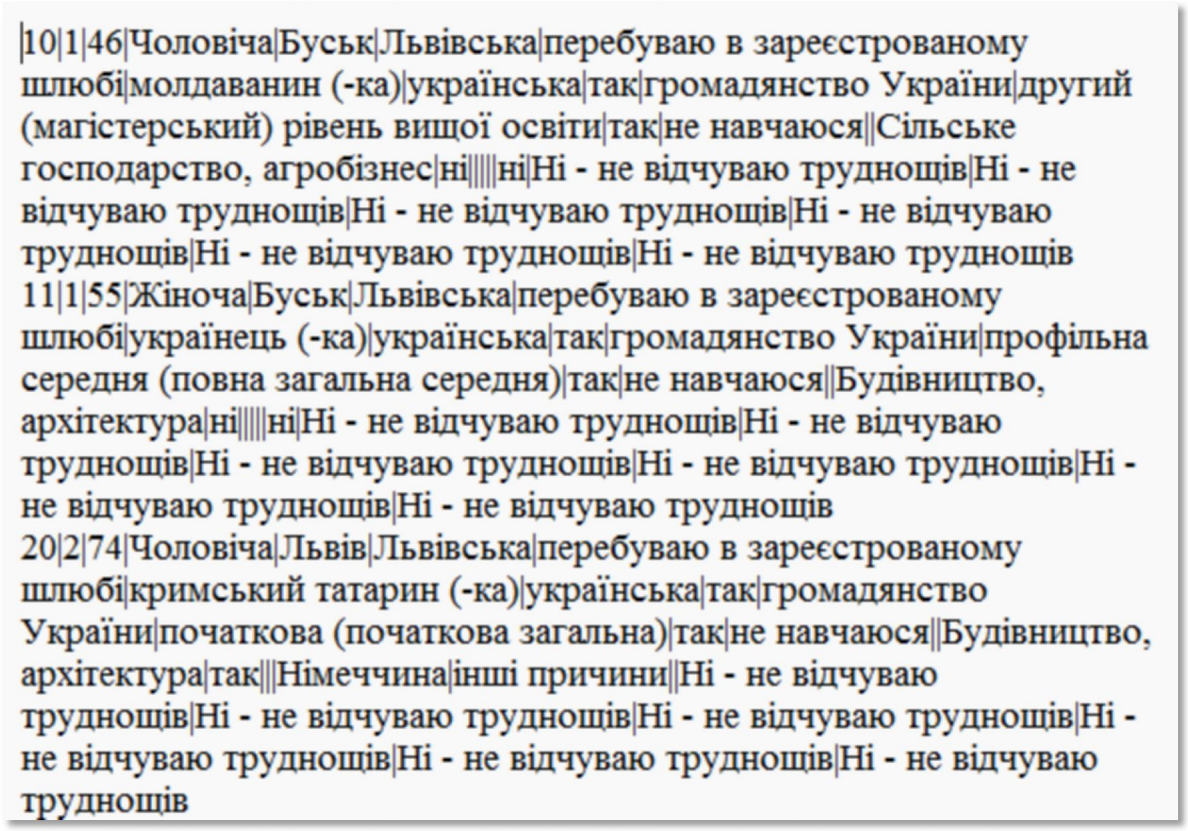
10|1|46|Чоловіча|Буськ|Львівська|перебуваю в зареєстрованому шлюбі|молдаванин (-ка)|українська|так|громадянство України|другий (магістерський) рівень вищої освіти|так|не навчаюся||Сільське господарство, агробізнес|ні|||||ні|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів
11|1|55|Жіноча|Буськ|Львівська|перебуваю в зареєстрованому шлюбі|українець (-ка)|українська|так|громадянство України|профільна середня (повна загальна середня)|так|не навчаюся||Будівництво, архітектура|ні|||||ні|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів
20|2|74|Чоловіча|Львів|Львівська|перебуваю в зареєстрованому шлюбі|кримський татарин (-ка)|українська|так|громадянство України|початкова (початкова загальна)|так|не навчаюся||Будівництво, архітектура|так|||Німеччина|інші причини||Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів|Ні - не відчуваю труднощів

Рисунок 2.5.1 – Текстовий файл з результатами

Для того, щоб у майбутньому скористатися прогнозуванням, були згенеровані не лише поточні дані, але й історичні дані з перепису населення (за десять, двадцять, тридцять років тому).

3 ЕТАП КЛАСТЕРИЗАЦІЇ

3.1 Метод ліктя

В алгоритмів кластеризації, які засновані на розділенні, присутній значний недолік — заздалегідь невідома точна кількість кластерів, які потрібно сформувати.

Таким чином, кількість кластерів, на які розбиваються дані, необхідно або обирати самостійно, або використовувати методи, спеціально призначені для визначення оптимальної кількості кластерів.

Метод "ліктя" — евристичний підхід, який застосовують у таких алгоритмах кластеризації, як **`K-modes`**, **`K-means`** і **`K-prototypes`**.

Коли кількість кластерів збільшується, загальна дисперсія кластерів швидко зменшується.

У алгоритмі K-modes дисперсія між кластерами вимірюється за допомогою відстаней між центрами кластерів. Однак тут ці "центри" є не середніми значеннями, а модами для кожного атрибута в кластері.

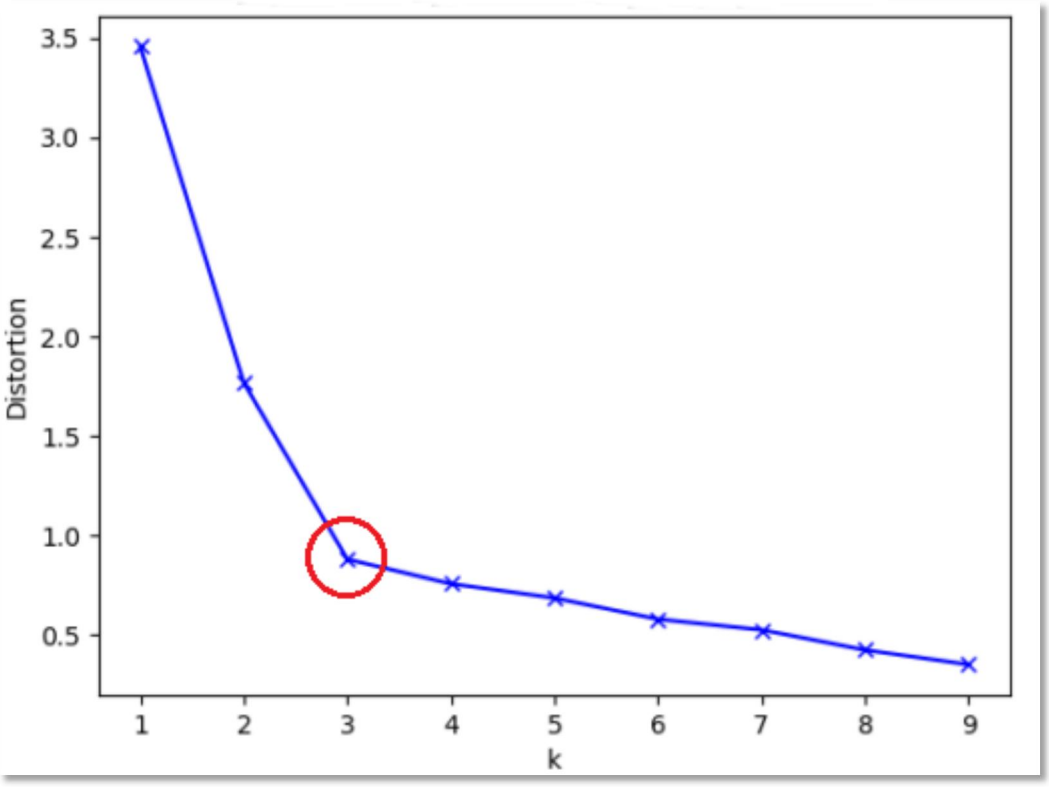

Рисунок 3.1.1 - Метод ліктя

Після досягнення певної точки процес зменшення дисперсії уповільнюється — дисперсія між кластерами або стає майже незмінною, або зменшується дуже повільно.

Якщо побудувати графік, що показує залежність загальної кластерної дисперсії від кількості кластерів, він набуває форми зігнутого ліктя.

Точка, після якої зниження дисперсії значно сповільнюється, називається "ліктем". Саме ця точка вважається оптимальним вибором для кількості кластерів. [11]

3.2 Програмні засоби. Python

Python — це високорівнева інтерпретована мова програмування загального призначення, відома своїм простим і зрозумілим синтаксисом.

Python здобув велику популярність у сфері науки про дані, машинного навчання та аналітики завдяки своїй гнучкості, багатій екосистемі бібліотек і активній спільноті користувачів.

Однією з ключових бібліотек для машинного навчання у Python є `Scikit-learn`. Вона надає реалізації популярних алгоритмів кластеризації, таких як K-Means, K-modes, DBSCAN, Affinity Propagation та Agglomerative Clustering.

Окрім Scikit-learn, Python має інші потужні бібліотеки, які створюють основу для ефективного кластерного аналізу:

1) NumPy — для числових обчислень;
2) Pandas — для обробки даних;
3) Matplotlib і Seaborn — для візуалізації кластерів.

Ці інструменти забезпечують аналітиків і науковців усім необхідним для проведення ефективного кластерного аналізу. [12]

3.3 Підготовка даних

Створимо **Notebook** документ веб-інтерактивного обчислювального середовища **Jupyter Notebook**, який назвемо **K-modes** (див. Додаток В).

Перед тим, як застосовувати кластеризацію до даних, необхідно спочатку обробити та підготувати дані.

Підключившись до бази даних, отримуємо список унікальних міст з таблиці **Citizen**. Це необхідно для того, щоб сформувати окремий набір даних для кожного міста.

Функція `process_city` відповідає за обробку даних для одного міста.

Сформувавши запит для поточного міста, виконуємо його та завантажуємо дані у новий набір даних. З цього набору видаляються стовпці `'citizen_id'`, `'household_id'`, `'household'`, `'city'` та `'region'`, оскільки вони заважатимуть аналізу.

Метод **K-modes** працює з категоріальними даними, тому для стовпців, які мають тип `integer` (стовпці `‘age’` та `‘house_size’`) створюються нові категорії, які об’єднують певні значення у групи.

Таким чином, формуються вікові групи `'0-17'`, `'18-29'`, `'30-44'`, `'45-59'`, `'60+'`.

Розмір житлової площі матиме такі групи: `'20-56'`, `'57-93'`, `'94-130'`, `'131-167'`, `'168-200'`.

Після цього стовпці `‘age’` та `‘house_size’` видаляються, оскільки вони не будуть безпосередньо використовуватися під час кластеризації.

Всі значення у сформованому наборі даних перетворюються на строковий тип.

Визначимо групи стовпців для кожної підмножини, за якою буде проводитися кластеризація: 'Відомості щодо житлових умов': `['house_size_interval', 'building_type', 'ownership']`, 'Відомості щодо етнічного походження': `['ethnic_origin',`

'native_language', 'ukrainian_language', 'citizenship'], 'Відомості щодо освіти': ['education_level', 'education', 'preschool'], 'Відомості щодо наявності роботи для різних сфер діяльності': ['job', 'job_category', 'location_reason'], 'Відомості щодо місцезнаходження основної роботи': ['job_category', 'job_location_city', 'job_location_country'], 'Відомості щодо статусу інвалідності': ['walking', 'eyesight', 'hearing', 'cognitive', 'self_care', 'communication'].

Далі проходимося по кожному місту та формуємо підмножини, обираючи лише потрібні стовпці. Кожна підмножина зберігається у словнику.

3.4 Застосування кластеризації до даних

Перейдемо до кластеризації даних.

Кожен результат кластеризації зберігатиметься в окремому файлі, тому спочатку ініціалізуємо директорію для збереження файлів.

Далі ітеруємо по кожному місту та його підмножинам, застосовуючи **K-modes**.

Оскільки K-modes не є методом, що здатний самостійно визначити оптимальну кількість кластерів, скористаємося методом ліктя.

Будемо пробувати кількість кластерів від 2 до 10, але з перериванням при досягненні ліктя.

Запустивши метод **K-modes** з певною кількістю кластерів, зберігаємо вартість кластеризації та мітки кластерів.

Перевіряємо, чи стало зменшення вартості незначним. Якщо так, лікоть знайдено, і цикл переривається

Якщо лікоть не було знайдено, обирається максимальна кількість кластерів (у даному випадку 10).

Після цього до кожного набору даних застосовуються оптимальні мітки кластерів, а результати кластеризації зберігаються в окремих файлах.

	A	B	C	D	E
1	job,job_category,location_reason,cluster				
2	так,"Бухгалтерія, аудит",інші соціально-побутові вигоди,1				
3	ні,"Бухгалтерія, аудит",None,1				
4	так,"Робочі спеціальності, виробництво",None,3				
5	ні,"Будівництво, архітектура",None,2				
6	так,"Бухгалтерія, аудит",None,1				
7	так,"Робочі спеціальності, виробництво",None,3				
8	так,"Бухгалтерія, аудит",None,1				
9	ні,"Сільське господарство, агробізнес",None,4				
10	None,None,None,0				
11	так,"Бухгалтерія, аудит",None,1				
12	ні,"Будівництво, архітектура",None,2				
13	ні,"Освіта, наука",None,4				
14	так,"Будівництво, архітектура",None,2				
15	так,"Бухгалтерія, аудит",None,1				
16	так,"Робочі спеціальності, виробництво",None,3				
17	ні,"Продаж, закупівля",None,4				
18	ні,"Бухгалтерія, аудит",None,1				
19	так,"Будівництво, архітектура",None,2				
20	так,"Бухгалтерія, аудит",None,1				

Рисунок 3.4.1 – Формат файлу з результатами кластеризації

3.5 Частотний розподіл ознак у кластерах

Частотний розподіл — це інструмент у статистиці, який допомагає упорядковувати дані з метою дійти значущих висновків. Він показує, як часто будь-які конкретні значення трапляються в наборі даних.

Частотний розподіл ілюструє, як часто кожне значення змінної зустрічається в наборі даних. Він зображає кількість випадків для кожного можливого значення в цьому наборі. [13]

Для наочності зробимо візуалізацію частотного розподілу різних ознак у кожному кластері за допомогою бібліотеки **`matplotlib.pyplot`** (див. Додаток В).

Частотний розподіл допоможе зрозуміти, як різні ознаки розподіляються серед різних кластерів.

Також частотний розподіл дасть нам змогу оцінити якість класифікації даних.

Якщо певні значення домінують у одному кластері, це може свідчити про хорошу кластеризацію.

З іншого боку, рівномірний розподіл може вказувати на проблеми в класифікації.

Всі графіки зберігатимуться в окремих файлах, тому вказуємо директорію для збереження зображень.

У циклі `for()` формуємо частотні таблиці для кожної ознаки, не рахуючи стовпця `'cluster'`.

Наведемо приклад результату частотної таблиці для ознаки `'ownership'` підмножини “Відомості щодо житлових умов”:

```
ownership  оренда (найм) у окремих громадян  \
cluster
0                                       2455
1                                        976

ownership  приватна власність членів домогосподарства  \
cluster
0                                                 2412
1                                                  969

ownership  приватна власність юридичних осіб
cluster
0                                       3295
1                                        171
```

Рисунок 3.5.1 - Вигляд частотної таблиці

На основі частотних таблиць створюються графіки з частотним розподілом ознак, які потім зберігаються у файлах формату .png.

Розглянемо декілька графіків, які були сформовані таким чином.

Перший графік показує частотний розподіл ознаки `'job'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса.

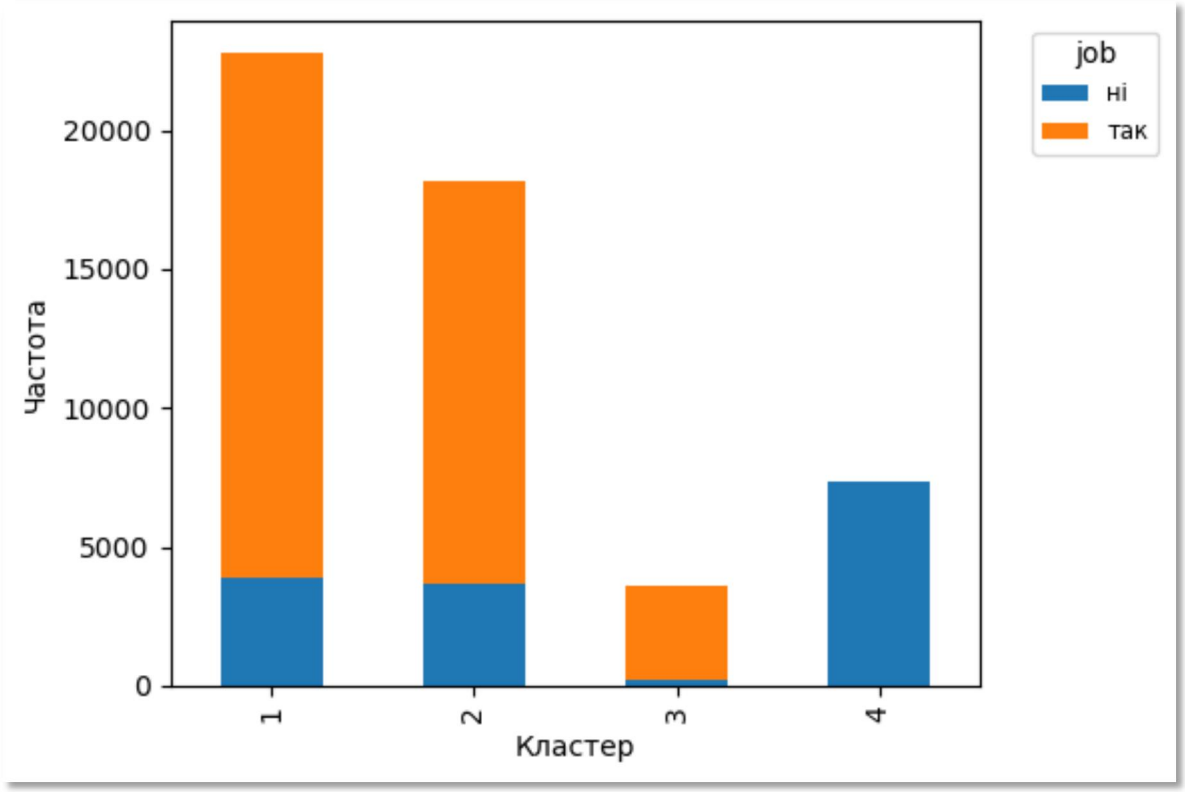

Рисунок 3.5.2 - Частотний розподіл ознаки `'job'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса

Перший, другий та третій кластери здебільшого складаються з громадян, які мають роботу.

Четвертий кластер повністю складається з безробітних громадян.

Важливо також зазначити, що зазвичай нумерація кластерів починається з 0. Відсутність цього кластера на графіку свідчить про те, що у цьому кластері немає громадян, які можуть працювати, тобто цей кластер повністю складається з дітей.

Перейдемо до наступного графіку, який зображатиме частотний розподіл ознаки `'job_category'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса.

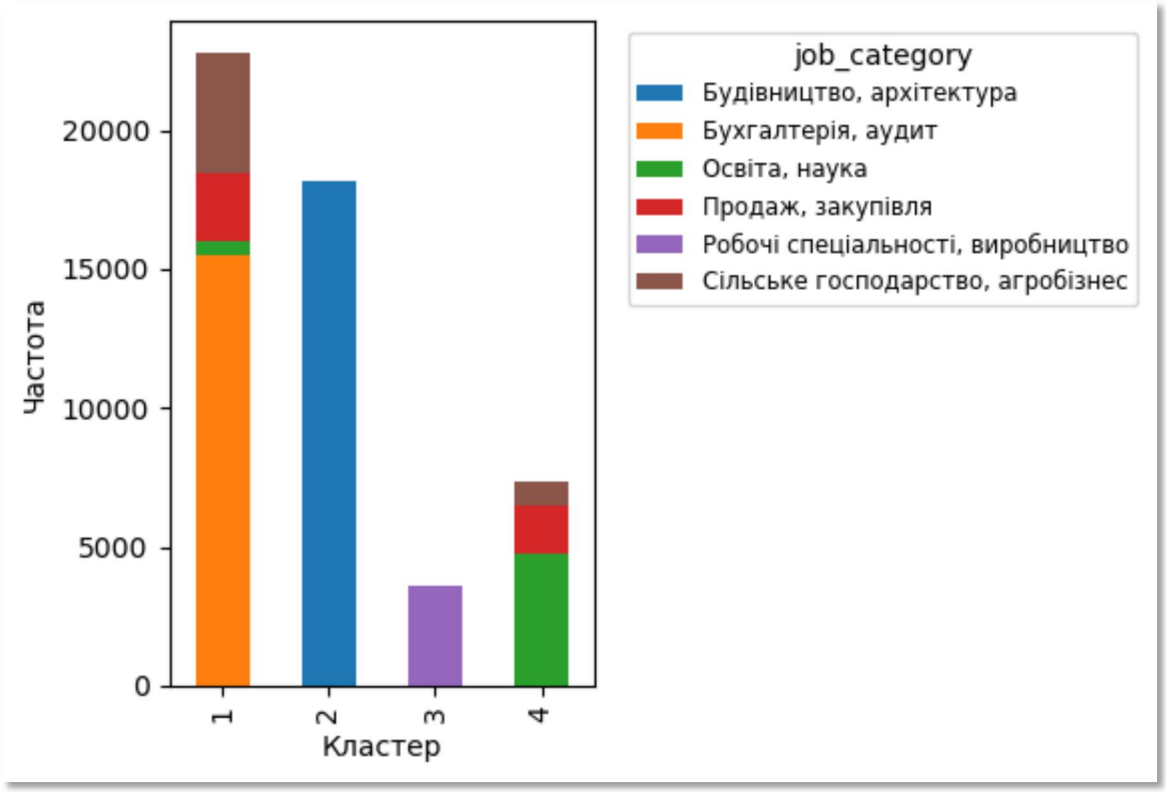

Рисунок 3.5.3 - Частотний розподіл ознаки `'job_category'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса

Перший кластер здебільшого складається з громадян, які працюють у сфері “Бухгалтерія, аудит”, другий — з працівників сфери “Будівництво, архітектура”, третій — з працівників сфери “Робочі спеціальності, виробництво”, четвертий — з громадян, які працюють у сфері “Освіта, наука”.

Як можна побачити, невелика частина працівників сфери “Освіта, наука” знаходиться у першому кластері, але більшість з них знаходиться у кластері, в якому ознака наявності роботи має значення “ні”.

Це свідчить про те, що більшість працівників сфери “Освіта, наука” в Одесі є безробітними на даний момент і потребують вжиття певних заходів для підвищення кількості робочих місць.

Далі подивимося на графік частотного розподілу ознаки `'location_reason'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса.

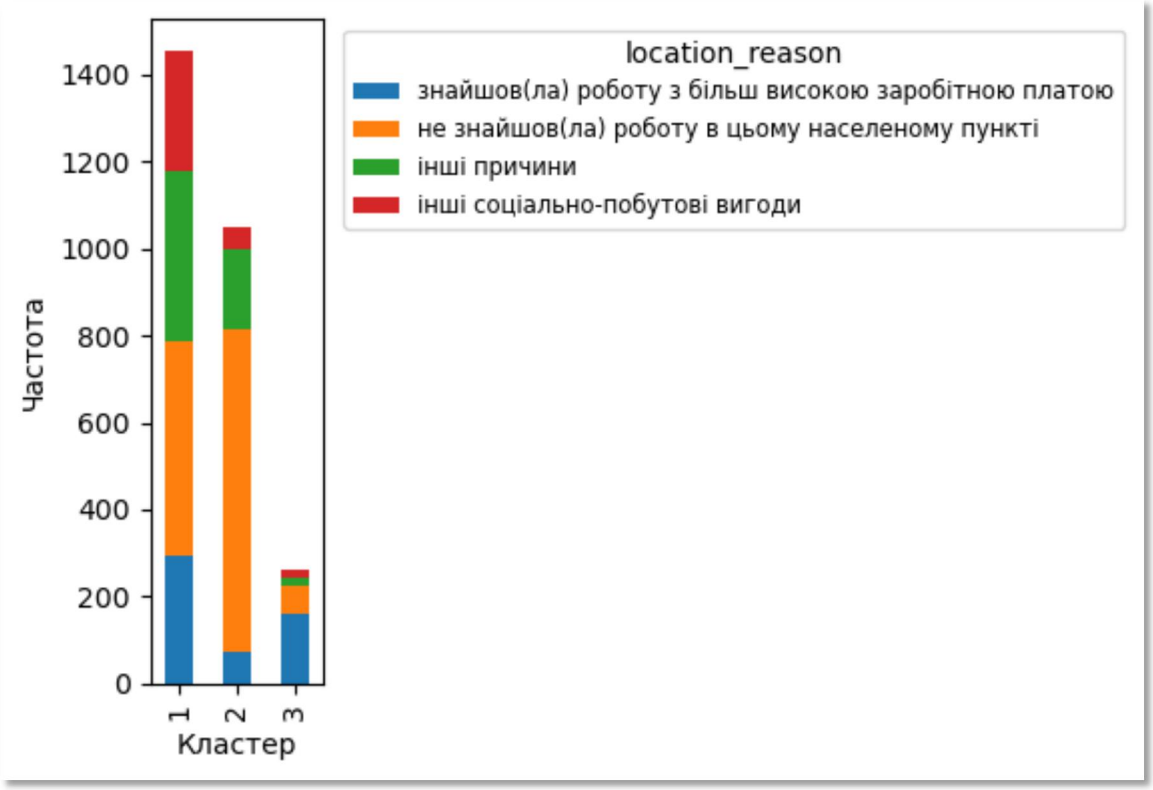

Рисунок 3.5.4 - Частотний розподіл ознаки `'location_reason'` підмножини “Відомості щодо наявності роботи для різних сфер діяльності” у кластерах міста Одеса

На даному графіку розподіл не такий чіткий, як у попередніх випадках, але все одно можна зробити певні висновки щодо причини, з якої громадяни, які мають роботу в іншому місті, не знайшли роботу в місті Одеса.

У другому кластері (який здебільшого складається з працівників сфери “Будівництво, архітектура”) основна причина роботи в іншому місті — “не знайшов(ла) роботу в цьому населеному пункті”.

У третьому кластері (який повністю складається з громадян, які працюють у сфері “Робочі спеціальності, виробництво”) основна причина, з якої громадяни не працюють у місті Одеса, — “знайшов(ла) роботу з більш високою заробітною платою”.

Перейдемо до наступного прикладу.

Розглянемо частотний розподіл ознаки 'education_level' підмножини "Відомості щодо освіти" у кластерах міста Київ.

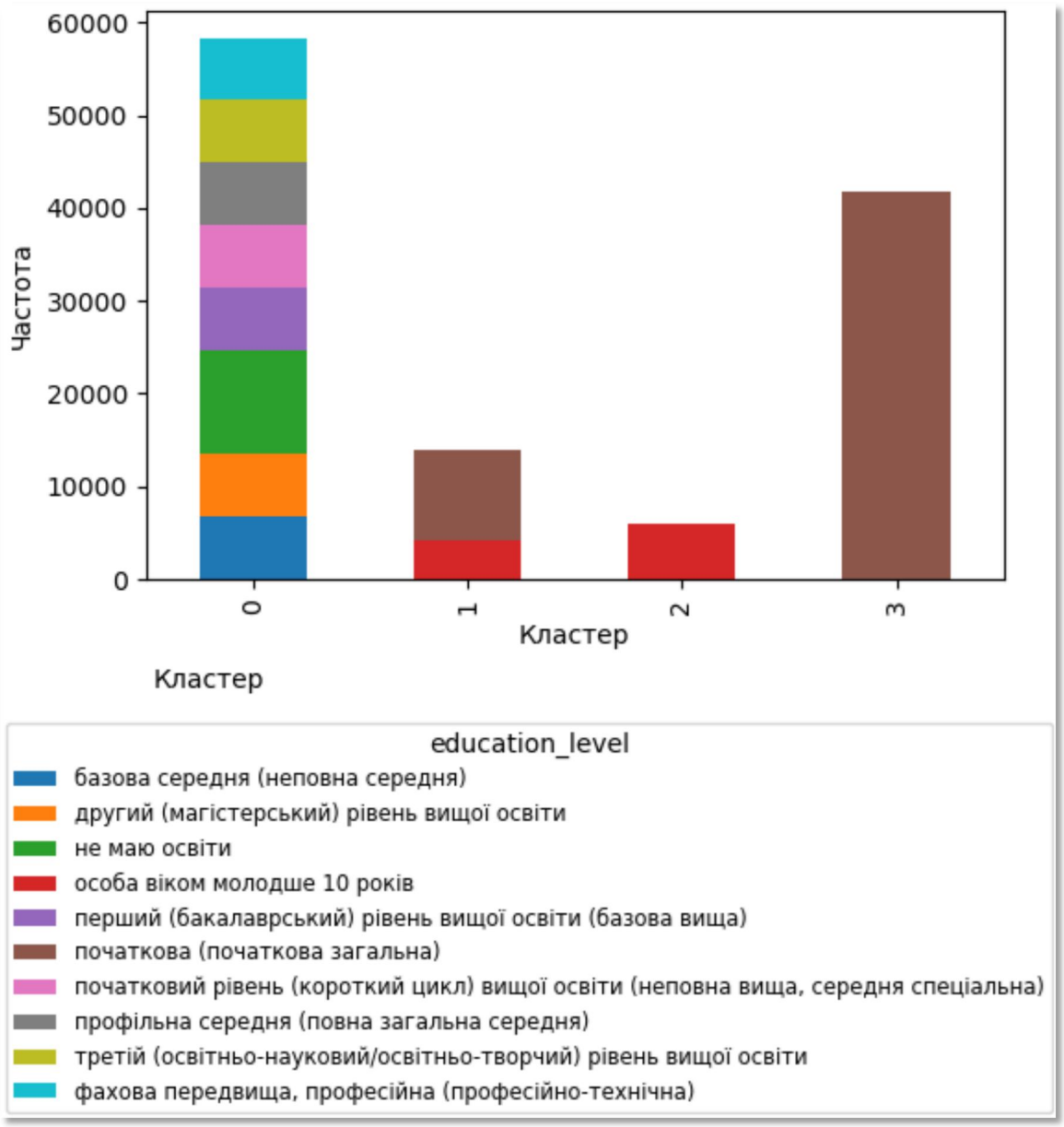

Рисунок 3.5.5 - Частотний розподіл ознаки 'education_level' підмножини "Відомості щодо освіти" у кластерах міста Київ

Кластер номер 0 можна охарактеризувати як кластер громадян, які здебільшого мають певний рівень освіти, однак частотний розподіл у даному випадку не є чітким.

Перший кластер здебільшого складається з громадян, які мають початкову загальну освіту, але має також частку осіб віком молодше 10 років.

Другий кластер повністю складається з осіб молодше 10 років.

Третій кластер повністю складається з громадян, які мають лише початкову загальну освіту. Це може свідчити про обмежений доступ до якісної освіти.

Ця частка громадян потребує вжиття певних заходів — забезпечення доступності середньої та професійної освіти через державні ініціативи, організації вечірніх шкіл або дистанційних курсів для дорослих, які з різних причин не завершили середню освіту.

Перейдемо до частотного розподілу ознаки `'education'` підмножини “Відомості щодо освіти” у кластерах міста Київ.

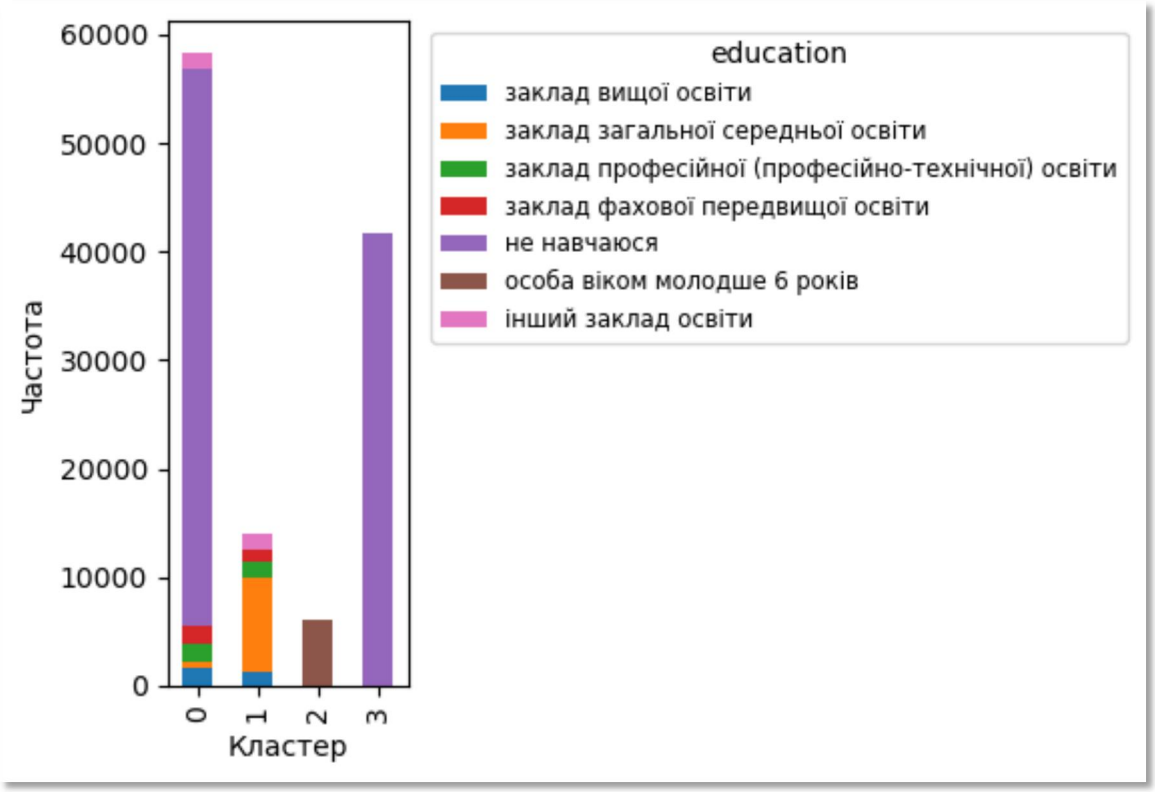

Рисунок 3.5.6 - Частотний розподіл ознаки `'education'` підмножини “Відомості щодо освіти” у кластерах міста Київ

Громадяни, які належать до до кластерів номер 0 та 3, на даний момент не навчаються у навчальному закладі.

Перший кластер складається з громадян, які на даний момент навчаються у закладах загальної середньої освіти. Враховуючи результати

частотного розподілу ознаки 'education_level', можна зробити висновок, що цей кластер складається з дітей та підлітків.

Другий кластер повністю складається з осіб віком молодше 6 років.

Нарешті, розглянемо останній частотний розподіл підмножини "Відомості щодо освіти" у кластерах міста Київ — частотний розподіл ознаки 'preschool'.

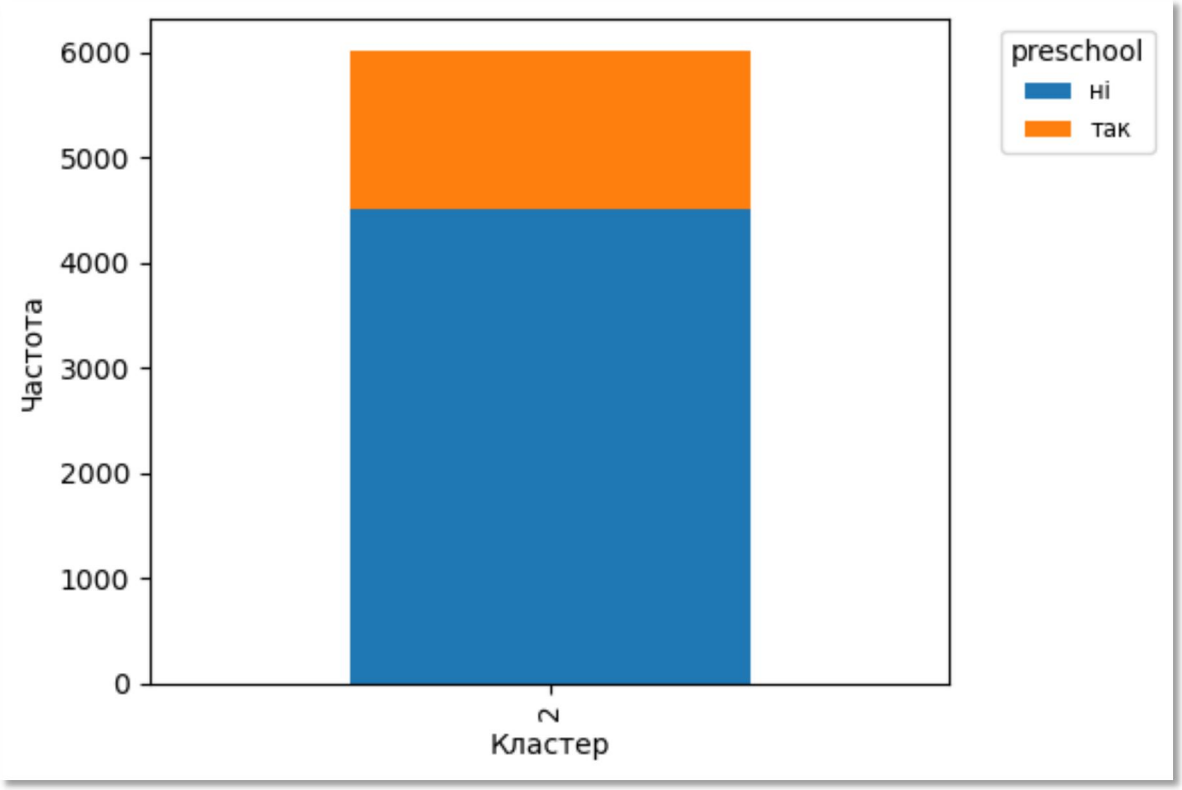

Рисунок 3.5.7 - Частотний розподіл ознаки 'preschool' підмножини "Відомості щодо освіти" у кластерах міста Київ

На графіку можна побачити частотний розподіл лише одного кластеру, що свідчить про те, що всі інші кластери не містять осіб, які можуть ходити до дошкільного навчального закладу.

Другий кластер здебільшого складається з дітей, які не ходять до дошкільного навчального закладу. Це може свідчити про те, що у місті недостатньо місць для дітей у дошкільних навчальних закладах.

4 ЕТАП ПРОГНОЗУВАННЯ

4.1 Косинусна подібність

Косинусна подібність — це математичний метод визначення схожості між двома наборами даних, заснований на вимірюванні кута між їхніми векторами. Вона дає змогу оцінити, наскільки два об’єкти даних схожі, незалежно від їх розміру чи масштабу, що дозволяє не занурюватися в конкретні значення окремих точок даних.

Ця метрика широко застосовується в таких сферах, як обробка природної мови, пошукові алгоритми та системи рекомендацій. Косинусна подібність допомагає визначити семантичну схожість між текстами, наборами даних або навіть зображеннями.

Особливістю косинусної подібності є те, що вона ґрунтується на порівнянні напрямків векторів у багатовимірному просторі, а не їхніх величин.

Процес обчислення косинусної подібності включає:

1) Скалярний добуток двох векторів, який показує, наскільки вектори вирівняні в одному напрямку.
2) Обчислення величин (довжин) кожного вектора для нормалізації.
3) Поділ скалярного добутку на добуток величин векторів, щоб отримати значення косинусної подібності.

Результат 1 означає повну подібність (вектори спрямовані в одному напрямку).

Результат 0 вказує на відсутність подібності (вектори перпендикулярні).

Результат -1 свідчить про протилежність (вектори спрямовані у протилежні боки).

Завдяки своїй здатності фокусуватися на напрямку векторів, косинусна подібність часто використовується для кластеризації документів, побудови систем рекомендацій та пошукових алгоритмів, де важливо розуміти подібність між текстами чи іншими об'єктами. [14]

4.2 Пошук історичних кластерів, найбільш схожих на поточні

Перед тим, як застосовувати лінійну регресію до даних, необхідно визначитися, на основі яких кластерів буде відбуватися прогнозування, оскільки в історичних даних після кластеризації будуть сформовані кластери, які відрізняються від кластерів у поточних даних.

Задача цього етапу — знайти історичні кластери, які найбільше схожі на поточні.

Для цього створюємо **`Notebook`** документ веб-інтерактивного обчислювального середовища **`Jupyter Notebook`**, який назвемо **`CensusAnalisys`** (див. Додаток Д).

У циклі `for()` відбувається прохід по всім містам і підмножинам поточних даних.

Частотні таблиці, які були отримані під час етапу візуалізації частотного розподілу окремих ознак у кластерах, порівнюються за допомогою функції `compare_clusters_by_cosine`, в якій відбувається обчислення косинусної подібності рядків частотних таблиць.

Порівняння відбувається між поточною частотною таблицею та всіма історичними частотними таблицями, в результаті чого визначаються історичні кластери, які найбільш схожі на поточні кластери.

Історичні кластери, найбільш схожі на поточні кластери, формують своєрідний “ланцюг” починаючи з поточних кластерів і закінчуючи найдавнішими історичними кластерами.

Подивимося, які історичні кластери виявилися найбільш схожими на кластери у місті Ізмаїл у підмножині “Відомості щодо етнічного походження”:

```
Місто: Ізмаїл
  Підмножина: Відомості щодо етнічного походження
    Кластер 2024: 0
      2014: 0
      2004: 1
      1994: 0

    Кластер 2024: 1
      2014: 0
      2004: 0
      1994: 0

    Кластер 2024: 2
      2014: 1
      2004: 2
      1994: 2
```

Рисунок 4.2.1 - “Ланцюг” кластерів, найбільш схожих на поточні кластери підмножини “Відомості щодо етнічного походження” міста Ізмаїл

Для більш наочної демонстрації результатів пошуку історичних кластерів, найбільш схожих на поточні, створимо функцію `plot_cluster_chain` для побудови графіка ланцюга кластерів.

Графіки, які зображують результати, зберігаються в окремих зображеннях формату .png.

Подивимося на приклад графіку, який був сформований для кластерів міста Дубляни підмножини “Відомості щодо статусу інвалідності”:

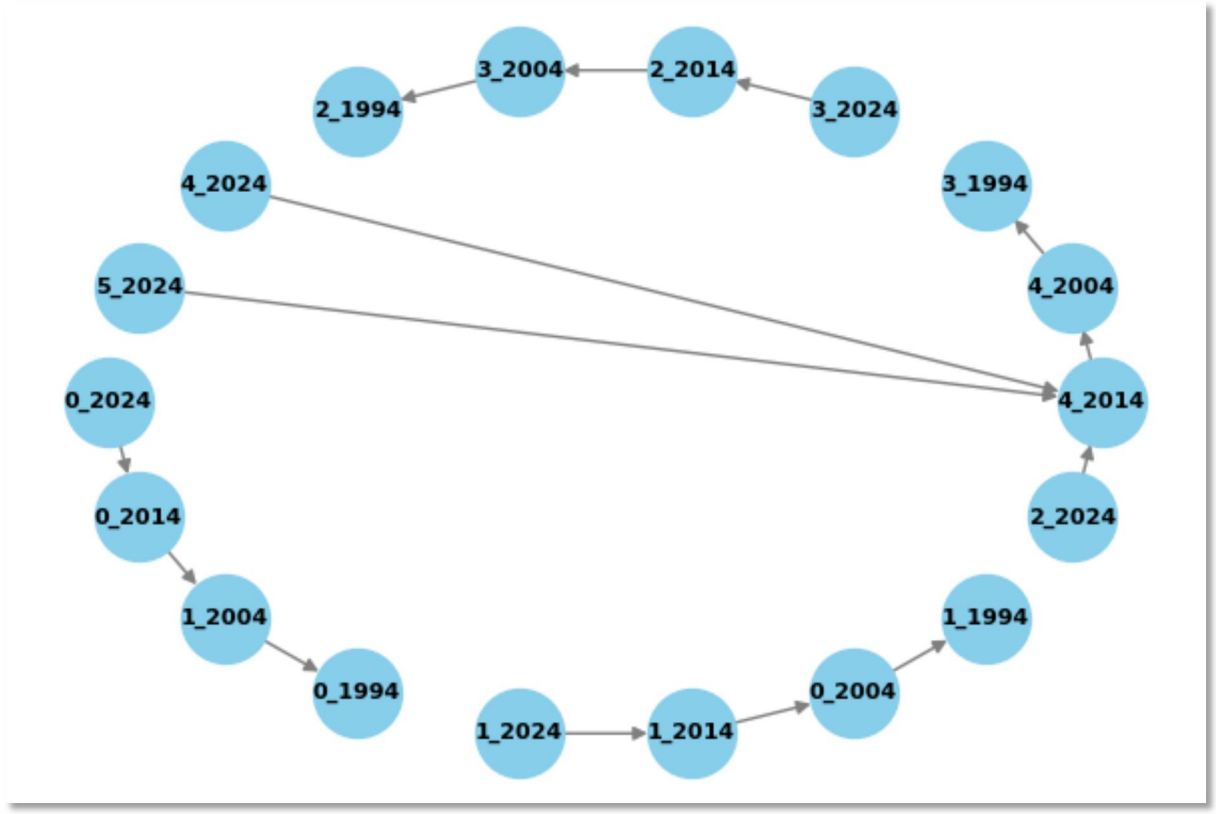

Рисунок 4.2.2 - Графік “ланцюга” кластерів, найбільш схожих на поточні кластери підмножини “Відомості щодо статусу інвалідності” міста Дубляни

Як можна побачити, у даному випадку “ланцюжки” мають здебільшого просту структуру, однак другий, четвертий та п’ятий поточні кластери найбільш схожі на четвертий кластер 2014 року. Таким чином, історичний кластер номер 4 був розділений на три кластери у 2024 році.

4.3 Застосування прогнозування до кластерів

Для прогнозування використовується модель лінійної регресії (див. Додаток Д).

У циклі `for()` проходимося по всім містам та підмножинам, отримуємо частотні таблиці для кожного року і на їхній основі прогнозуємо значення для 2034 року.

Подивимося на прогноз для ознаки 'ethnic_origin' у кластері номер 0 підмножини "Відомості щодо етнічного походження" у місті Ізмаїл:

```
Місто: Ізмаїл

  Підмножина: Відомості щодо етнічного походження

    Кластер 2024: 0

      Колонка: ethnic_origin
        білорус (-ка) => Прогноз на 2034 рік: 302
        кримський татарин (-ка) => Прогноз на 2034 рік: 334
        молдаванин (-ка) => Прогноз на 2034 рік: 215
        поляк (-ка) => Прогноз на 2034 рік: 296
        росіянин (-ка) => Прогноз на 2034 рік: 868
        румун (-ка) => Прогноз на 2034 рік: 1070
        українець (-ка) => Прогноз на 2034 рік: 5837
```

Рисунок 4.3.1 - Результати прогнозування на 2034 рік ознаки 'ethnic_origin' у кластері номер 0 підмножини "Відомості щодо етнічного походження" у місті Ізмаїл

Порівняємо ці результати з поточними даними:

```
Місто: Ізмаїл
  Підмножина: Відомості щодо етнічного походження
    Характеристика: ethnic_origin
ethnic_origin   білорус    кримський татарин      молдаванин
cluster
0                  352                  350            1680
ethnic_origin    поляк     росіянин      румун     українець
cluster
0                  302         1130       1643          6829
```

Рисунок 4.3.2 - Поточні дані для ознаки 'ethnic_origin' у кластері номер 0 підмножини "Відомості щодо етнічного походження" у місті Ізмаїл

Як можна помітити, більшість прогнозованих значень схожі на поточні, окрім значень для громадян молдавського походження. Це може свідчити про можливу зміну структури даного кластера у майбутньому.

5 ФОРМУВАННЯ ЗВІТІВ

Після завершення етапів кластеризації та прогнозування можна переходити до формування звітів щодо добробуту та потреб кожного міста.

Спочатку у кожному кластері знаходяться ознаки, які є характерними для них (ознаки, які зустрічаються найчастіше) (див. Додаток Д).

Функція `is_problematic` перевіряє, чи є кластери проблемними за певними характеристиками для поточних та прогнозованих ознак, найбільш характерних для цих кластерів.

Наприклад, значення рівня освіти “не маю освіти” є фактором, що відносить кластер до категорії проблемних кластерів. Це може свідчити про те, що у місті обмежений доступ до безкоштовної або доступної базової освіти для дітей та дорослих.

Далі у функції `describe_problematic_clusters` на основі кожного проблемного кластера міста відбувається формування звітів щодо добробуту кожного міста, потребами різних соціальних груп громадян та пропозиціями щодо шляхів задоволення цих потреб.

Наприклад, якщо більшість громадян, які відносяться до соціальної групи, яка має проблеми із ходінням, пропонується облаштування пандусів, ліфтів, спеціальних доріжок та зручних тротуарів для людей з обмеженими можливостями

Приклад цього звіту для міста Одеса наведений у додатку Е.

6 ОЦІНКА ЯКОСТІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

6.1 Метрики оцінки якості кластеризації

Метрики оцінки якості кластеризації дозволяють визначити, наскільки вдало обраний алгоритм кластеризації розділив дані на кластери. Вони допомагають оцінити компактність (елементи в кластері мають бути близькими один до одного) та відокремленість (кластери мають бути добре відділені один від одного).

Основні метрики для оцінки кластеризації:

1) **`Silhouette Score`** (Коефіцієнт силуету)

Ця метрика оцінює якість кластеризації шляхом вимірювання, наскільки схожий кожен об'єкт на об'єкти свого кластеру, порівняно з об'єктами сусіднього кластеру.

Формула:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

де:

$a(i)$ — середня відстань від точки i до інших точок того ж кластеру;

$b(i)$ — середня відстань від точки i до точок найближчого сусіднього кластеру.

Значення близькі до 1 означають, що точка добре вписується у свій кластер.

Значення близькі до 0 означають, що точка знаходиться між двома кластерами.

Значення < 0 свідчать про помилкову кластеризацію.

2) **`Davies-Bouldin Index`** (Індекс Девіса-Боулдіна)

Ця метрика вимірює середню «схожість» між кластерами, де схожість визначається як відношення відстані між кластерами до їхніх розмірів.

Формула:

$$DB = \frac{1}{N}\sum_{i=1}^{N} \max_{j \neq i}\left(\frac{\sigma_i + \sigma_j}{d(i,j)}\right)$$

де:

σ — середня відстань точок у кластері до його центроїду;

$d(i,j)$ — відстань між центроїдами кластерів i та j;

N — кількість кластерів.

Чим менше значення DB, тим краще. Низьке значення означає, що кластери компактні й добре відокремлені.

3) **`Calinski-Harabasz Index`** (Індекс Калінські-Харабаш)

Цей індекс оцінює співвідношення між дисперсією між кластерами та всередині кластерів.

Дисперсія — це статистична міра, яка показує, наскільки значення даних відхиляються від їхнього середнього значення (середнього арифметичного). Вона використовується для оцінки розсіювання або варіативності даних.

Формула:

$$CH = \frac{tr(B_k)/(k-1)}{tr(W_k)/(n-k)}$$

де:

$tr(B_k)$ — дисперсія між кластерами;

$tr(W_k)$ — внутрішньо-кластерна дисперсія;

k — кількість кластерів;

n — загальна кількість точок.

Чим більше значення CH, тим краще.

Високий індекс CH свідчить про добре відокремлені та компактні кластери. [15]

6.2 Оцінка якості кластеризації

Обчислимо три різні метрики для оцінки якості кластеризації: **Silhouette Score**, **Davies-Bouldin Index** та **Calinski-Harabasz Index**.

Для цього створимо новий **Notebook** документ веб-інтерактивного обчислювального середовища **Jupyter Notebook**, який назвемо **QualityAssessment** (див. Додаток Г).

`davies_bouldin_score` — функція для обчислення **Davies-Bouldin Index**. Менші значення даного індексу вказують на кращу кластеризацію, а більші — на гіршу якість кластеризації.

`calinski_harabasz_score` — функція для обчислення **Calinski-Harabasz Index**. Більші значення цього індексу вказують на кращу кластеризацію.

`silhouette_score` — функція для обчислення **Silhouette Score**. Значення близьке до 1 вказує на хорошу кластеризацію, 0 — на нечіткі межі кластерів, а негативне значення — на погану якість кластеризації.

Перевіримо значення цих метрик для кластерів та підмножин декількох різних міст.

Розглянемо якість кластеризації у Києві за підмножиною “Відомості щодо наявності роботи для різних сфер діяльності”:

```
Silhouette Score: 0.8408959841537827
Davies-Bouldin Index: 0.4005789349567444
Calinski-Harabasz Index: 103170.02370707109
```

Рисунок 6.2.1 – Оцінка якості кластеризації у Києві за підмножиною “Відомості щодо наявності роботи для різних сфер діяльності”

Результат **Silhouette Score** 0.8408959841537827 — дуже висока якість кластеризації. Це означає, що обраний алгоритм успішно згрупував дані.

Результат **Davies-Bouldin Index** 0.4005789349567444 — дуже низьке значення, що підтверджує гарну кластеризацію. Кластери мають чітке відокремлення і мінімальне перекриття.

Результат **Calinski-Harabasz Index** 103170.02370707109 — дуже високе значення, яке свідчить про значний рівень відділення кластерів і їхню компактність.

Далі розглянемо якість кластеризації у Львові за підмножиною “Відомості щодо місцезнаходження основної роботи”:

```
Silhouette Score: 0.6948658566878703
Davies-Bouldin Index: 0.7042492469927847
Calinski-Harabasz Index: 54646.17719842194
```

Рисунок 6.2.2 – Оцінка якості кластеризації у Львові за підмножиною “Відомості щодо місцезнаходження основної роботи”

Результат **Silhouette Score** 0.6948658566878703 — це хороший результат, який свідчить про те, що більшість точок правильно віднесені до своїх кластерів. Проте є деякі точки, які розташовані близько до меж кластерів.

Результат **Davies-Bouldin Index** 0.7042492469927847 — це добрий показник, який підтверджує, що кластери чітко розділені, і між ними мінімальне перекриття.

Результат **Calinski-Harabasz Index** 54646.17719842194 — значення досить високе, що означає, що кластери добре розділені, а точки в межах кластерів компактні.

У всіх випадках дані метрики вказують на високу якість кластеризації: кластери чітко відокремлені один від одного, точки в межах кожного кластера сильно згруповані.

Відповідно, алгоритм кластеризації добре справився із завданням.

Ці результати є ознакою того, що дані добре підходять до обраного алгоритму кластеризації, а число кластерів було вибране коректно.

6.3 Метрики оцінки якості прогнозування

Щоб оцінити якість прогнозування за наявності фактичних та прогнозованих даних, можна застосувати кілька підходів.

Середня абсолютна помилка (**`Mean Absolute Error`**) вимірює середнє значення абсолютних помилок між реальними та прогнозованими значеннями даних.

Формула:

$$\boldsymbol{MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y^i - \widehat{Y}\right|}$$

де:

$\boldsymbol{n}$ — загальна кількість спостережень (реальних $\boldsymbol{i}$ прогнозованих пар).

$\boldsymbol{Y^i}$ — реальне значення для $\boldsymbol{i}$-того спостереження.

$\boldsymbol{\widehat{Y}}$ — прогнозоване значення для $\boldsymbol{i}$-того спостереження.

Якщо реальні та прогнозовані дані дуже схожі, то **`MAE`** буде маленьким. Якщо ж прогнози сильно відрізняються від фактичних даних, **`MAE`** буде великим.

Середня квадратична помилка (**`Mean Squared Error`**) вимірює середнє значення квадратів різниць між реальними та прогнозованими значеннями у наборі даних.

Формула:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}\left|Y^i - \widehat{Y}\right|^2}$$

де:

$\boldsymbol{n}$ — загальна кількість спостережень (реальних $\boldsymbol{i}$ прогнозованих пар).

Y^i — реальне значення для i-того спостереження.

$\hat{Y}$ — прогнозоване значення для i-того спостереження.

MSE дає більше значення великим помилкам. Якщо присутні прогнози з великими відхиленнями від фактичних значень, **MSE** буде більш чутливим до цих помилок.

Чим менше значення **MSE,** тим краще модель.

Корінь середньої квадратичної помилки (**Root Mean Squared Error**) є квадратним коренем з **MSE**.

Він надає більш інтуїтивно зрозуміле значення в тій самій одиниці вимірювання, що й дані. Це дозволяє більш чітко оцінити середнє відхилення прогнозованих значень. [16]

6.4 Оцінка якості прогнозування

Оскільки для прогнозованих даних немає фактичних даних, з якими можна було б їх порівняти, спрогнозуємо дані для 2024 року на основі історичних даних (див. Додаток Д).

Спрогнозовані та фактичні дані знаходяться у різних форматах (спрогнозовані мають вигляд вкладених словників, а фактичні — вигляд датафреймів), тому приведемо їх до однакового вигляду (`freq_dict[city][subset_name][column][cluster][feature]`).

Повторимо серед спрогнозованих кластерів процедуру пошуку тих кластерів, які найбільш схожі на фактичні кластери 2024 року.

Далі візьмемо для прикладу фактичні значення 2024 року та значення, які були спрогнозовані для 2024 року, для одного кластеру підмножини “Відомості щодо освіти” міста Львова:

```
[ 1534  1501 46439  1485]
[ 1451  1435 44694  1434]
```

Рисунок 6.4.1 – Фактичні та прогнозовані дані у Львові за підмножиною “Відомості щодо освіти” (кластер номер 0)

Як можна побачити, значення дещо відрізняються, але найбільшою перешкодою в обчисленні розглянутих вище метрик є третє значення, яке є значно більшим за інші значення у кластері (і в випадку історичних, і фактичних даних).

Такі великі значення суттєво впливають на всі метрики помилки.

MAE враховує абсолютну різницю, тому велика помилка (у тисячах) у значенні 46439 домінує над іншими.

MSE і **RMSE** ще чутливіші, оскільки вони зводять різниці до квадрату, що значно збільшує вплив великих відхилень.

Коли дані не приведені до спільного масштабу, метрики, які працюють із вихідними значеннями (як **MAE, MSE**), сильно залежать від масштабу даних.

Приведення всіх значень до одного масштабу (наприклад, нормалізація) зробить можливим порівняння помилок між різними величинами.

Нормалізація полягає в тому, щоб перетворити всі значення на шкалу від 0 до 1, де найбільше значення стає рівним 1.

Приклад нормалізації:

$$\boldsymbol{Y_{real\_norm}} = \frac{\boldsymbol{Y_{real}}}{\boldsymbol{max(Y_{real})}}, \boldsymbol{Y_{pred\_norm}} = \frac{\boldsymbol{Y_{pred}}}{\boldsymbol{max(Y_{real})}}$$

де:

$\boldsymbol{Y_{real}}$ — це реальні значення.

$\boldsymbol{Y_{pred}}$ — це прогнозовані значення.

$\boldsymbol{max(Y_{real})}$ — максимальне значення серед реальних даних.

$\boldsymbol{Y_{real\_norm}}$ — нормалізовані реальні значення.

$\boldsymbol{Y_{pred\_norm}}$ — нормалізовані прогнозовані значення.

Після нормалізації фактичних та прогнозованих даних перейдемо до застосування метрик оцінювання якості до нормалізованих даних.

Для **MAE** використовується функція `mean_absolute_error` з бібліотеки `sklearn.metrics`.

MSE обчислюється через функцію `mean_squared_error`, яка також є частиною `sklearn.metrics`.

RMSE — корінь з **MSE**, для обчислення цього ми використовуємо `np.sqrt()`.

Подивимося на результати, отримані після застосування метрик **MAE**, **MSE** та **RMSE**:

```
Нормалізована середня абсолютна помилка (MAE):
0.010470725037145514
Нормалізована середня квадратична помилка (MSE):
0.0003545973227681479
Корінь нормалізованої середньої квадратичної помилки (RMSE):
0.01883075470521954
```

Рисунок 6.4.2 – Нормалізовані результати **MAE**, **MSE** та **RMSE**

Після нормалізації даних результати метрик (**MAE**, **MSE**, **RMSE**) потрібно правильно інтерпретувати, оскільки вони обчислюються на масштабованих значеннях.

Оскільки нам потрібні помилки в оригінальному масштабі даних, потрібно виконати "денормалізацію" результатів.

Якщо дані були нормалізовані шляхом ділення на максимальне значення, денормалізоване значення **MAE** буде:

$$MAE_{orig} = MAE_{norm} \times Y_{max}$$

де:

MAE_{orig} — результати метрики **MAE**.

MAE_{norm} — нормалізовані результати метрики **MAE**.

Y_{max} — максимальне значення серед реальних даних.

Денормалізація **RMSE** виконується аналогічно до **MAE**:

$$RMSE_{orig} = RMSE_{norm} \times Y_{max}$$

де:

$RMSE_{orig}$ — результати метрики **RMSE**.

$RMSE_{norm}$ — нормалізовані результати метрики **RMSE**.

Y_{max} — максимальне значення серед реальних даних.

Оскільки помилки зводяться до квадрату, денормалізація **MSE** потребує врахування квадрата масштабу:

$$MSE_{orig} = MSE_{norm} \times Y_{max}^2$$

де:

MSE_{orig} — результати метрики **MSE.**

MSE_{norm} — нормалізовані результати метрики **MSE.**

Y_{max} — максимальне значення серед реальних даних.

Провівши денормалізацію, отримаємо такі результати:

```
Денормалізована MAE: 486.2500000000005
Денормалізована MSE: 764717.750000002
Денормалізована RMSE: 874.4814177556902
```

Рисунок 6.4.3 – Денормалізовані результати **MAE, MSE** та **RMSE**

MAE (Середня абсолютна помилка) 486.25 означає, що у середньому, прогноз відхиляється від реальних значень на 486.25.

Враховуючи масштаб даних (найбільше значення — 46439), це менше 1% від найбільшого значення, що є непоганим результатом.

Але якщо порівняти з меншими значеннями (1534, 1501 тощо), помилка виглядає великою. Це підтверджує, що велике значення домінує над іншими в обчисленні помилок.

MSE (Середня квадратична помилка) 764717.75 — це значення виглядає великим, але це очікувано через чутливість MSE до великих значень (різниця в 1745 між 46439 і 44694 значно збільшує цей показник).

RMSE (Корінь середньої квадратичної помилки): 874.48 — у середньому, відхилення прогнозу від реальних даних становить близько 874.48, приблизно 1.9% від найбільшого значення (46439). Даний результат є допустимим для прогнозування даних з подібними масштабами значень.

Малий відсоток помилки для великих значень вказує на те, що модель добре прогнозує ті дані, де значення є високими (наприклад, 46439).

Для менших значень прогноз менш точний, оскільки помилка виглядає значною в абсолютних значеннях (486 для 1500 — це приблизно 30% помилки). Однак цей результат прийнятний, оскільки діапазон значень є широким.

Незважаючи на велику загальну помилку, можна зробити висновок, що у загальному модель не дуже сильно помиляється.

Однак великий діапазон значень може вимагати додаткового коригування або переробки моделі для підвищення точності, особливо для малих значень.

ВИСНОВКИ

Під час виконання магістерського дослідження визначено проблематику предметної області добробуту та потреб громадян віртуальної країни та сформульовано задачі для її вирішення.

Для вирішення поставлених задач та реалізації інформаційної технології аналізу добробуту та потреб громадян віртуальної країни з метою прийняття відповідних рішень влади обрано методи машинного навчання, зокрема метод кластеризації K-modes та лінійну регресію.

Дана інформаційна технологія реалізує усі поставлені задачі:

1) аналіз даних для виявлення основних соціальних груп, які мають схожі потреби;
2) прогнозування майбутніх потреб громадян на основі поточних даних про громадян;
3) графічне представлення результатів для зручного сприйняття управлінцями;
4) формування звітів з аналізом добробуту та потреб громадян для кожного міста віртуальної країни.

Якість інформаційної технології оцінювалася за кількома критеріями, серед яких компактність та відокремленість кластерів різних соціальних груп громадян та середнє відхилення прогнозованих значень від фактичних даних.

Обрані метрики оцінки якості кластеризації вказують на високу якість кластеризації: кластери чітко відокремлені один від одного, точки в межах кожного кластера сильно згруповані.

Малий відсоток помилки прогнозування для великих значень вказує на те, що модель добре прогнозує ті дані, де значення є високими, однак великий діапазон значень може вимагати додаткового коригування або переробки моделі для підвищення точності, особливо для малих значень.

Порівняно з ручною обробкою даних, інформаційна технологія надає можливість проводити аналіз без впливу людського фактору (схильність до помилок, суб’єктивність, повільність процесу, обмеження в масштабуванні).

Дана інформаційна технологія має такі перспективи подальшого розвитку:

1) можливість створення окремих моделей для кожного кластера замість глобального прогнозування для всіх кластерів. Це дозволить враховувати унікальні характеристики кожного з них;

2) інтеграція зовнішніх факторів, які впливають на кластеризацію або прогнози (наприклад, економічні зміни, політичні впливи).

Для більш правильної оцінки якості інформаційної технології потрібно зв’язатися та протестувати технологію з органами місцевої влади.

За результатами даного дослідження зроблено доповідь та опубліковано тези на двадцять першій всеукраїнській конференції студентів і молодих науковців “Інформатика, інформаційні системи та технології”, яка проводилася факультетом математики, фізики та інформаційних технологій Одеського національного університету імені І. І. Мечникова та фізико-математичним факультетом державного закладу “Південноукраїнський національний педагогічний університет імені К. Д. Ушинського” 26 квітня 2024 року [17], опубліковано тези на XXIV Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів “Стан, досягнення та перспективи інформаційних систем і технологій”, яка проводилася Одеським національним технологічним університетом, Університетом Інформатики і прикладних знань, м. Лодзь, Польща та Інститутом комп`ютерної інженерії, автоматизації, робототехніки та програмування ім. П. Н. Платонова 18-19 квітня 2024 року. [18]

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційна сторінка Всеукраїнського перепису населення [Електронний ресурс]. – Режим доступу: https://ukrcensus.gov.ua/

2. Comparing Clustering vs Classification: When to Use Each [Електронний ресурс] - Режим доступу: https://machinelearningmodels.org/comparing-clustering-vs-classification-when-to-use-each/

3. K-Mode Clustering in Python - GeeksforGeeks [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/k-mode-clustering-in-python/

4. ML | K-Medoids clustering with solved example - GeeksforGeeks [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/

5. Hierarchical Clustering in Machine Learning - GeeksforGeeks [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/hierarchical-clustering/

6. How to Choose among Three Forecasting Models: Machine Learning, Statistical and Expert | Bain & Company [Електронний ресурс] - Режим доступу: https://www.bain.com/insights/how-to-choose-among-three-forecasting-models/

7. 10 Machine Learning Algorithms to Know in 2025 | Coursera [Електронний ресурс] - Режим доступу: https://www.coursera.org/articles/machine-learning-algorithms

8. Наказ Держстату від 05 січня 2022 року № 2 "Про затвердження Програми Всеукраїнського перепису населення 2023 року"

9. PostgreSQL: About [Електронний ресурс] - Режим доступу: https://www.postgresql.org/about/#:~:text=PostgreSQL%20has%20earned%20a%20strong%20reputation%20for%20its,software%20to%20consistently%20deliver%20performant%20and%20innovative%20solutions.

10. C# Programming: What It Is, How It's Used + How to Learn It | Coursera [Електронний ресурс] - Режим доступу: https://www.coursera.org/articles/c-sharp?msockid=1e4174ed09e0680f08da602108ef69cd

11. Elbow Method in Python for K-Means and K-Modes Clustering - Coding Infinite [Електронний ресурс] - Режим доступу: https://codinginfinite.com/elbow-method-in-python-for-k-means-and-k-modes-clustering/
12. How to implement clustering algorithms in Python: A Step-by-Step Approach [Електронний ресурс] - Режим доступу: https://dataheadhunters.com/academy/how-to-implement-clustering-algorithms-in-python-a-step-by-step-approach/
13. Frequency Distribution in Statistics - Table, Graphs, Formula and Examples [Електронний ресурс] - Режим доступу: https://www.geeksforgeeks.org/frequency-distribution/
14. What is Cosine Similarity? A Comprehensive Guide | DataStax [Електронний ресурс]. – Режим доступу: https://www.datastax.com/guides/what-is-cosine-similarity
15. How to measure clustering performances when there are no ground truth? | by Haitian Wei | Medium [Електронний ресурс]. – Режим доступу: https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c
16. MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? | by Akshita Chugh | Analytics Vidhya | Medium [Електронний ресурс]. – Режим доступу: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e
17. Нуждіна М.І. Інформаційна технологія підтримки прийняття рішень з розвитку інфраструктури віртуальної країни. / Нуждіна М.І., Царенко О.П. // Тези доповідей двадцять першої всеукраїнської конференції студентів і молодих науковців “Інформатика, інформаційні системи та технології”. Одеса, ОНУ імені І.І.Мечникова, ПНПУ імені К.Д.Ушинського, 26 квітня 2024 р. – Одеса, 2024, С. 31-32

18. Нуждіна М.І. Інформаційна технологія підтримки прийняття рішень з розвитку інфраструктури віртуальної країни. / Нуждіна М.І., Царенко О.П. // Тези доповідей XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів “Стан, досягнення та перспективи інформаційних систем і технологій”. Одеса, Одеський національний технологічний університет, Університет Інформатики і прикладних знань, м.Лодзь, Польща, Інститут комп`ютерної інженерії, автоматизації, робототехніки та програмування ім.П.Н.Платонова, 18-19 квітня 2024 р. – Одеса, 2024, С. 186-188

ДОДАТОК А.1

Запити на створення доменів бази даних

```
create domain gender_d as varchar
check(value in ('Чоловіча', 'Жіноча'));

create domain marital_status_d as varchar
check(value in ('ніколи не перебував(ла) у шлюбі', 'перебуваю в зареєстрованому шлюбі', 'удівець', 'удова', 'перебуваю в незареєстрованому шлюбі', 'розлучений(а) (зареєстроване розірвання шлюбу)', 'розійшовся(лася) (незареєстроване розірвання шлюбу)', 'особа віком молодше 15 років'));

create domain citizenship_d as varchar
check(value in ('громадянство України', 'громадянство іншої держави', 'особа без громадянства', 'громадянство не визначене'));

create domain education_level_d as varchar
check(value in ('третій (освітньо-науковий/освітньо-творчий) рівень вищої освіти', 'другий (магістерський) рівень вищої освіти', 'початкова (початкова загальна)', 'перший (бакалаврський) рівень вищої освіти (базова вища)', 'початковий рівень (короткий цикл) вищої освіти (неповна вища, середня спеціальна)', 'фахова передвища, професійна (професійно-технічна)', 'базова середня (неповна середня)', 'профільна середня (повна загальна середня)', 'не маю освіти', 'особа віком молодше 10 років'));

create domain education_d as varchar
check(value in ('заклад вищої освіти', 'заклад фахової передвищої освіти', 'інший заклад освіти', 'заклад професійної (професійно-технічної) освіти', 'заклад загальної середньої освіти', 'особа віком молодше 6 років', 'не навчаюся'));

create domain yes_no as varchar
check(value in ('так', 'ні'));

create domain location_reason_d as varchar
check(value in ('не знайшов(ла) роботу в цьому населеному пункті', 'знайшов(ла) роботу з більш високою заробітною платою', 'інші соціально-побутові вигоди', 'інші причини'));

create domain disability_d as varchar
check(value in ('Ні - не відчуваю труднощів', 'Так - відчуваю деякі труднощі', 'Так - відчуваю великі труднощі', 'Взагалі не можу цього зробити'));

create domain building_type_d as varchar
```

```
check(value in ('квартира у дво- або багатоквартирному будинку',
'квартира в нежитловій будівлі', 'одноквартирний будинок
(індивідуальний)', 'частина одноквартирного будинку
(індивідуального)', 'кімната (частина кімнати) у гуртожитку',
'кімната (частина кімнати) у готелі', 'установа із рухомим
складом населення', 'інституційна установа', 'інше житлове
приміщення', 'житлового приміщення немає (безпритульний)'));

create domain ownership_d as varchar
check(value in ('приватна власність членів домогосподарства',
'державна власність', 'комунальна власність', 'приватна
власність юридичних осіб', 'оренда (найм) у окремих громадян',
'не знаю'));
```

ДОДАТОК А.2

Запити на створення таблиць бази даних

```
create table Household
(
     household_id serial not null primary key,
     building_type building_type_d not null,
     internet yes_no not null,
     house_size integer,
     constraint size_check check (house_size > 0),
     ownership ownership_d
);

create table Citizen
(
     citizen_id serial not null primary key,
     household integer not null,
     foreign key(household) references Household(household_id)
     on delete restrict on update cascade,
     age integer not null,
     gender gender_d not null,
     city varchar not null,
     region varchar not null,
     marital_status marital_status_d not null,
     ethnic_origin varchar not null,
     native_language varchar not null,
     ukrainian_language yes_no not null,
     citizenship citizenship_d not null,
     education_level education_level_d not null,
     literacy yes_no not null,
     education education_d not null,
     preschool yes_no,
     job_category varchar,
     job yes_no,
     job_location_city varchar,
     job_location_region varchar,
     job_location_country varchar,
     location_reason location_reason_d,
     job_search yes_no,
     walking disability_d not null,
     eyesight disability_d not null,
     hearing disability_d not null,
     cognitive disability_d not null,
     self_care disability_d not null,
     communication disability_d not null
);
```

ДОДАТОК А.3

Запити на створення тригерів і функцій бази даних

Тригер для того, щоб не дозволити громадянам молодше 15 років обирати будь-який сімейний (шлюбний) стан окрім ‘особа віком молодше 15 років’ та відповідати на запитання щодо зайнятості, а громадяням старше 15 років – обирати сімейний (шлюбний) стан ‘особа віком молодше 15 років’: визначення тригерної функції міститься у лістингу А.3.1, а визначення тригера – у лістингу А.3.2.

```
create or replace function check_15_year_old()
returns trigger as $$
begin
    if new.age < 15 and (new.marital_status != 'особа віком молодше 15 років' or new.job is not null or new.job_location_country is not null or new.job_category is not null or new.job_location_region is not null or new.job_location_city is not null or new.location_reason is not null or new.job_search is not null) then
        raise exception 'Особа віком молодше 15 років';
    end if;
     if new.age >= 15 and new.marital_status = 'особа віком молодше 15 років' then
        raise exception 'Особа віком старше 15 років';
    end if;
    return new;
end;
$$ language plpgsql;
```

Лістинг А.3.1 – Код тригерної функції для обмеження вибору сімейного стану та відповідей на запитання щодо зайнятості залежно від віку громадян

```
create trigger trigger_15_year_old
before insert or update on Citizen
for each row execute function check_15_year_old();
```

Лістинг А.3.2 – Визначення тригера для обмеження вибору сімейного стану та відповідей на запитання щодо зайнятості залежно від віку громадян

Тригер для того, щоб переконатися, що громадяни молодше 10 років обирають значення рівня освіти 'особа віком молодше 10 років', а громадяни старше 10 років не можуть обирати це значення: визначення тригерної функції міститься у лістингу А.3.3, а визначення тригера – у лістингу А.3.4.

```
create or replace function check_education_level()
returns trigger as $$
begin
    if new.age < 10 and new.education_level != 'особа віком молодше 10 років' then raise exception 'Особа віком молодше 10 років';
    end if;
     if new.age >= 10 and new.education_level = 'особа віком молодше 10 років' then raise exception 'Особа віком старше 10 років';
    end if;
    return new;
end;
$$ language plpgsql;
```

Лістинг А.3.3 – Код тригерної функції для обмеження вибору рівня освіти

```
create trigger education_level_trigger
before insert or update on Citizen
for each row execute function check_education_level();
```

Лістинг А.3.4 – Визначення тригера для обмеження вибору рівня освіти

Тригер для того, щоб громадяни молодше 6 років обирали значення типу закладу освіти 'особа віком молодше 6 років', а громадяни старше 6 років не могли обирати це значення: визначення тригерної функції міститься у лістингу А.3.5, а визначення тригера – у лістингу А.3.6.

```
create or replace function check_education()
returns trigger as $$
begin
    if new.age < 6 and new.education != 'особа віком молодше 6 років' then raise exception 'Особа віком молодше 6 років';
    end if;
     if new.age >= 6 and new.education = 'особа віком молодше 6 років' then raise exception 'Особа віком старше 6 років';
```

```
    end if;
    return new;
end;
$$ language plpgsql;
```

Лістинг А.3.5 – Код тригерної функції для обмеження вибору типу закладу освіти, у якому навчається респондент, залежно від віку громадян

```
create trigger education_trigger
before insert or update on Citizen
for each row execute function check_education();
```

Лістинг А.3.6 – Визначення тригера для обмеження вибору типу закладу освіти, у якому навчається респондент, залежно від віку громадян

Тригер для того, щоб переконатися, що громадяни молодше 8 років, які не відвідують загальноосвітній заклад освіти, відповідають на питання щодо відвідування дитиною дошкільного закладу освіти, а громадяни старше 8 років не можуть відповідати на це питання: визначення тригерної функції міститься у лістингу А.3.7, а визначення тригера – у лістингу А.3.8.

```
create or replace function check_preschool()
returns trigger as $$
begin
    if new.age < 8 and (new.education = 'особа віком молодше 6
років' or new.education = 'не навчаюся') and new.preschool is
null then raise exception 'Не відмічено відвідування дитиною
дошкільного закладу освіти';
    end if;
     if new.age >= 8 and new.preschool is not null then
          raise exception 'Особа віком старше 8 років';
     end if;
    return new;
end;
$$ language plpgsql;
```

Лістинг А.3.7 – Код тригерної функції для обмеження відповіді на питання щодо відвідування дошкільного закладу освіти залежно від віку громадян

```
create trigger preschool_trigger
before insert or update on Citizen
for each row execute function check_preschool();
```

Лістинг А.3.8 – Визначення тригера для обмеження відповіді на питання щодо відвідування дошкільного закладу освіти залежно від віку громадян

Тригер для обмеження відповідей на питання щодо розміру загальної площі житлового приміщення та типу володіння житловим приміщенням: визначення тригерної функції міститься у лістингу А.3.9, а визначення тригера – у лістингу А.3.10.

```
create or replace function check_house()
returns trigger as $$
begin
    if (new.building_type = 'квартира у дво- або багатоквартирному будинку' or new.building_type = 'квартира в нежитловій будівлі' or new.building_type = 'одноквартирний будинок (індивідуальний)' or new.building_type = 'частина одноквартирного будинку (індивідуального)' or new.building_type = 'кімната (частина кімнати) у гуртожитку') and (new.house_size is null or new.ownership is null) then raise exception 'Немає відповіді на питання для одноквартирного будинку (індивідуального), частини одноквартирного будинку (індивідуального), окремої квартири, комунальної квартири)';
    end if;
     if (new.building_type = 'кімната (частина кімнати) у готелі' or new.building_type = 'установа із рухомим складом населення' or new.building_type = 'інституційна установа' or new.building_type = 'інше житлове приміщення' or new.building_type = 'житлового приміщення немає (безпритульний)') and (new.house_size is not null or new.ownership is not null) then raise exception 'Відповіді на питання для одноквартирного будинку (індивідуального), частини одноквартирного будинку (індивідуального), окремої квартири, комунальної квартири)';
    end if;
    return new;
end;
$$ language plpgsql;
```

Лістинг А.3.9 – Код тригерної функції для обмеження відповідей на питання щодо розміру загальної площі житлового приміщення та типу володіння житловим приміщенням

```
create trigger house_trigger
before insert or update on Household
for each row execute function check_house();
```

Лістинг А.3.10 – Визначення тригера для обмеження відповідей на питання щодо розміру загальної площі житлового приміщення та типу володіння житловим приміщенням

ДОДАТОК Б

Лістинг програмної реалізації проєкту DataRandomizer

```
using System;
using System.Collections.Generic;
using System.IO;
namespace DataRandomizer
{
    internal class Program
    {
        static List<string> gender = new List<string> { "Чоловіча", "Жіноча" };
        static List<string> marital_status = new List<string> { "ніколи не перебував(ла) у шлюбі", "перебуваю в зареєстрованому шлюбі", "перебуваю в незареєстрованому шлюбі", "розлучений(а) (зареєстроване розірвання шлюбу)", "розійшовся(лася) (незареєстроване розірвання шлюбу)" };
        static List<string> marital_men = new List<string> { "удівець" };
        static List<string> marital_women = new List<string> { "удова" };
        static string marital_minors = "особа віком молодше 15 років";
        static List<string> citizenship = new List<string> { "громадянство іншої держави", "особа без громадянства", "громадянство не визначене", "громадянство України" };
        static List<string> education_level = new List<string> { "третій (освітньо-науковий/освітньо-творчий) рівень вищої освіти", "другий (магістерський) рівень вищої освіти", "перший (бакалаврський) рівень вищої освіти (базова вища)", "фахова передвища, професійна (професійно-технічна)", "початковий рівень (короткий цикл) вищої освіти (неповна вища, середня спеціальна)", "базова середня (неповна середня)", "профільна середня (повна загальна середня)" };
        static List<string> education_level_minors = new List<string> { "початкова (початкова загальна)" };
        static string no_education = "не маю освіти";
        static string education_level_children = "особа віком молодше 10 років";
        static string education_children = "особа віком молодше 6 років";
        static List<string> education = new List<string> { "заклад вищої освіти", "заклад фахової передвищої освіти", "заклад професійної (професійно-технічної) освіти", "інший заклад освіти", "не навчаюся" };
        static string education_minors = "заклад загальної середньої освіти";
        static List<string> yes_no = new List<string> { "так", "ні" };
```

```
        static List<string> location_reason = new List<string>
{ "не знайшов(ла) роботу в цьому населеному пункті", "знайшов(ла)
роботу з більш високою заробітною платою", "інші соціально-
побутові вигоди", "інші причини" };
        static List<string> disability = new List<string> { "Ні
- не відчуваю труднощів", "Так - відчуваю деякі труднощі", "Так
- відчуваю великі труднощі", "Взагалі не можу цього зробити" };
        static List<string> building_type = new List<string>
{ "квартира у дво- або багатоквартирному будинку", "квартира в
нежитловій будівлі", "одноквартирний будинок (індивідуальний)",
"частина одноквартирного будинку (індивідуального)", "кімната
(частина кімнати) у гуртожитку", "кімната (частина кімнати) у
готелі", "установа із рухомим складом населення", "інституційна
установа", "інше житлове приміщення", "житлового приміщення
немає (безпритульний)" };
        static List<string> ownership = new List<string>
{ "приватна власність членів домогосподарства", "державна
власність", "комунальна власність", "приватна власність
юридичних осіб", "оренда (найм) у окремих громадян", "не знаю" };
        static List<string> region = new List<string>
{ "Одеська", "Львівська", "Київська"};
        static List<string> city_kyiv = new List<string>
{ "Тараща", "Яготин", "Київ" };
        static List<string> city_odesa = new List<string>
{ "Ізмаїл", "Кілія", "Одеса" };
        static List<string> city_lviv = new List<string>
{ "Буськ", "Дубляни", "Львів" };
        static List<string> country = new List<string>
{ "Сполучені Штати Америки", "Польща", "Німеччина", "Франція",
"Італія" };
        static List<string> languages = new List<string>
{ "кримськотатарська", "російська", "польська", "румунська",
"українська", "англійська", "німецька", "іспанська",
"французька", "італійська" };
        static List<string> ethnic_origin = new List<string>
{ "кримський татарин (-ка)", "росіянин (-ка)", "білорус (-ка)",
"поляк (-ка)", "румун (-ка)", "молдаванин (-ка)", "українець (-
ка)" };
        static List<string> job_category = new List<string>
{ "Будівництво, архітектура", "Бухгалтерія, аудит", "Освіта,
наука", "Продаж, закупівля", "Робочі спеціальності, виробництво",
"Сільське господарство, агробізнес" };
        static void Main(string[] args)
        {
            Random random = new Random();
            // випадковим чином генеруємо 100000 домогосподарств
            int household = 0, citizen = 0;
            StreamWriter writer = new
StreamWriter("C:\\Users\\Asus\\Documents\\PopulationCensus\\Hous
ehold.txt");
            StreamWriter writer2 = new
StreamWriter("C:\\Users\\Asus\\Documents\\PopulationCensus\\Citi
zen.txt");
```

```
            for (int i = 0; i < 100000; i++)
            {
                // id домогосподарства
                household++;
                //   генеруємо   кількість   людей   у   цьому
домогосподарстві
                int number_of_people = random.Next(1, 8);
                string random_region = null, random_city = null;
                // генеруємо область
                int region_index = random.Next(region.Count + 1);
                if (region_index < region.Count)
                    random_region = region[region_index];
                // у Київській області більша кількість жителів
                else random_region = region[region.Count - 1];
                // генеруємо місто
                if (random_region == "Одеська")
                {
                    // з більшою ймовірністю громадяни житимуть
в обласних центрах
                    int                city_index                =
random.Next(city_odesa.Count + 2);
                    if (city_index < city_odesa.Count)
                        random_city = city_odesa[city_index];
                    else               random_city               =
city_odesa[city_odesa.Count - 1];
                }
                else if (random_region == "Львівська")
                {
                    int city_index = random.Next(city_lviv.Count
+ 2);
                    if (city_index < city_lviv.Count)
                        random_city = city_lviv[city_index];
                    else random_city = city_lviv[city_lviv.Count
- 1];
                }
                else
                {
                    int city_index = random.Next(city_kyiv.Count
+ 2);
                    if (city_index < city_kyiv.Count)
                        random_city = city_kyiv[city_index];
                    else random_city = city_kyiv[city_kyiv.Count
- 1];
                }
                // тип будівлі
                int             buiding_type_index             =
random.Next(building_type.Count);
                string           random_building_type          =
building_type[buiding_type_index];
                string       random_ownership       =       null,
random_house_size = null;
                // питання щодо інтернету
                int internet_index = random.Next(yes_no.Count);
```

```
                string random_internet = yes_no[internet_index];
                // питання для певних типів будівель
                if (random_building_type == "квартира у дво- або
багатоквартирному будинку" || random_building_type == "квартира
в нежитловій будівлі" || random_building_type == "одноквартирний
будинок (індивідуальний)" || random_building_type == "частина
одноквартирного будинку (індивідуального)" ||
random_building_type == "кімната (частина кімнати) у гуртожитку")
                {
                    // розмір будівлі
                    int house_size = random.Next(20, 200);
                    random_house_size =
Convert.ToString(house_size);
                    // тип володіння житловим приміщенням
                    int ownership_index =
random.Next(ownership.Count);
                    random_ownership= ownership[ownership_index];
                }
                // записуємо результати до файлу

writer.WriteLine($"{household}|{random_building_type}|{random_in
ternet}|{random_house_size}|{random_ownership}");
                //  генеруємо інформацію щодо кожного жителя
домогосподарства окремо
                for (int j = 0; j < number_of_people; j++)
                {
                    citizen++;
                    // генеруємо стать
                    int gender_index = random.Next(gender.Count);
                    string random_gender = gender[gender_index];
                    string   random_location_reason   =   null,
random_job = null, random_marital_status = null, random_job_city
= null, random_job_region = null, random_ethnic_origin = null,
random_native_language  =  null,  random_citizenship  =  null,
random_walking = null, random_eyesight = null, random_hearing =
null,  random_cognitive  =  null,  random_self_care  =  null,
random_communication   =   null,   random_literacy   =   null,
random_education   =   null,   random_preschool   =   null,
random_job_category   =   null,   random_ukrainian   =   null,
random_job_country   =   null,   random_job_search   =   null,
random_education_level = null;
                    string citizen_id =
Convert.ToString(household) + Convert.ToString(j);
                    // генеруємо вік
                    int age = 0;
                    do
                    {
                        age = random.Next(0, 90);
                    }
                    // уникаємо ситуацій, коли дитина єдиною у
домогосподарстві
                    while (age < 15 && j < 1);
                    // громадянство
```

```
                    int citizenship_index = random.Next(100);
                    if (random_city == "Київ")
                        random_citizenship                          =
Citizenship(citizenship_index, 30, 32, 35);
                    else if (random_city == "Одеса")
                        random_citizenship                          =
Citizenship(citizenship_index, 25, 28, 30);
                    else if (random_city == "Львів")
                        random_citizenship                          =
Citizenship(citizenship_index, 20, 22, 25);
                    else
                        random_citizenship                          =
Citizenship(citizenship_index, 5, 7, 10);
                    if   (random_citizenship   !=   "громадянство
України")
                    {
                        int         ethnic_origin_index         =
random.Next(ethnic_origin.Count);
                        // випадкова національність
                        random_ethnic_origin                        =
ethnic_origin[ethnic_origin_index];
                        int        native_language_index        =
random.Next(languages.Count);
                        // випадкова рідна мова
                        random_native_language                      =
languages[native_language_index];
                    }
                    else
                    if (random_region == "Одеська")
                    {
                        // генеруємо рідну мову (для Одеської
області українська, російська або румунська)
                        int        native_language_index        =
random.Next(100);
                        if (native_language_index < 51)
                            random_native_language                  =
languages[4];
                        else if (native_language_index < 81)
                            random_native_language                  =
languages[1];
                        else       random_native_language       =
languages[3];
                        // генеруємо національність
                        int ethnic_origin_index = random.Next(0,
100);
                        if (ethnic_origin_index == 0)
                            random_ethnic_origin                    =
ethnic_origin[0];
                        else if (ethnic_origin_index == 1)
                            random_ethnic_origin                    =
ethnic_origin[2];
                        else if (ethnic_origin_index == 2)
```

```
                        random_ethnic_origin                =
ethnic_origin[3];
                    else if (ethnic_origin_index < 16)
                        random_ethnic_origin                =
ethnic_origin[1];
                    else if (ethnic_origin_index < 32)
                        random_ethnic_origin                =
ethnic_origin[4];
                    else if (ethnic_origin_index < 48)
                        random_ethnic_origin                =
ethnic_origin[5];
                    else        random_ethnic_origin        =
ethnic_origin[6];
                }
                else if (random_region == "Львівська")
                {
                    int        native_language_index        =
random.Next(100);
                    if (native_language_index < 81)
                        random_native_language= languages[4];
                    else if (native_language_index < 94)
                        random_native_language= languages[2];
                    else random_native_language=languages[3];
                    int ethnic_origin_index = random.Next(0,
100);
                    if (ethnic_origin_index == 0)
                        random_ethnic_origin                =
ethnic_origin[0];
                    else if (ethnic_origin_index == 1)
                        random_ethnic_origin                =
ethnic_origin[1];
                    else if (ethnic_origin_index == 2)
                        random_ethnic_origin                =
ethnic_origin[2];
                    else if (ethnic_origin_index < 16)
                        random_ethnic_origin                =
ethnic_origin[5];
                    else if (ethnic_origin_index < 32)
                        random_ethnic_origin                =
ethnic_origin[4];
                    else if (ethnic_origin_index < 48)
                        random_ethnic_origin                =
ethnic_origin[3];
                    else        random_ethnic_origin        =
ethnic_origin[6];
                }
                else
                {
                    int        native_language_index        =
random.Next(100);
                    if (native_language_index < 76)
                        random_native_language              =
languages[4];
```

```
                        else
                            random_native_language              =
languages[1];
                        int ethnic_origin_index = random.Next(0,
100);
                        if (ethnic_origin_index == 0)
                            random_ethnic_origin                =
ethnic_origin[4];
                        else if (ethnic_origin_index == 1)
                            random_ethnic_origin                =
ethnic_origin[5];
                        else if (ethnic_origin_index == 2)
                            random_ethnic_origin                =
ethnic_origin[2];
                        else if (ethnic_origin_index < 16)
                            random_ethnic_origin                =
ethnic_origin[0];
                        else if (ethnic_origin_index < 32)
                            random_ethnic_origin                =
ethnic_origin[3];
                        else if (ethnic_origin_index < 48)
                            random_ethnic_origin                =
ethnic_origin[1];
                        else        random_ethnic_origin        =
ethnic_origin[6];
                    }
                    // сімейний стан
                    if (age < 15)
                    {
                        random_marital_status = marital_minors;
                    }
                    else
                    {
                        if (random_gender == "Жіноча")
                        {
                            List<List<string>>  list_of_lists   =
new List<List<string>> { marital_status, marital_women };
                            int        random_list_index        =
random.Next(list_of_lists.Count);
                            List<string>     selected_list      =
list_of_lists[random_list_index];
                            int   random_marital_status_index   =
random.Next(selected_list.Count);
                            random_marital_status               =
selected_list[random_marital_status_index];
                        }
                        else
                        {
                            List<List<string>>  list_of_lists   =
new List<List<string>> { marital_status, marital_men };
                            int        random_list_index        =
random.Next(list_of_lists.Count);
```

```
                            List<string>     selected_list      =
list_of_lists[random_list_index];
                            int   random_marital_status_index   =
random.Next(selected_list.Count);
                            random_marital_status               =
selected_list[random_marital_status_index];
                        }
                    }
                    // володіння українською мовою
                    if (random_native_language == "українська")
                        random_ukrainian = yes_no[0];
                    else
                    {
                        if  (random_citizenship  !=  "громадянство
України")
                        {
                            int ukrainian_index = random.Next(2);
                            if (ukrainian_index == 1)
                                random_ukrainian = yes_no[0];
                            else random_ukrainian = yes_no[1];
                        }
                        else
                        {
                            int ukrainian_index= random.Next(10);
                            if (ukrainian_index == 1)
                                random_ukrainian = yes_no[1];
                            else random_ukrainian = yes_no[0];
                        }
                    }
                    // наявність інвалідності
                    random_hearing = Disability();
                    random_walking = Disability();
                    random_eyesight = Disability();
                    random_communication = Disability();
                    random_cognitive = Disability();
                    random_self_care = Disability();
                    // рівень освіти
                    if (age < 10)
                        random_education_level                  =
education_level_children;
                    else
                    {
                        int education_index = random.Next(100);
                        if (education_index < 10)
                            random_education_level= no_education;
                        else
                        {
                            if (age < 15)
                                random_education_level          =
education_level_minors[0];
                            else
                            {
```

```
                                List<List<string>> list_of_lists
=      new      List<List<string>>      {      education_level,
education_level_minors };
                                int      random_list_index      =
random.Next(list_of_lists.Count);
                                List<string>   selected_list   =
list_of_lists[random_list_index];
                                int  random_education_level_index
= random.Next(selected_list.Count);
                                random_education_level          =
selected_list[random_education_level_index];
                            }
                        }
                    }
                    // грамотність
                    int literacy_index = random.Next(2);
                    if (random_education_level =="не маю освіти")
                    {
                        if (literacy_index == 0)
                            random_literacy = yes_no[1];
                        else random_literacy = yes_no[0];
                    }
                    else
                        random_literacy = yes_no[0];
                    // навчальний заклад
                    if (age < 6)
                        random_education = education_children;
                    else
                    {
                        if (age < 15)
                            random_education = education_minors;
                        else
                        //   громадяни   молодше   25   здебільшого
навчатимуться у навчальних закладах
                        if (age < 25)
                        {
                            int      random_education_index      =
random.Next(education.Count);
                            random_education                    =
education[random_education_index];
                        }
                        else
                        {
                            int      random_education_index      =
random.Next(100);
                            if(random_education_index==0)
                            {
                                random_education_index          =
random.Next(education.Count);
                                random_education                =
education[random_education_index];
                            }
                            else
```

```
                        random_education = education[4];
                    }
                }
                // дитсадок
                if(age<6)
                {
                    int preschool_index = random.Next(4);
                    if (preschool_index == 0)
                        random_preschool = yes_no[0];
                    else random_preschool = yes_no[1];
                }
                // сфера діяльності
                if (age > 15)
                {
                    int category_index = random.Next(100);
                    if (random_city == "Київ")
                        random_job_category                 =
JobCategory(category_index, 5, 10, 20, 40, 70);
                    else if (random_city == "Одеса")
                        random_job_category                 =
JobCategory(category_index, 35, 65, 75, 83, 90);
                    else if (random_city == "Львів")
                        random_job_category                 =
JobCategory(category_index, 20, 53, 60, 72, 80);
                    else
                        random_job_category                 =
JobCategory(category_index, 17, 32, 57, 72, 88);
                }
                // наявність роботи
                if (age > 15)
                {
                    int job_index = random.Next(100);
                    if (random_city == "Київ")
                        random_job                          =
JobByCategory(random_job_category, job_index, 20, 90, 40, 30, 80,
95);
                    else if (random_city == "Одеса")
                        random_job                          =
JobByCategory(random_job_category, job_index, 80, 75, 10, 60, 94,
83);
                    else if (random_city == "Львів")
                        random_job                          =
JobByCategory(random_job_category, job_index, 50, 91, 76, 24, 56,
16);
                    else
                        random_job                          =
JobByCategory(random_job_category, job_index, 90, 34, 17, 50, 90,
82);
                }
                // пошук роботи
                if (random_job=="ні")
                {
                    int job_index = random.Next(3);
```

```
                    if (job_index == 0)
                        random_job_search= yes_no[0];
                    else
                        random_job_search = yes_no[1];
                }
                // країна працевлаштування
                if (random_job =="так")
                {
                    int job_country_index = random.Next(100);
                    if (random_city == "Одеса")
                        random_job_country                  =
JobCountry(job_country_index, 10, 11, 12, 13, 14);
                    else if (random_city == "Київ")
                        random_job_country                  =
JobCountry(job_country_index, 1, 14, 15, 16, 17);
                    else if (random_city == "Львів")
                        random_job_country                  =
JobCountry(job_country_index, 1, 2, 13, 14, 15);
                    else
                        random_job_country                  =
JobCountry(job_country_index, 1, 2, 3, 7, 10);
                }
                // регіон працевлаштування
                if (random_job_country == "Україна")
                    random_job_region = random_region;
                // місто працевлаштування
                if (random_job_region == "Одеська")
                {
                    int city_index = random.Next(100);
                    if (city_index < 70)
                        random_job_city = random_city;
                    else if (city_index < 90)
                        random_job_city = "Одеса";
                    else
                    {
                        int            rand_city            =
random.Next(city_odesa.Count);
                        random_job_city                     =
city_odesa[rand_city];
                    }
                }
                else if (random_job_region == "Київська")
                {
                    int city_index = random.Next(100);
                    if (city_index < 50)
                        random_job_city = random_city;
                    else if (city_index < 90)
                        random_job_city = "Київ";
                    else
                    {
                        int            rand_city            =
random.Next(city_kyiv.Count);
                        random_job_city=city_kyiv[rand_city];
```

```
                    }
                }
                else if (random_job_region == "Львівська")
                {
                    int city_index = random.Next(100);
                    if (city_index < 85)
                        random_job_city = random_city;
                    else if (city_index < 93)
                        random_job_city = "Львів";
                    else
                    {
                        int            rand_city            =
random.Next(city_lviv.Count);
                        random_job_city=city_lviv[rand_city];
                    }
                }
                //  чому  відрізняються  місце  проживання  та
місце працевлаштування
                if (random_job != "ні" && random_job != null
&& random_job_city != random_city)
                {
                    int job_index = random.Next(100);
                    if (random_city == "Київ")
                        random_location_reason              =
Reason(random_job_category,  job_index,  50,  69,  82,  80,  86,  90,
40, 70, 89, 29, 78, 89, 14, 23, 56, 45, 59, 78);
                    else if (random_city == "Одеса")
                        random_location_reason              =
Reason(random_job_category,  job_index,  71,  79,  84,  23,  43,  67,
81, 84, 90, 91, 95, 97, 24, 87, 95, 37, 68, 79);
                    else if (random_city == "Львів")
                        random_location_reason              =
Reason(random_job_category,  job_index,  25,  32,  48,  84,  89,  96,
78, 79, 99, 32, 58, 87, 94, 96, 98, 58, 78, 92);
                    else       random_location_reason       =
Reason(random_job_category,  job_index,  67,  83,  96,  43,  65,  92,
89, 94, 99, 63, 73, 94, 93, 96, 98, 23, 67, 93);
                }
writer2.WriteLine($"{citizen_id}|{household}|{age}|{random_gende
r}|{random_city}|{random_region}|{random_marital_status}|{random
_ethnic_origin}|{random_native_language}|{random_ukrainian}|{ran
dom_citizenship}|{random_education_level}|{random_literacy}|{ran
dom_education}|{random_preschool}|{random_job_category}|{random_
job}|{random_job_city}|{random_job_region}|{random_job_country}|
{random_location_reason}|{random_job_search}|{random_walking}|{r
andom_eyesight}|{random_hearing}|{random_cognitive}|{random_self
_care}|{random_communication}");
                }
            }
            writer.Close();
            writer2.Close();
        }
```

```
        static string JobLocation(int job_index, int x, int y,
int z)
        {
            if (job_index < x)
                return location_reason[0];
            else if (job_index < y)
                return location_reason[1];
            else if (job_index < z)
                return location_reason[2];
            else
                return location_reason[3];
        }
        static string JobCategory(int category_index, int a, int
b, int c, int d, int e)
        {
            if (category_index < a)
                return job_category[0];
            else if (category_index < b)
                return job_category[1];
            else if (category_index < c)
                return job_category[2];
            else if (category_index < d)
                return job_category[3];
            else if (category_index < e)
                return job_category[4];
            else
                return job_category[5];
        }
        static string Citizenship(int citizenship_index, int x,
int y, int z)
        {
            if (citizenship_index < x)
                return citizenship[0];
            else if (citizenship_index < y)
                return citizenship[1];
            else if (citizenship_index < z)
                return citizenship[2];
            else
                return citizenship[3];
        }
        static string Job(int job_index, int x)
        {
            if (job_index < x)
                return yes_no[0];
            else
                return yes_no[1];
        }
        static string JobByCategory(string random_job_category,
int job_index, int a, int b, int c, int d, int e, int f)
        {
            if (random_job_category== "Будівництво, архітектура")
                return Job(job_index, a);
            else if (random_job_category == "Бухгалтерія, аудит")
```

```
                return Job(job_index, b);
            else if (random_job_category == "Освіта, наука")
                return Job(job_index, c);
            else if (random_job_category == "Продаж, закупівля")
                return Job(job_index, d);
            else    if    (random_job_category    ==    "Робочі
спеціальності, виробництво")
                return Job(job_index, e);
            else
                return Job(job_index, f);
        }
        static  string  Reason(string  random_job_category,  int
job_index, int a, int b, int c, int d, int e, int f, int g, int
h, int i, int j, int k, int l, int m, int n, int o, int p, int q,
int r)
        {
            if (random_job_category== "Будівництво, архітектура")
                return JobLocation(job_index, a, b, c);
            if (random_job_category == "Бухгалтерія, аудит")
                return JobLocation(job_index, d, e, f);
            if (random_job_category == "Освіта, наука")
                return JobLocation(job_index, g, h, i);
            if (random_job_category == "Продаж, закупівля")
                return JobLocation(job_index, j, k, l);
            if  (random_job_category  ==  "Робочі  спеціальності,
виробництво")
                return JobLocation(job_index, m, n, o);
            else
                return JobLocation(job_index, p, q, r);
        }
        static  string  JobCountry(int  job_country_index,  int  a,
int b, int c, int d, int e)
        {
            if (job_country_index < a) return country[0];
            else if(job_country_index < b)
                return country[1];
            if (job_country_index < c) return country[2];
            else if (job_country_index < d) return country[3];
            else if (job_country_index < e) return country[4];
            else return "Україна";
        }
        static string Disability()
        {
            Random random = new Random();
            int disability_index = random.Next(100);
            if (disability_index < 80) return disability[0];
            else if (disability_index < 90) return disability[1];
            else if (disability_index < 95) return disability[2];
            else return disability[3];
        }
    }
}
```

ДОДАТОК В

Лістинг програмної реалізації проєкту K-modes

```
from sqlalchemy import create_engine
import pandas as pd
from concurrent.futures import ThreadPoolExecutor
# Параметри підключення до бази даних
username = 'postgres'
password = 'postgres'
database = 'PopulationCensus'
host = 'localhost'
port = '5433'
# Створення URL підключення
DATABASE_URL =
f'postgresql+psycopg2://{username}:{password}@{host}:{port}/{dat
abase}'
# Створення движка SQLAlchemy
engine = create_engine(DATABASE_URL)
# Отримуємо список унікальних міст з таблиці Citizen
city_query = "SELECT DISTINCT city FROM Citizen "
cities_df = pd.read_sql(city_query, engine)
cities = cities_df['city'].tolist()
# Функція для обробки даних для одного міста
def process_city(city):
    # Формуємо запит для поточного міста
    query = f"""
        SELECT * FROM Citizen
        JOIN Household ON Citizen.household =
Household.household_id
        WHERE Citizen.city = '{city}'
    """
    # Виконуємо запит та завантажуємо дані у DataFrame
    df = pd.read_sql(query, engine)
    # Видаляємо непотрібні стовпці
    df = df.drop(columns=['citizen_id', 'household_id',
'household','city', 'region'], errors='ignore')
    # Визначаємо вікові групи
    bins = [0, 18, 30, 45, 60, 90]  # Визначаємо межі вікових
груп
    labels = ['0-17', '18-29', '30-44', '45-59', '60+']  # Назви
вікових груп
    # Замінюємо вік на вікові групи
    df['age_group'] = pd.cut(df['age'].astype(int), bins=bins,
labels=labels, right=False)
    # Тепер можна видалити стовпець 'age'
    df = df.drop(columns=['age'], errors='ignore')
    bins = [20, 57, 94, 131, 168, 200]
    labels = ['20-56', '57-93', '94-130', '131-167', '168-200']
```

```
    df['house_size_interval'] =
pd.cut(df['house_size'].astype(float), bins=bins, labels=labels,
right=False)
    df = df.drop(columns=['house_size'], errors='ignore')
    # Перетворимо всі значення на рядковий тип
    df = df.astype(str)
    # Повертаємо назву міста та DataFrame
    return city, df
# Словник для зберігання результатів
results = {}
with ThreadPoolExecutor(max_workers=8) as executor:
    for city, df in executor.map(process_city, cities):
        results[city] = df  # Зберігаємо DataFrame для кожного
міста у словнику
# Визначаємо групи стовпців для кожної підмножини
subset_columns = {
    'Відомості щодо житлових умов' : ['house_size_interval',
'building_type', 'ownership'],
    'Відомості щодо етнічного походження': ['ethnic_origin',
'native_language', 'ukrainian_language', 'citizenship'],
    'Відомості щодо освіти': ['education_level', 'education',
'preschool'],
    'Відомості щодо наявності роботи для різних сфер діяльності':
['job', 'job_category', 'location_reason'],
    'Відомості щодо місцезнаходження основної роботи':
['job_category', 'job_location_city', 'job_location_country'],
    'Відомості щодо статусу інвалідності': ['walking',
'eyesight', 'hearing', 'cognitive', 'self_care',
'communication',]
}
# Створюємо словник для зберігання підмножин
subsets = {}
# Ітеруємо по кожному місту і створюємо підмножини
for city, df in results.items():
    subsets[city] = {}  # Словник для підмножин поточного міста
    for subset_name, columns in subset_columns.items():
        # Створюємо підмножину, вибираючи лише потрібні стовпці
        subset_df = df[columns].copy() if all(col in df.columns
for col in columns) else pd.DataFrame()
        # Зберігаємо підмножину у словнику
        subsets[city][subset_name] = subset_df
from kmodes.kmodes import KModes
import os
# Ініціалізація директорії для збереження файлів
output_dir = r"C:\Users\Asus\Documents\PopulationCensus\1994"
# Ітеруємо по кожному місту та його підмножинам, застосовуємо K-
Modes
k_modes_results = {}
for city, city_subsets in subsets.items():
    k_modes_results[city] = {}    # Словник для зберігання
результатів кластеризації
    for subset_name, subset_df in city_subsets.items():
        if not subset_df.empty:
```

```
            cost = []  # Список для зберігання вартості
            clustering_results = {}  # Словник для зберігання міток кластерів

            # Пробуємо кількість кластерів від 2 до 10, але з перериванням при досягненні ліктя
            for n_clusters in range(2, 11):
                km = KModes(n_clusters=n_clusters, init='Huang', n_init=5, verbose=1)
                labels = km.fit_predict(subset_df)
                # Зберігаємо вартість кластеризації та мітки кластерів
                cost.append(km.cost_)
                clustering_results[n_clusters] = labels
                # Припиняємо цикл, якщо зменшення вартості стало незначним
                if n_clusters > 2 and (cost[-2] - cost[-1]) < 0.1 * cost[-2]:
                    optimal_n_clusters = n_clusters - 1
                    break
                else:
                # Якщо лікоть не знайдено, вибираємо максимальну кількість кластерів
                optimal_n_clusters = 10
            # Застосовуємо оптимальні мітки кластерів до DataFrame
            subset_df['cluster'] = clustering_results[optimal_n_clusters]
            k_modes_results[city][subset_name] = subset_df
            subset_df.to_csv(os.path.join(output_dir, f"{city}_{subset_name}.csv"), index=False)
import pandas as pd
import os
input_dir = r"C:\Users\Asus\Documents\PopulationCensus\2024"
subsets_from_files = {}
for filename in os.listdir(input_dir):
    if filename.endswith('.csv'):
        city, subset_name = filename.replace('.csv', '').split('_', 1)
        # Завантажуємо дані з файлу в DataFrame
        file_path = os.path.join(input_dir, filename)
        df = pd.read_csv(file_path)
        # Ініціалізуємо словник для поточного міста, якщо він ще не існує
        if city not in subsets_from_files:
            subsets_from_files[city] = {}
        # Зберігаємо DataFrame у словнику під відповідною підмножиною
        subsets_from_files[city][subset_name] = df
import matplotlib.pyplot as plt
import os
# Папка для збереження зображень
save_dir = r"C:\Users\Asus\Documents\PopulationCensus\2024"
```

```
os.makedirs(save_dir, exist_ok=True)
n_rows = 2  # кількість рядків для підграфіків
n_cols = 1  # кількість стовпців для підграфіків
# Визначаємо розмір графіків
plt.figure(figsize=(10, 10))
# Змінна для відстеження номера підграфіка та номера файлу
plot_index = 1
file_index = 1  # Лічильник для файлів
cluster_counts_summary = {}
for city, city_subsets in subsets_from_files.items():
    for subset_name, subset_df in city_subsets.items():
        # Візуалізація частоти значень ознак за кластерами
        for column in subset_df.columns[:-1]:    # виключаємо
стовпець 'cluster'
            # Створюємо частотну таблицю для кожної ознаки
            freq_table                                          =
subset_df.groupby('cluster')[column].value_counts().unstack(fill
_value=0)
            # Створюємо підграфік
            plt.subplot(n_rows, n_cols, plot_index)
            freq_table.plot(kind='bar',             stacked=True,
ax=plt.gca())
            # Зберігаємо результати підрахунку до словника
            if city not in cluster_counts_summary:
                cluster_counts_summary[city] = {}
            if subset_name not in cluster_counts_summary[city]:
                cluster_counts_summary[city][subset_name] = {}
            # Зберігаємо до словника
            cluster_counts_summary[city][subset_name][column]   =
freq_table
            plt.title(f'{city} {column} {subset_name}')
            plt.xlabel('Кластер')
            plt.ylabel('Частота')
            # Розміщуємо легенду зовні графіка
            plt.legend(title=column,  bbox_to_anchor=(1.05,  1),
loc='upper left')
            plot_index += 1
            if plot_index > n_rows * n_cols:
                plt.tight_layout()
                plt.savefig(os.path.join(save_dir,
f'clusters_distribution_{city}_{subset_name}_{file_index}.png'),
bbox_inches='tight')
                plt.figure(figsize=(10, 10))
                plot_index = 1  # Скидання індексу
                file_index += 1
# Зберігаємо будь-які графіки, що залишилися.
if plot_index > 1:
    plt.tight_layout()
    # Зберігаємо останній набір графіків, додаємо file_index
    plt.savefig(os.path.join(save_dir,
f'clusters_distribution_{city}_{subset_name}_{file_index}.png'),
bbox_inches='tight')
```

ДОДАТОК Г

Лістинг програмної реалізації проєкту QualityAssessment

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False)
X = subsets_from_files["Київ"]["Відомості щодо наявності роботи
для різних сфер діяльності"].iloc[:, :-1]
y = subsets_from_files["Київ"]["Відомості щодо наявності роботи
для різних сфер діяльності"].iloc[:, -1]
encoded_data = encoder.fit_transform(X)

from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import silhouette_score

score = silhouette_score(encoded_data, y, metric='hamming')
print("Silhouette Score:", score)
score = davies_bouldin_score(encoded_data, y)
print("Davies-Bouldin Index:", score)
score = calinski_harabasz_score(encoded_data, y)
print("Calinski-Harabasz Index:", score)
```

ДОДАТОК Д

Лістинг програмної реалізації проєкту CensusAnalysis

```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
# Ініціалізація змінної для зберігання ланцюгів кластерів
cluster_chains = {}
# Функція для обчислення косинусної подібності
def compare_clusters_by_cosine(row1, row2):
    # Перетворюємо рядки в масиви
    vector1 = np.array(row1.values).reshape(1, -1)
    vector2 = np.array(row2.values).reshape(1, -1)
    # Обчислюємо косинусну подібність
    return cosine_similarity(vector1, vector2)[0][0]
# Проходимо по всіх містах і підмножинах
for city, city_subsets in cluster_counts_summary.items():
    cluster_chains[city] = {}  # Ініціалізація словника для
ланцюга міста
    for subset_name, subset_data in city_subsets.items():
        cluster_chains[city][subset_name] = {}  # Ініціалізація
ланцюга для підмножини
        for column, freq_table in subset_data.items():
            # Отримуємо частотні таблиці для кожного року
            freq_table_2024                                    =
cluster_counts_summary[city][subset_name][column]
            freq_table_2014                                    =
cluster_counts_summary2[city][subset_name][column]
            freq_table_2004                                    =
cluster_counts_summary3[city][subset_name][column]
            freq_table_1994                                    =
cluster_counts_summary4[city][subset_name][column]
            # Для кожного кластера 2024 року
            for        cluster_2024,        row_2024        in
freq_table_2024.iterrows():
                # Порівняння з кластерами 2014 року
                most_similar_2014 = None
                max_similarity_2014   =   -1      #   Початково
встановлюємо мінімальну подібність
                for       cluster_2014,       row_2014       in
freq_table_2014.iterrows():
                    similarity_2014                            =
compare_clusters_by_cosine(row_2024, row_2014)
                    if similarity_2014 > max_similarity_2014:
                        max_similarity_2014 = similarity_2014
                        most_similar_2014 = cluster_2014
                # Порівняння з кластерами 2004 року
                most_similar_2004 = None
                max_similarity_2004 = -1
                for       cluster_2004,       row_2004       in
freq_table_2004.iterrows():
```

```
                    similarity_2004 =
compare_clusters_by_cosine(row_2024, row_2004)
                    if similarity_2004 > max_similarity_2004:
                        max_similarity_2004 = similarity_2004
                        most_similar_2004 = cluster_2004
                # Порівняння з кластерами 1994 року
                most_similar_1994 = None
                max_similarity_1994 = -1
                for       cluster_1994,       row_1994       in
freq_table_1994.iterrows():
                    similarity_1994 =
compare_clusters_by_cosine(row_2024, row_1994)
                    if similarity_1994 > max_similarity_1994:
                        max_similarity_1994 = similarity_1994
                        most_similar_1994 = cluster_1994
                # Зберігаємо ланцюг для поточного кластера 2024
року
                cluster_chains[city][subset_name][cluster_2024]
= {
                    '2014': most_similar_2014,
                    '2004': most_similar_2004,
                    '1994': most_similar_1994
                }
# Виведення ланцюга кластерів
for city, city_chains in cluster_chains.items():
    print(f"Місто: {city}")
    for subset_name, subset_chains in city_chains.items():
        print(f"  Підмножина: {subset_name}")
        for cluster_2024, chain in subset_chains.items():
            print(f"    Кластер 2024: {cluster_2024}")
            print(f"      2014: {chain['2014']}")
            print(f"      2004: {chain['2004']}")
            print(f"      1994: {chain['1994']}")
            print()  # Печатаємо порожній рядок для розділення
import matplotlib.pyplot as plt
import networkx as nx
import os
# Функція для побудови графіка ланцюга кластерів
def    plot_cluster_chain(cluster_chain,    city,    subset_name,
save_dir):
    G = nx.DiGraph()  # Орієнтований граф для ланцюга кластерів
    # Додаємо вершини і ребра для кожного року
    for cluster_2024, years in cluster_chain.items():
        # Додаємо вершини для 2024, 2014, 2004, 1994
        G.add_node(f"{cluster_2024}_2024")
        G.add_node(f"{years['2014']}_2014")
        G.add_node(f"{years['2004']}_2004")
        G.add_node(f"{years['1994']}_1994")
        # Додаємо ребра між кластерами 2024 -> 2014 -> 2004 ->
1994
        G.add_edge(f"{cluster_2024}_2024",
f"{years['2014']}_2014")
```

```
        G.add_edge(f"{years['2014']}_2014",
f"{years['2004']}_2004")
        G.add_edge(f"{years['2004']}_2004",
f"{years['1994']}_1994")
    # Малюємо графік з покращенням розташування
    plt.figure(figsize=(12, 8))
    # Застосовуємо специфічний алгоритм розташування для
ланцюгів
    pos = nx.shell_layout(G)    # Альтернативний алгоритм для
ланцюгів, щоб елементи не перекривалися
    nx.draw(G, pos, with_labels=True, node_size=2000,
node_color="skyblue", font_size=10, font_weight="bold",
arrows=True, edge_color="gray")
    # Заголовок і налаштування
    plt.title(f"Ланцюг кластерів для міста {city}, підмножина
{subset_name}")
    plt.tight_layout()
    # Зберігаємо графік
    plt.savefig(os.path.join(save_dir,
f"cluster_chain_{city}_{subset_name}.png"))
    plt.close()
# Папка для збереження зображень
save_dir                                                =
r"C:\Users\Asus\Documents\PopulationCensus\ClusterChains"
os.makedirs(save_dir, exist_ok=True)
# Створення графіків для всіх ланцюгів кластерів
for city, city_subsets in cluster_chains.items():
    for subset_name, subset_chain in city_subsets.items():
        # Побудова графіка ланцюга кластерів для кожного міста і
підмножини
        plot_cluster_chain(subset_chain, city, subset_name,
save_dir)
from sklearn.linear_model import LinearRegression
import numpy as np
# Функція для прогнозування за допомогою лінійної регресії
def forecast_value(years, values):
    # Перетворюємо роки і значення в масиви
    years_array = np.array(years).reshape(-1, 1)  # Роки
    values_array = np.array(values).reshape(-1, 1)   # Значення
характеристик
    # Створюємо модель лінійної регресії
    model = LinearRegression()
    # Навчаємо модель
    model.fit(years_array, values_array)
    # Прогнозуємо для 2034 року
    forecast = model.predict(np.array([[2034]]))
    # Округлюємо результат і перевіряємо, чи не менше 0
    forecast_value = round(forecast[0][0])    # Округлення до
найближчого цілого
    return max(forecast_value, 0)   # Якщо значення менше 0,
повертаємо 0
# Ініціалізація змінної для зберігання ланцюгів кластерів для
2034 року
```

```
cluster_chains_2034 = {}
# Проходимо по всіх містах і підмножинах
for city, city_subsets in cluster_counts_summary.items():
    cluster_chains_2034[city] = {}  # Ініціалізація словника для ланцюга міста
    for subset_name, subset_data in city_subsets.items():
        cluster_chains_2034[city][subset_name] = {}  # Ініціалізація ланцюга для підмножини
        for column, freq_table in subset_data.items():
            # Отримуємо частотні таблиці для кожного року
            freq_table_2024 = cluster_counts_summary[city][subset_name][column]
            freq_table_2014 = cluster_counts_summary2[city][subset_name][column]
            freq_table_2004 = cluster_counts_summary3[city][subset_name][column]
            freq_table_1994 = cluster_counts_summary4[city][subset_name][column]
            # Для кожного кластера 2024 року
            for cluster_2024, row_2024 in freq_table_2024.iterrows():
                if cluster_2024 not in cluster_chains_2034[city][subset_name]:
cluster_chains_2034[city][subset_name][cluster_2024] = {}
                # Додаємо колонку (column) перед характеристиками
                if column not in cluster_chains_2034[city][subset_name][cluster_2024]:
cluster_chains_2034[city][subset_name][cluster_2024][column] = {}
                # Для кожної характеристики в поточному кластері
                for feature in row_2024.index:
                    # Визначаємо значення для кожного року
                    values = []
                    years = [1994, 2004, 2014, 2024]
values.append(freq_table_1994.loc[cluster_2024, feature] if cluster_2024 in freq_table_1994.index else 0)

values.append(freq_table_2004.loc[cluster_2024, feature] if cluster_2024 in freq_table_2004.index else 0)

values.append(freq_table_2014.loc[cluster_2024, feature] if cluster_2024 in freq_table_2014.index else 0)
                    values.append(row_2024[feature])
                    # Прогнозуємо значення для 2034 року
                    predicted_value = forecast_value(years, values)
                    # Зберігаємо прогнозоване значення для 2034 року
cluster_chains_2034[city][subset_name][cluster_2024][column][feature] = predicted_value

# Виведення прогнозованих значень для 2034 року (для всіх характеристик)
```

```
for city, city_chains in cluster_chains_2034.items():
    print(f"\nМісто: {city}")
    for subset_name, subset_chains in city_chains.items():
        print(f"\n  Підмножина: {subset_name}")
        for cluster_2024, cluster_data in subset_chains.items():
            print(f"\n    Кластер 2024: {cluster_2024}")

            # Для кожної колонки і характеристики в поточному
кластері
            for column, column_data in cluster_data.items():
                print(f"\n      Колонка: {column}")
                for       feature,       predicted_value       in
column_data.items():
                    print(f"        {feature} => Прогноз на 2034
рік: {predicted_value}")
            print()  # Печатаємо порожній рядок для розділення
# приводимо поточні та прогнозовані дані до однакового вигляду
freq_dict = {}
for city, city_subsets in cluster_counts_summary.items():
    freq_dict[city]={}
    for subset_name, subset_data in city_subsets.items():
        freq_dict[city][subset_name]={}
        for column, freq_table in subset_data.items():
            freq_dict[city][subset_name][column]={}
            for cluster, row in freq_table.iterrows():
                freq_dict[city][subset_name][column][cluster]   =
{}
                for     feature,     value     in     row.items():
freq_dict[city][subset_name][column][cluster][feature] = value
print(freq_dict)
freq_dict2 = {}
for city, city_chains in cluster_chains_2034.items():
    freq_dict2[city]={}
    for subset_name, subset_chains in city_chains.items():
        freq_dict2[city][subset_name]={}
        for cluster, cluster_data in subset_chains.items():
            for column, column_data in cluster_data.items():
                if column not in freq_dict2[city][subset_name]:
                    freq_dict2[city][subset_name][column]={}
                if            cluster            not            in
freq_dict2[city][subset_name][column]:
freq_dict2[city][subset_name][column][cluster] = {}
                for   feature,   value   in   column_data.items():
freq_dict2[city][subset_name][column][cluster][feature] = value
print(freq_dict2)
result_dict = {}
# Проходимо по містах
for city in freq_dict:
    result_dict[city] = {}
    # Проходимо по підмножинах
    for subset_name in freq_dict[city]:
        result_dict[city][subset_name] = {}
        # Проходимо по характеристиках (columns)
```

```
        for column in freq_dict[city][subset_name]:
            result_dict[city][subset_name][column] = {}
            # Проходимо по кластерах
            for cluster in freq_dict[city][subset_name][column]:
                result_dict[city][subset_name][column][cluster]
= {}
                # Отримуємо ознаки та значення для кластера з
freq_dict
                features_dict                                  =
freq_dict[city][subset_name][column][cluster]
                features_dict2                                 =
freq_dict2[city][subset_name][column][cluster]

                # Сума значень для нормалізації (freq_dict)
                total_2024 = sum(features_dict.values())

                # Знаходимо найбільшу ознаку (freq_dict)
                max_feature_2024       =       max(features_dict,
key=features_dict.get)
                max_value_2024 = features_dict[max_feature_2024]
                percent_2024    =    round((max_value_2024    /
total_2024) * 100, 2) if total_2024 > 0 else 0.0

                # Сума значень для нормалізації (freq_dict2)
                total_2034 = sum(features_dict2.values())

                # Знаходимо найбільшу ознаку (freq_dict2)
                max_feature_2034       =       max(features_dict2,
key=features_dict2.get)
                max_value_2034 = features_dict2[max_feature_2034]
                percent_2034    =    round((max_value_2034    /
total_2034) * 100, 2) if total_2034 > 0 else 0.0

                # Визначаємо відсоток для найбільшої ознаки з
freq_dict у freq_dict2
                percent_2024_in_2034                           =
round((features_dict2.get(max_feature_2024,  0)  /  total_2034)  *
100, 2) if total_2034 > 0 else 0.0

                result_dict[city][subset_name][column][cluster]
= {
                    "Поточна        найпоширеніша        ознака":
max_feature_2024,
                    "Значення  поточної  найпоширенішої  ознаки":
max_value_2024,
                    "Відсоток  поточної  найпоширенішої  ознаки":
percent_2024,
                    "Прогнозований       відсоток       поточної
найпоширенішої ознаки": percent_2024_in_2034,
                    "Прогнозована     найпоширеніша     ознака":
max_feature_2034,
                    "Відсоток    прогнозованої    найпоширенішої
ознаки": percent_2034
```

```
                }
def is_problematic(cluster_data, city):
    """
    Перевіряє, чи кластер є проблемним за певними характеристиками
    для максимальних ознак у 2024 та 2034 роках.
    Повертає True, якщо кластер проблемний.
    """
    problematic = False
    # Перевіряємо всі умови для кожної характеристики
    for column, data in cluster_data.items():
        if column == "ethnic_origin":
            if ((data["Поточна найпоширеніша ознака"] != "українець (-ка)" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
                (data["Прогнозована найпоширеніша ознака"] != "українець (-ка)" and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                    problematic = True
        elif column == "native_language":
            if ((data["Поточна найпоширеніша ознака"] != "українська" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
                (data["Прогнозована найпоширеніша ознака"] != "українська" and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                    problematic = True
        elif column == "ukrainian_language":
            if ((data["Поточна найпоширеніша ознака"] == "ні" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
                (data["Прогнозована найпоширеніша ознака"] == "ні" and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                    problematic = True
        elif column == "citizenship":
            if ((data["Поточна найпоширеніша ознака"] != "громадянство України" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
                (data["Прогнозована найпоширеніша ознака"] != "громадянство України" and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                    problematic = True
        elif column == "building_type":
            if ((data["Поточна найпоширеніша ознака"] == "житлового приміщення немає (безпритульний)" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
                (data["Прогнозована найпоширеніша ознака"] == "житлового приміщення немає (безпритульний)" and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                    problematic = True
        elif column == "ownership":
            if ((data["Поточна найпоширеніша ознака"] == "оренда (найм) у окремих громадян" and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
```

```
            (data["Прогнозована найпоширеніша ознака"] ==
"оренда (найм) у окремих громадян" and data["Відсоток
прогнозованої найпоширенішої ознаки"] > 60)):
                problematic = True
        elif column == "job_location_city":
            if ((data["Поточна найпоширеніша ознака"] != city
and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
            (data["Прогнозована найпоширеніша ознака"] != city
and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                problematic = True
        elif column == "job_location_country":
            if ((data["Поточна найпоширеніша ознака"] !=
"Україна" and data["Відсоток поточної найпоширенішої ознаки"] >
60) or
            (data["Прогнозована найпоширеніша ознака"] !=
"Україна" and data["Відсоток прогнозованої найпоширенішої
ознаки"] > 60)):
                problematic = True
        elif column == "job":
            if ((data["Поточна найпоширеніша ознака"] == "ні"
and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
            (data["Прогнозована найпоширеніша ознака"] == "ні"
and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                problematic = True
        elif column == "preschool":
            if ((data["Поточна найпоширеніша ознака"] == "ні"
and data["Відсоток поточної найпоширенішої ознаки"] > 60) or
            (data["Прогнозована найпоширеніша ознака"] == "ні"
and data["Відсоток прогнозованої найпоширенішої ознаки"] > 60)):
                problematic = True
        elif column == "education_level":
            if ((data["Поточна найпоширеніша ознака"] == "не маю
освіти" and data["Відсоток поточної найпоширенішої ознаки"] > 60)
or
            (data["Прогнозована найпоширеніша ознака"] == "не
маю освіти" and data["Відсоток прогнозованої найпоширенішої
ознаки"] > 60) or
            (data["Поточна найпоширеніша ознака"] == "початкова
(початкова загальна)" and data["Відсоток поточної найпоширенішої
ознаки"] > 60) or
            (data["Прогнозована найпоширеніша ознака"] ==
"початкова (початкова загальна)" and data["Відсоток
прогнозованої найпоширенішої ознаки"] > 60)):
                problematic = True
        elif column in ['walking', 'eyesight', 'hearing',
'cognitive', 'self_care', 'communication']:
            if ((data["Поточна найпоширеніша ознака"] != "Ні -
не відчуваю труднощів" and data["Відсоток поточної
найпоширенішої ознаки"] > 60) or
            (data["Прогнозована найпоширеніша ознака"] != "Ні -
не відчуваю труднощів" and data["Відсоток прогнозованої
найпоширенішої ознаки"] > 60)):
                problematic = True
```

```
    return problematic
def collect_problematic_clusters(result_dict):
    problematic_clusters = {}     # Створюємо словник для зберігання результатів
    for city, subsets in result_dict.items():
        # Додаємо місто до словника
        problematic_clusters[city] = {}
        for subset_name, columns in subsets.items():
            # Додаємо підмножину
            problematic_clusters[city][subset_name] = {}
            clusters = {}
            for column, column_data in columns.items():
                for cluster, data in column_data.items():
                    if cluster not in clusters:
                        clusters[cluster] = {}
                    clusters[cluster][column] = data
            # Перевіряємо проблемні кластери і додаємо їх до словника
            for cluster, cluster_data in clusters.items():
                is_problem = is_problematic(cluster_data, city) # Перевіряємо, чи кластер проблемний
                if is_problem:
                    problematic_clusters[city][subset_name][cluster] = {}  # Додаємо кластер в словник
                    for column, data in cluster_data.items():
                        problematic_clusters[city][subset_name][cluster][column] = {}
                        for key, value in data.items():
                            # Якщо поточна та прогнозована ознаки співпадають, не додаємо Прогнозований відсоток поточної найпоширенішої ознаки
                            if key == "Прогнозований відсоток поточної найпоширенішої ознаки" and data["Поточна найпоширеніша ознака"] == data["Прогнозована найпоширеніша ознака"]:
                                continue
                            problematic_clusters[city][subset_name][cluster][column][key] = value
    return problematic_clusters
# Викликаємо функцію і зберігаємо результат
problematic_clusters = collect_problematic_clusters(result_dict)
def describe_problematic_clusters(problematic_clusters):
    for city, subsets in problematic_clusters.items():
        # Якщо у місті немає проблемних кластерів
        if not subsets:
            print(f"У місті {city} не було зафіксовано соціальних груп, які потребують негайного втручання.")
            continue
        print(f"Місто: {city}")
        for subset_name, clusters in subsets.items():
            for cluster, cluster_data in clusters.items():
                print("У місті виявлено соціальну групу:")
                # Описуємо етнічне походження
                if "ethnic_origin" in cluster_data:
```

```
                    ethnicity_data                                  =
cluster_data["ethnic_origin"]
                    current_ethnicity  =  ethnicity_data["Поточна
найпоширеніша ознака"]
                    predicted_ethnicity                             =
ethnicity_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Дана соціальна група складається з
громадян  такого  етнічного  походження:  {current_ethnicity}.",
end="")
                    # Додаємо примітку про культурні центри
                    if  current_ethnicity  not  in  ["українець  (-
ка)", "росіянин (-ка)"]:
                        print(" які можуть потребувати створення
відповідних культурних центрів.", end="")
                    print()  # Завершуємо рядок
                    #  Додаємо  прогнозоване  значення  етнічного
походження
                    if current_ethnicity != predicted_ethnicity:
                        print(f"  Також варто звернути увагу на
громадян такого етнічного походження: {predicted_ethnicity}, "
                              "які,  згідно  з  прогнозами,  у
майбутньому можуть складати більшу частину даної соціальної
групи.")
                # Описуємо рідну мову
                if "native_language" in cluster_data:
                    language_data                                   =
cluster_data["native_language"]
                    current_language   =   language_data["Поточна
найпоширеніша ознака"]
                    predicted_language                              =
language_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Рідна мова даної соціальної групи
- {current_language}.", end="")
                    # Додаємо примітку про освіту на рідній мові
                    if  current_language  not  in  ["українська",
"російська"]:
                        print("  Дана  соціальна  група  може
потребувати підтримки освіти на рідній мові.", end="")
                    print()  # Завершуємо рядок
                    # Додаємо прогнозоване значення рідної мови
                    if current_language != predicted_language:
                        print(f"     Згідно  з  прогнозами,  у
майбутньому найпоширенішою рідною мовою у даній соціальній групі
може стати: {predicted_language}.")
                # Описуємо володіння українською мовою
                if "ukrainian_language" in cluster_data:
                    ukrainian_language_data                         =
cluster_data["ukrainian_language"]
                    current_ukrainian                               =
ukrainian_language_data["Поточна найпоширеніша ознака"]
                    predicted_ukrainian                             =
ukrainian_language_data["Прогнозована найпоширеніша ознака"]
```

```
                    print(f"  Володіння українською мовою даної соціальної групи - {current_ukrainian}.", end="")
                    if current_ukrainian == "ні":
                        print("  Дана соціальна група може потребувати організації курсів вивчення української мови для дорослих і дітей, "
                              "забезпечення доступу до перекладачів у лікарнях, соціальних службах та інших критичних установах, "
                              "та надання важливої інформації про здоров’я, освіту та працевлаштування на мові, зрозумілій цій частці населення.", end="")
                    print()  # Завершуємо рядок
                    if current_ukrainian != predicted_ukrainian:
                        print(f"  Володіння українською мовою у даній соціальній групі у майбутньому може змінитися на {predicted_ukrainian}.", end="")
                        if current_ukrainian == "так" and predicted_ukrainian == "ні":
                            print(" Це може свідчити про те, що українська мова потребує популяризації у даній соціальній групі.", end="")
                        print()  # Завершуємо рядок
                # Описуємо громадянство
                if "citizenship" in cluster_data:
                    citizenship_data = cluster_data["citizenship"]
                    current_citizenship = citizenship_data["Поточна найпоширеніша ознака"]
                    predicted_citizenship = citizenship_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Більшість громадян, які відносяться до даної соціальної групи, мають таке громадянство: {current_citizenship}.", end="")
                    # Додаємо примітки для різних типів громадянства
                    if current_citizenship == "громадянство іншої держави":
                        print("  Дана соціальна група може потребувати організації культурних та соціальних заходів для знайомства з місцевими традиціями та допомоги в адаптації до нового середовища.", end="")
                    elif current_citizenship == "особа без громадянства":
                        print("  Дана соціальна група може потребувати створення центрів для консультування та допомоги особам без громадянства.", end="")
                    elif current_citizenship == "громадянство не визначене":
                        print("  Дана соціальна група може потребувати допомоги в зборі та оформленні документів для визначення громадянства.", end="")
                    print()  # Завершуємо рядок
```

```
                    # Прогнозування зміни громадянства
                    if current_citizenship != predicted_citizenship:
                        print(f"  У майбутньому найпоширенішим громадянством у даній соціальній групі буде {predicted_citizenship}.", end="")
                        # Додаємо примітки для прогнозованих змін
                        if predicted_citizenship == "громадянство іншої держави":
                            print("  Варто попіклуватися про створення тимчасових та постійних житлових комплексів для новоприбулих.", end="")
                        elif predicted_citizenship == "особа без громадянства":
                            print("  Варто попіклуватися про створення програм для можливого надання громадянства або статусу постійного резидента.", end="")
                        elif predicted_citizenship == "громадянство не визначене":
                            print("  Варто попіклуватися про створення процедур для визнання осіб з невизначеним громадянством, забезпечення їхньої реєстрації та легалізації в державних органах.", end="")
                        print()  # Завершуємо рядок
                # Описуємо рівень освіти
                if "education_level" in cluster_data:
                    education_level_data = cluster_data["education_level"]
                    current_education_level = education_level_data["Поточна найпоширеніша ознака"]
                    predicted_education_level = education_level_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Більшість громадян, які відносяться до даної соціальної групи, мають такий рівень освіти: {current_education_level}.", end="")
                    # Додаємо примітку для "не маю освіти"
                    if current_education_level == "не маю освіти":
                        print(" Дана соціальна група потребує доступу до безкоштовної або доступної базової освіти для дітей та дорослих.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни рівня освіти
                    if current_education_level != predicted_education_level:
                        print(f"  Згідно з прогнозами, у майбутньому більшість громадян у даній соціальній групі матимуть такий рівень освіти: {predicted_education_level}.", end="")
                        # Додаємо примітки для "не маю освіти"
                        if predicted_education_level == "не маю освіти":
```

```
                            print("  Дана  соціальна  група  у
майбутньому може потребувати організації програм перепідготовки
для дорослих, зокрема для тих, хто не мав можливості здобути
освіту в молодому віці.", end="")
                        print()  # Завершуємо рядок
                # Описуємо тип закладу освіти
                if "education" in cluster_data:
                    education_data = cluster_data["education"]
                    current_education  =  education_data["Поточна
найпоширеніша ознака"]
                    predicted_education                          =
education_data["Прогнозована найпоширеніша ознака"]
                    print(f"   Отримання освіти та тип закладу
освіти, у якому навчається більшість громадян, які відносяться
до даної соціальної групи: {current_education}.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни типу закладу освіти
                    if current_education != predicted_education:
                        print(f"     Згідно  з  прогнозами,  у
майбутньому більшість громадян у даній соціальній групі
навчатимуться у таких закладах освіти: {predicted_education}.",
end="")
                        print()  # Завершуємо рядок
                # Описуємо відвідування дошкільних закладів
                if "preschool" in cluster_data:
                    preschool_data = cluster_data["preschool"]
                    current_preschool  =  preschool_data["Поточна
найпоширеніша ознака"]
                    predicted_preschool                          =
preschool_data["Прогнозована найпоширеніша ознака"]
                    if current_preschool in ["так", "ні"]:
                        print(f"  Відвідування більшості дітей,
які відносяться до даної соціальної групи, дошкільного закладу
освіти: {current_preschool}.", end="")
                        print()  # Завершуємо рядок
                        # Додаємо примітку для "ні"
                        if current_preschool == "ні":
                            print("  У  місті  варто  побудувати
нові або розширити наявні установи, щоб знизити навантаження на
наявні заклади.", end="")
                        print()  # Завершуємо рядок
                        # Прогнозування зміни відвідування
                        if          current_preschool          !=
predicted_preschool:
                            print(f"  У майбутньому відвідування
дітьми дошкільного закладу освіти зміниться таким чином:
{predicted_preschool}.", end="")
                            if predicted_preschool == "ні":
                                print("     У     місті     варто
потурбуватися про розширення наявних установ, щоб знизити
навантаження на наявні заклади у майбутньому.", end="")
                            print()  # Завершуємо рядок
                # Описуємо сферу діяльності
```

```
                if "job_category" in cluster_data:
                    job_category_data                                =
cluster_data["job_category"]
                    current_job_category                             =
job_category_data["Поточна найпоширеніша ознака"]
                    predicted_job_category                           =
job_category_data["Прогнозована найпоширеніша ознака"]
                    print(f"      Сфера   діяльності   більшості
громадян,   які   належать   до   даної   соціальної   групи:
{current_job_category}.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни сфери діяльності
                    if          current_job_category          !=
predicted_job_category:
                        print(f"  У майбутньому сфера діяльності
даної   соціальної   групи   може   змінитися   таким   чином:
{predicted_job_category}.", end="")
                        print()  # Завершуємо рядок
                # Описуємо наявність роботи
                if "job" in cluster_data:
                    job_data = cluster_data["job"]
                    current_job        =        job_data["Поточна
найпоширеніша ознака"]
                    predicted_job    =    job_data["Прогнозована
найпоширеніша ознака"]
                    print(f"   Наявність роботи впродовж тижня,
який передував даті перепису населення, з метою отримання оплати
або доходу (прибутку) у грошовому або натуральному вигляді,
даної соціальної групи: {current_job}.", end="")
                    if current_job == "ні":
                        print(" Дана соціальна група потребує
створення нових робочих місць.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування наявності роботи
                    if current_job != predicted_job:
                        print(f"   Прогноз для наявності роботи
для даної соціальної групи у майбутньому: {predicted_job}.",
end="")
                        if predicted_job == "ні":
                            print(" Дана соціальна група може
потребувати створення нових робочих місць у майбутньому.",
end="")
                        print()  # Завершуємо рядок
                # Описуємо місцезнаходження основної роботи
                if "job_location_country" in cluster_data:
                    job_location_country_data                        =
cluster_data["job_location_country"]
                    current_job_location_country                     =
job_location_country_data["Поточна найпоширеніша ознака"]
                    predicted_job_location_country                   =
job_location_country_data["Прогнозована найпоширеніша ознака"]
```

```
                    print(f"   Місцезнаходження основної роботи
(країна)            даної              соціальної              групи:
{current_job_location_country}.", end="")
                    if current_job_location_country != "Україна":
                        print(" Це може свідчити про недостатній
розвиток місцевої економіки, відсутність робочих місць у певних
галузях або обмеженість професійних можливостей у місті.",
end="")
                    print()  # Завершуємо рядок
                    #   Прогнозування   зміни   місцезнаходження
основної роботи
                    if      current_job_location_country      !=
predicted_job_location_country:
                        print(f"  У майбутньому місцезнаходження
основної роботи (країна) даної соціальної групи може змінитися
таким чином: {predicted_job_location_country}.", end="")
                        if    current_job_location_country    ==
"Україна":
                            print("   Варто   попіклуватися   про
розвиток місцевої економіки та наявність робочих місць у місті.",
end="")
                        print()  # Завершуємо рядок
                # Описуємо місцезнаходження основної роботи
(місто)
                if "job_location_city" in cluster_data:
                    job_location_city_data                        =
cluster_data["job_location_city"]
                    current_job_location_city                     =
job_location_city_data["Поточна найпоширеніша ознака"]
                    predicted_job_location_city                   =
job_location_city_data["Прогнозована найпоширеніша ознака"]
                    print(f"   Місцезнаходження основної роботи
(місто) даної соціальної групи: {current_job_location_city}.",
end="")
                    if current_job_location_city != city:
                        print(" Це може свідчити про недостатній
розвиток місцевої економіки, відсутність робочих місць у певних
галузях або обмеженість професійних можливостей у місті.",
end="")
                    print()  # Завершуємо рядок
                    #   Прогнозування   зміни   місцезнаходження
основної роботи
                    if        current_job_location_city       !=
predicted_job_location_city:
                        print(f"  У майбутньому місцезнаходження
основної роботи (місто) даної соціальної групи може змінитися
таким чином: {predicted_job_location_city}.", end="")
                        if current_job_location_city == city:
                            print("   Варто   попіклуватися   про
розвиток місцевої економіки та наявність робочих місць у місті.",
end="")
                        print()  # Завершуємо рядок
                # Описуємо причину, чому респонденти не працюють
```

```
                if "location_reason" in cluster_data:
                    location_reason_data                    =
cluster_data["location_reason"]
                    current_location_reason                 =
location_reason_data["Поточна найпоширеніша ознака"]
                    predicted_location_reason               =
location_reason_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Причина, через яку респонденти не
працюють     в     населеному     пункті,     де     проживають:
{current_location_reason}.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни причини
                    if         current_location_reason         !=
predicted_location_reason:
                        print(f"     Прогноз   на   майбутнє   для
причини, через яку респонденти не працюють в населеному пункті,
де проживають: {predicted_location_reason}.", end="")
                        print()  # Завершуємо рядок
                # Описуємо тип будівлі та житлового приміщення
                if "building_type" in cluster_data:
                    building_type_data                      =
cluster_data["building_type"]
                    current_building_type                   =
building_type_data["Поточна найпоширеніша ознака"]
                    predicted_building_type                 =
building_type_data["Прогнозована найпоширеніша ознака"]
                    print(f"      Тип   будівлі   та   житлового
приміщення   більшості   громадян,   що   відносяться   до   даної
соціальної групи: {current_building_type}.", end="")
                    # Додаткове повідомлення, якщо тип будівлі
"безпритульний"
                    if   current_building_type   ==   "житлового
приміщення немає (безпритульний)":
                        print(" Дана соціальна група потребує
розробки і реалізації програм соціального житла, яке може бути
надане тимчасово або на умовах оренди для осіб, які не мають
власного даху над головою. Пропонується використання незайнятих
будівель для розміщення тимчасових притулків.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни типу будівлі
                    if          current_building_type          !=
predicted_building_type:
                        print(f"  У майбутньому тип будівлі та
житлового приміщення більшості громадян, що відносяться до даної
соціальної     групи,     може     змінитися     таким     чином:
{predicted_building_type}.", end="")
                        if predicted_building_type == "житлового
приміщення немає (безпритульний)":
                            print("   Варто   попіклуватися   про
розробку програм соціального житла у місті.", end="")
                        print()  # Завершуємо рядок
                if "ownership" in cluster_data:
                    ownership_data = cluster_data["ownership"]
```

```
                    current_ownership = ownership_data["Поточна
найпоширеніша ознака"]
                    predicted_ownership                        =
ownership_data["Прогнозована найпоширеніша ознака"]
                    print(f"  Тип володіння житловим приміщенням
більшості громадян, що відносяться до даної соціальної групи:
{current_ownership}.", end="")
                    # Додаткове повідомлення, якщо тип володіння
"оренда (найм)"
                    if current_ownership == "оренда (найм) у
окремих громадян":
                        print(" Дана соціальна група потребує
створення спеціальних програм для забезпечення молоді та
студентів доступним житлом, зокрема будівництво гуртожитків або
пільгових квартир для молодих сімей.", end="")
                    print()  # Завершуємо рядок
                    # Прогнозування зміни типу володіння
                    if current_ownership != predicted_ownership:
                        print(f"   У майбутньому тип володіння
житловим приміщенням більшості громадян, що відносяться до даної
соціальної групи, може змінитися таким чином:
{predicted_ownership}.", end="")
                        if predicted_ownership == "оренда (найм)
у окремих громадян":
                            print("   Варто попіклуватися про
розробку програм соціального житла у місті.", end="")
                        print()  # Завершуємо рядок
                if 'walking' in cluster_data:
                    current_walking                            =
cluster_data['walking']['Поточна найпоширеніша ознака']
                    predicted_walking                          =
cluster_data['walking']['Прогнозована найпоширеніша ознака']
                    print(f"   Наявність проблем із ходінням у
громадян даної соціальної групи: {current_walking}.")
                    if current_walking != "Ні - не відчуваю
труднощів":
                        print(" Дана соціальна група потребує
облаштування пандусів, ліфтів, спеціальних доріжок та зручних
тротуарів для людей з обмеженими можливостями.")
                    if current_walking != predicted_walking:
                        print(f"     У майбутньому наявність
проблем із ходінням у громадян даної соціальної групи може
змінитися таким чином: {predicted_walking}.")
                        if predicted_walking != "Ні - не
відчуваю труднощів":
                            print("   Варто попіклуватися про
облаштування пандусів, ліфтів, спеціальних доріжок та зручних
тротуарів для людей з обмеженими можливостями.")
                if 'eyesight' in cluster_data:
                    current_eyesight                           =
cluster_data['eyesight']['Поточна найпоширеніша ознака']
                    predicted_eyesight                         =
cluster_data['eyesight']['Прогнозована найпоширеніша ознака']
```

```
                    print(f"    Наявність проблем із зором у
громадян даної соціальної групи: {current_eyesight}.")
                    if current_eyesight != "Ні - не відчуваю
труднощів":
                        print(" Дана соціальна група потребує
встановлення тактильних доріжок та спеціальних сигналів на
перехрестях для незрячих людей.")
                    if current_eyesight != predicted_eyesight:
                        print(f"      У майбутньому наявність
проблем із зором у громадян даної соціальної групи може
змінитися таким чином: {predicted_eyesight}.")
                        if predicted_eyesight != "Ні - не
відчуваю труднощів":
                            print("  Варто попіклуватися про
встановлення тактильних доріжок та спеціальних сигналів на
перехрестях для незрячих людей.")
                if 'hearing' in cluster_data:
                    current_hearing                          =
cluster_data['hearing']['Поточна найпоширеніша ознака']
                    predicted_hearing                        =
cluster_data['hearing']['Прогнозована найпоширеніша ознака']
                    print(f"    Наявність проблем із слухом у
громадян даної соціальної групи: {current_hearing}.")
                    if current_hearing != "Ні - не відчуваю
труднощів":
                        print(" Дана соціальна група потребує
забезпечення доступу до перекладачів жестовою мовою у всіх
публічних установах, встановлення в громадських місцях
спеціальних систем для візуальних або тактильних попереджень.")
                    if current_hearing != predicted_hearing:
                        print(f"      У майбутньому наявність
проблем із слухом у громадян даної соціальної групи може
змінитися таким чином: {predicted_hearing}.")
                        if predicted_hearing != "Ні - не
відчуваю труднощів":
                            print("  Варто попіклуватися про
забезпечення доступу до перекладачів жестовою мовою у всіх
публічних установах, встановлення в громадських місцях
спеціальних систем для візуальних або тактильних попереджень.")
                if 'cognitive' in cluster_data:
                    current_cognitive                        =
cluster_data['cognitive']['Поточна найпоширеніша ознака']
                    predicted_cognitive                      =
cluster_data['cognitive']['Прогнозована найпоширеніша ознака']
                    print(f"  Наявність проблем із когнітивними
здібностями у громадян даної соціальної групи:
{current_cognitive}.")
                    if current_cognitive != "Ні - не відчуваю
труднощів":
                        print(" Дана соціальна група потребує
послуг догляду на дому, забезпечення доступу до психологічної
підтримки, а також організація спеціальних центрів, де надаються
консультації і допомога в адаптації до соціуму.")
```

```
                    if current_cognitive != predicted_cognitive:
                        print(f"     У майбутньому наявність
проблем із когнітивними здібностями у громадян даної соціальної
групи може змінитися таким чином: {predicted_cognitive}.")
                        if predicted_cognitive != "Ні - не
відчуваю труднощів":
                            print(" Варто попіклуватися про
організацію спеціальних центрів, де надаються консультації і
допомога в адаптації до соціуму.")
                if 'self_care' in cluster_data:
                    current_self_care                          =
cluster_data['self_care']['Поточна найпоширеніша ознака']
                    predicted_self_care                        =
cluster_data['self_care']['Прогнозована найпоширеніша ознака']
                    print(f"  Наявність проблем із доглядом за
собою у громадян даної соціальної групи: {current_self_care}.")
                    if current_self_care != "Ні - не відчуваю
труднощів":
                        print(" Дана соціальна група потребує
розширення програм для людей, яким необхідна допомога вдома
(підтримка в приготуванні їжі, гігієнічних процедурах,
прибиранні).")
                    if current_self_care != predicted_self_care:
                        print(f"     У майбутньому наявність
проблем із доглядом за собою у громадян даної соціальної групи
може змінитися таким чином: {predicted_self_care}.")
                        if predicted_self_care != "Ні - не
відчуваю труднощів":
                            print(" Варто попіклуватися про
розширення програм для людей, яким необхідна допомога вдома.")
                if 'communication' in cluster_data:
                    current_communication                      =
cluster_data['communication']['Поточна найпоширеніша ознака']
                    predicted_communication                    =
cluster_data['communication']['Прогнозована найпоширеніша
ознака']
                    print(f"  Наявність проблем із спілкуванням
у громадян даної соціальної групи: {current_communication}.")
                    if current_communication != "Ні - не
відчуваю труднощів":
                        print(" Дана соціальна група потребує
проведення тренінгів для осіб, які мають труднощі з комунікацією,
що дозволяють їм полегшити вираження своїх думок та почуттів.")
                    if current_communication !=
predicted_communication:
                        print(f"     У майбутньому наявність
проблем із спілкуванням у громадян даної соціальної групи може
змінитися таким чином: {predicted_communication}.")
                        if predicted_communication != "Ні - не
відчуваю труднощів":
                            print(" Варто попіклуватися про
створення тренінгів для осіб, які мають труднощі з
комунікацією.")
```

ДОДАТОК Е

Звіт щодо добробуту та потреб населення

Місто: Одеса
У місті виявлено соціальну групу:
Дана соціальна група складається з громадян такого етнічного походження: молдаванин (-ка). які можуть потребувати створення відповідних культурних центрів.
Рідна мова даної соціальної групи - румунська. Дана соціальна група може потребувати підтримки освіти на рідній мові.
Згідно з прогнозами, у майбутньому найпоширенішою рідною мовою у даній соціальній групі може стати: німецька.
Володіння українською мовою даної соціальної групи - ні. Дана соціальна група може потребувати організації курсів вивчення української мови для дорослих і дітей, забезпечення доступу до перекладачів у лікарнях, соціальних службах та інших критичних установах, та надання важливої інформації про здоров'я, освіту та працевлаштування на мові, зрозумілій цій частці населення.
Більшість громадян, які відносяться до даної соціальної групи, мають таке громадянство: громадянство іншої держави. Дана соціальна група може потребувати організації культурних та соціальних заходів для знайомства з місцевими традиціями та допомоги в адаптації до нового середовища.

У місті виявлено соціальну групу:
Сфера діяльності більшості громадян, які належать до даної соціальної групи: Освіта, наука.
Місцезнаходження основної роботи (країна) даної соціальної групи: Італія. Це може свідчити про недостатній розвиток місцевої економіки, відсутність робочих місць у певних галузях або обмеженість професійних можливостей у місті.
У майбутньому місцезнаходження основної роботи (країна) даної соціальної групи може змінитися таким чином: Сполучені Штати Америки.

У місті виявлено соціальну групу:
Сфера діяльності більшості громадян, які належать до даної соціальної групи: Будівництво, архітектура.
Місцезнаходження основної роботи (країна) даної соціальної групи: Україна.
Місцезнаходження основної роботи (місто) даної соціальної групи: Ізмаїл. Це може свідчити про недостатній розвиток місцевої економіки, відсутність робочих місць у певних галузях або обмеженість професійних можливостей у місті.

У місті виявлено соціальну групу:

Сфера діяльності більшості громадян, які належать до даної соціальної групи: Будівництво, архітектура.

Місцезнаходження основної роботи (країна) даної соціальної групи: Україна.

Місцезнаходження основної роботи (місто) даної соціальної групи: Кілія. Це може свідчити про недостатній розвиток місцевої економіки, відсутність робочих місць у певних галузях або обмеженість професійних можливостей у місті.

У місті виявлено соціальну групу:

Сфера діяльності більшості громадян, які належать до даної соціальної групи: Освіта, наука.

Наявність роботи впродовж тижня, який передував даті перепису населення, з метою отримання оплати або доходу (прибутку) у грошовому або натуральному вигляді, даної соціальної групи: ні. Дана соціальна група потребує створення нових робочих місць.

У місті виявлено соціальну групу:

Більшість громадян, які відносяться до даної соціальної групи, мають такий рівень освіти: не маю освіти. Дана соціальна група потребує доступу до безкоштовної або доступної базової освіти для дітей та дорослих.

Отримання освіти та тип закладу освіти, у якому навчається більшість громадян, які відносяться до даної соціальної групи: не навчаюся.

У місті виявлено соціальну групу:

Більшість громадян, які відносяться до даної соціальної групи, мають такий рівень освіти: особа віком молодше 10 років.

Отримання освіти та тип закладу освіти, у якому навчається більшість громадян, які відносяться до даної соціальної групи: особа віком молодше 6 років.

Відвідування більшості дітей, які відносяться до даної соціальної групи, дошкільного закладу освіти: ні.

У місті варто побудувати нові або розширити наявні установи, щоб знизити навантаження на наявні заклади.