Харківський національний університет імені В.Н. Каразіна

Факультет математики і інформатики

Кафедра прикладної математики

Кваліфікаційна робота

бакалавр

на тему **«Визначення прихованих атрибутів користувачів соціальних мереж за допомогою соціального графа »**

Виконала: студентка групи МП41 IV курсу, (перший бакалаврський рівень), спеціальності 113 Прикладна математика **Каруник Н.Д.В.**

Керівник: старший викладач кафедри прикладної математики **Сузікова О.Г.**

Рецензент: кандидат фіз.-мат. наук, доцент кафедри прикладної математики **Стєпанова К.В.**

Харків — 2023 рік

Анотації

Каруник Н.Д.В. Визначення прихованих атрибутів користувачів соціальних мереж за допомогою соціального графа.

Кваліфікаційна робота бакалавра на тему "Визначення прихованих атрибутів користувачів соціальних мереж за допомогою соціального графа".

Кваліфікаційна робота складається із: вступної частини, де описані суть задачі, постановка, мотивація і план дослідження, предмет, об'єкт і аргументовано актуальність; основної частини, що складається з 4 розділів; перший розділ присвячений теорії передбачення прихованих атрибутів; другий розділ присвячений збору інформації для конкретного прикладу відпрацювання алгоритму; третій розділ присвячений програмній реалізації алгоритму; у четвертому розділі основної частини розглянуто та проаналізовано результати програми. Після основної частини підбиті підсумки дослідження, стверджується досягнення поставленої задачі.

Об'єм роботи: 38ст.

Основна частина: 32ст.

Karunyk N.D.V I.V. Defining hidden attributes of social media users using the social graph.

The text of your abstract in English. Qualification work of the bachelor on the topic "Defining hidden attributes of social media users using the social graph".

Qualification work consists of: the introductory part, which describes the essence of the problem, setting, motivation and research plan, subject, object and reasoned relevance; the main part, consisting of 4 sections; the first section is devoted to the theory of prediction of hidden attributes; the second section is devoted to collecting information for a specific example of algorithm development; the third section is devoted to the software implementation of the algorithm; in the fourth section of the main part, the results of the program are considered and analyzed. After the main part, the results of the study are summed up, the achievement of the task is confirmed.

Total: 38p.

Main part: 32p.

Зміст

Вступ

Чи знали ви, що 90 відсотків людей, що мають доступ до інтернету, є користувачами соціальних мереж? Якщо спочатку, це була нова галузь розваг, що своєю новизною, простотою і доступністю затягувала все більше людей у свої світи, то зараз соціальні мережі легко виступають у ролі платформ для бізнесу, об'єктів впливу на думки населення і знаряддям для забору величезної кількості інформації про суб'єктів, що так активно нею діляться. Користувачі надають про себе дані як явно, так і неявно, і якщо дехто думає, що не надаючи свою дату народження та іншу персональну інформацію, він попіклувався про свою конфіденційність — це не так.

Ключові слова, чек-іни, місця знаходження, дати, персональна інформація, фото, друзі, лайки, підписники та багато-багато іншого — те, що цілком можливо обробити та отримати нові ресурси. У цій роботі я зупинюся на аналізі друзів користувача. Чому? Деяка персональна інформація є доволі важливою для розробників тих самих соціальних мереж, застосунків, реклами та іншого контенту, який, наприклад, має бути направлений на певну цільову аудиторію. Проблема в тому, що не всі користувачі вводять ці дані, що перешкоджає дослідженню. Але чим замінити пусте поле, щоб не зупиняти аналіз? Його можна заповнити передбаченням. Приблизне значення, отримане у результаті прогнозу, вдало вписується в алгоритми. Але як саме робити цей прогноз?

«Скажи мені, хто твій друг, і я скажу, хто ти», — це не «пустий звук». Люди дружать із подібними собі, і саме це лягло в основу мого прогнозування. Друзі, яких я буду аналізувати та отримувати наближені значення деяких параметрів користувача за допомогою різних алгоритмів, а далі оцінювати їх валідність.

До того ж, оцінка алгоритма буде відбуватися не тільки за отриманими результатами, а ще і за його ефективністю та швидкістю. Чому? Моє дослідження базується на дослідженні користувачів соціальних мереж. Цільовим об'єктом може бути людина із величезною кількістю друзів, або великі проєкти можуть використовувати алгоритм для передбачення параметрів одночасно великої кількості цільових користувачів. Наприклад, база даних Facebook містить у собі дані про більш ніж мільярд користувачів, які у свою чергу встановили між собою більше 100 мільярдів зв'язків. І це момент, коли бажання та абстрактні цілі стикаються із обмеженими ресурсами сучасних машин.

Варто також зазначити, що алгоритм, представлений далі у роботі створений для передбачення значень певних полів, а не знаходження точного розв'язку. Тому отримані результати можуть бути умовно далеко від реальності, що може бути пов'язано із особливостями дружби людей, на яку я і спираюсь під час дослідження. Незважаючи на це, алгоритми декількох методів пошуку будуть відпрацьовані та порівняні між собою.

Перейду до постановки задачі. У цій роботі буде наведено 5 методів передбачення пропущених атрибутів користувачів соціальних мереж, у свою

чергу методи будуть уточнені для кількісних і номінативних характеристик, на конкретному прикладі ці методи будуть відпрацьовані: буде розглянутий певний користувач соціальної мережі Facebook, досліджені його зв'язки, складений соціальний граф для цього користувача, зображений для наочності, алгоритми будуть відпрацьовані за 2 параметрами: вік, місто проживання. Також буде наведено програмний код мовою Java, розглянуто його особливості та основні моменти. У кінці результати будуть порівняні між собою, буде зроблено висновки. Задача дослідження кваліфікаційної роботи поставлена. Задача у вигляді списку:

1. Теорія 5 методів передбачення атрибутів за допомогою соціального графа. Уточнення для номінативних характеристик.
2. Огляд даних конкретного користувача. Збір даних, побудова графу.
3. Програмування алгоритму. Особливості коду. Основні моменти.
4. Запуск програми. Оцінка ефективності. Аналіз результатів.
5. Висновки.

Предмет кваліфікаційної роботи: створення ефективного алгоритму визначення прихованих атрибутів користувачів соціальних мереж за допомогою соціального графа.

Об'єкт кваліфікаційної роботи: атрибути користувачів соціальних мереж та граф їх зв'язків.

Мета кваліфікаційної роботи: отримати значення прихованих атрибутів «Вік», «Місто проживання» користувача за допомогою створення соціального графа та його дослідження різними методами, порівняння цих методів

за ефективністю.

Актуальність кваліфікаційної роботи: актуальність визначена існуванням вище зазначеної проблеми сьогодні, популярністю соціальних мереж та необхідністю у створені більшої кількості алгоритмів. До того ж, дослідження має великі перспективи для розвитку у вигляді вдосконалення алгоритмів, програми та навіть підключення користувацького інтерфейсу для популяризації серед охочих користувачів.

Розділ 1. Теорія методів передбачення прихованих атрибутів користувачів соціальних мереж

У даному розділі наведено 5 методів визначення прихованих атрибутів соціального графа. Розділ розділено на параграфи. Перший присвячений методам для кількісних характеристик. Другий розглядає уточнення попередніх методів для номінативних показників.

1.1. Кількісні атрибути

У цій секції будуть розглянуті методи обчислення для атрибутів, які є кількісними. Тобто їх значення виражається у вигляді числа.

1.1.1. Метод 1: середнє значення всіх вершин

У соціальному графі користувача відсіюю вузли, що мають пусте значення атрибуту. Введу позначення

$$v_i$$

для значення цього атрибуту для і-го вузла. Тоді шукане значення атрибуту для цільового користувача буде визначатися за формулою:

$$M_i = \frac{\sum_{i=1}^{n^\star} v_i}{n^\star}$$

де

$$n^\star$$

— кількість відфільтрованих вузлів, у яких наявне значення атрибуту.

Увага, ремарка: далі фільтрація вузлів, які не мають відомого атрибута, відбуватиметься автоматично як природній етап обчислення.

1.1.2. Метод 2: середнє значення мод кластерів

Для цього методу першим етапом необхідно кластеризувати вузли графа. Після кластеризації буде отримано кластери:

$$C = C_1, ..., C_k, \sum_{i=1}^{k} C_i = n$$

Тут k — кількість отриманих кластерів, n — загальна кількість вузлів у графі. Наступним етапом відфільтрую кластери, які містять у собі менше трьох вузлів:

$$C^\star = \left\{ C_i || C_i | > 3, i = \overline{i, k} \right\},$$

$$|C^\star| = k^\star, k^\star \leq k.$$

У кожному виділеному кластері визначаю моду показника характеристики:

$$m_i = mode(C_i), C_i \in C^\star, i = \overline{1, k^\star}.$$

Після усіх етапів шуканий атрибут цільового користувача визначаю за формулою:

$$M_2 = \frac{\sum_{i=1}^{k^\star} m_i}{k^\star}.$$

1.1.3. Метод 3: мода найбільшого кластера

Даний метод схожий на попередній. Першим етапом знову кластеризую соціальний граф як у попередньому методі. Далі з усіх кластерів обираю найбільший. У випадку, коли у декількох кластерах міститься однаково найбільша кількість користувачів, наступний крок розповсюджується на кожний з них.

$$C_{max} = max|C_i|, C_i \in C, i = \overline{1,k}.$$

Наступним кроком буде знаходження моди знайденого найбільшого кластера (мод декількох найбільших кластерів). Тоді шуканий прихований атрибут знаходжу за формулою:

$$M_3 = \begin{cases} mode(C_{max}) & |C_{max}| = 1 \\ \dfrac{\sum_{i=1}^{k^\star} mode(C_i)}{k^\star} & |C_{max}| = k^\star, k^\star > 1 \end{cases}$$

До речі, на цьому методі можна проілюструвати соціальний аспект дружби та прийняття рішення стосовно прогнозування. За найбільший кластер відповідає найбільша група друзів користувачів, пов'язаних між собою. На життєвій практиці це може відповідати робочому або навчальному колективу, який найближче до користувача. Наприклад, якщо ми досліджуємо школяря, то таким кластером може виступити група його однокласників.

1.1.4. Метод 4: середнє кластера із найменшими викидами

Першим етапом кластеризую соціальний граф так само, як і в попередніх двох методах. Наступним кроком відсіюю кластери, що містять лише 1 вузол:

$$C^\star = \left\{C_i || C_i| > 1, i = \overline{i,k}\right\},$$

$$|C^\star| = k^\star, k^\star \leq k.$$

Після цього обчислюю дисперсію атрибута у кожному з отриманих кластерів.

$$\sigma_i = disp(C_i), C_i \in C^\star, i = \overline{1, k^\star}$$

Далі обираю кластер, показник дисперсії якого мінімальний. Що це означає? Це означає, що люди у цій групі мають найменше відхилення від середнього значення їх віку, тобто матимемо кластер, учасники якого належать приблизно до однієї соціальної групи. За сенсом це приблизно нагадує суть попереднього методу.

$$\sigma_{min} = min(\sigma_i), i = \overline{1, k^\star}$$

Останнім кроком визначаю прихований атрибут цільового користувача за наступною формулою:

$$M_4 = \frac{\sum_{i=1}^{n^\star_{\sigma_{min}}} v_{\sigma_{min_i}}}{n^\star_{\sigma_{min}}},$$

де

$$n^\star_{\sigma_{min}}$$

— число вузлів у обраному кластері.

1.1.5. Метод 5: коефіцієнти впливу

Першим етапом кластеризую соціальний граф так само, як і в попередніх двох методах. Наступним кроком відсіюю кластери, що містять лише 1 вузол. Обчислюю у кожному з отриманих кластерів дисперсію та середнє значення атрибута:

$$C^\star = \left\{C_i || C_i| > 1, i = \overline{i, k}\right\},$$

$$|C^\star| = k^\star, k^\star \leq k.$$

$$\sigma_i = disp(C_i),$$

$$a_i = avg(C_i),$$

$$C_i \in C^\star, i = \overline{1, k^\star}$$

Наступним етапом визначаю для кожного кластера коефіцієнт його впливу таким чином, щоб кластер із найбільшими викидами мав найменший вплив на результат, і навпаки. Коефіцієнти визначаються за наступними формулами:

$$\sigma = \sum_{i=1}^{k^\star} \sigma_i,$$

$$\sigma'_i = \sigma - \sigma_i,$$

$$\sigma' = \sum_{i=1}^{k^\star} \sigma'_i,$$

$$coeff_i = \frac{\sigma'_i}{\sigma'}.$$

Для перевірки правильності результату можна порівняти сумму всіх коефіцієнтів із одиницею. Після всіх обчислень підраховую значення атрибута цільового користувача за наступною формулою:

$$\sum_{i=1}^{k^\star} a_i coeff_i.$$

1.2. Номінативні атрибути

У цій секції будуть розглянуті аналоги методів з попереднього параграфу для атрибутів, що є номінативними. Це ті атрибути, значення яких виражається у вигляді рядка.

1.2.1. Підготовка атрибутів

Складаю список усіх значень номінативної зміної, які зустрічаються за даним атрибутом. Кожному значенню присвою індекс. Приклад:

Value	index
Kharkiv	0
Kyiv	1
...	...
Poltava	n

Тепер замість строкових даних атрибутів використовую чисельні шляхом заміни строкового виразу чисельним індексом.

1.2.2. Аналоги методів з першого параграфу для номінативних зміних

Підставляючи замість строкових значень індексні, я буду робити передбачення для атрибутів. Нижче наведу список методів із першого параграфу та уточнення, яке треба зробити, щоб адаптувати їх для номінативних зміних, або не використовувати взагалі:

Метод 1: Обчилюємо не середнє всіх вузлів, а моду:

$$M_{1_{nom}} = mode(v_i)$$

Метод 2: Обчислюємо не середнє мод кластерів, а моду мод. Якщо мод декілька, то результатом є декілька рівнозначних відповідей:

$$C = C_i, ..., C_k, \sum_{i=1}^{k} C_i = n,$$

$$C^\star = \left\{C_i || C_i| > 3, i = \overline{1,k}\right\},$$

$$|C^\star| = k^\star, k^\star \leq k.$$

Метод 3: Адаптація не потрібна.

Метод 4: Перший етап адаптації: замість дисперсії у кластері обчислюємо показник кількості відхилення вузлів у кластері від моди за наступною

формулою:

$$C^\star = \left\{C_i || C_i| > 1, i = \overline{1,k}\right\},$$

$$|C^\star| = k^\star, k^\star \leq k.$$

$$\delta_i = \frac{|v_i : v_i \neq mode(C_i)|}{|C_i|}, C_i \in C^\star, i = \overline{1,k^\star}.$$

$$\delta_{min} = min(\delta_i), i = \overline{1,k^\star}$$

Другий етап адаптації: замість середнього значення кластера з мінімальним показником обчислюємо моду:

$$M_{4_{nom}} = mode(C_{\delta_{min}}).$$

Метод 5: Метод не використовується для номінативних значень.

В результаті передбачення атрибутів за кожним із методів отримуємо індекс передбаченого атрибута. Останнім кроком є відновлення строкового значення атрибута за отриманим індексом.

Розділ 2. Огляд користувача. Збір даних. Побудова графу

Для проведення дослідження та побудови графа маю визначити цільового користувача. Також необхідно зібрати дані про його друзів, значення атрибутів. У наступному розділі на основі цих даних буде створено та відпрацьовано алгоритм.

Для моделювання алгоритму буде узято 2 атрибути: «Вік» і «Місто проживання». На основі передбачення значень цих атрибутів буде проведено дослідження.

Для дослідження я обрала соціальну мережу Facebook. Псевдовипадковим чином обираю користувача цієї соціальної мережі. Надто вже псевдовипадковим чином це буде автор цієї курсової роботи — Каруник Нестані Дея Володимирівна.

Обраний користувач

Профіль у сучасному Facebook надає можливості для налаштування таких атрибутів користувача:

1. Робота та досвід
2. Освіта (коледж чи університет і середня школа)
3. Поточне місце проживання
4. Рідне місто

5. Контактна інформація

6. Стать

7. День народження

8. Стосунки (шлюб)

9. Члени родини

10. Важливі події життя

11. Цікаві сторінки

Для мого дослідження фіксую атрибути: 4 — Рідне місто, 7 — День народження (Вік). Звичайно, для нашого цільового користувача усі ці атрибути є пустими. Для того, щоб можна було порівняти передбачені результати із поточними, складу таблицю:

Атрибут	Поточне значення
Рідне місто	Харків
Вік	22

Тепер складаю список друзів користувача, одразу ж вказую значення атрибутів:

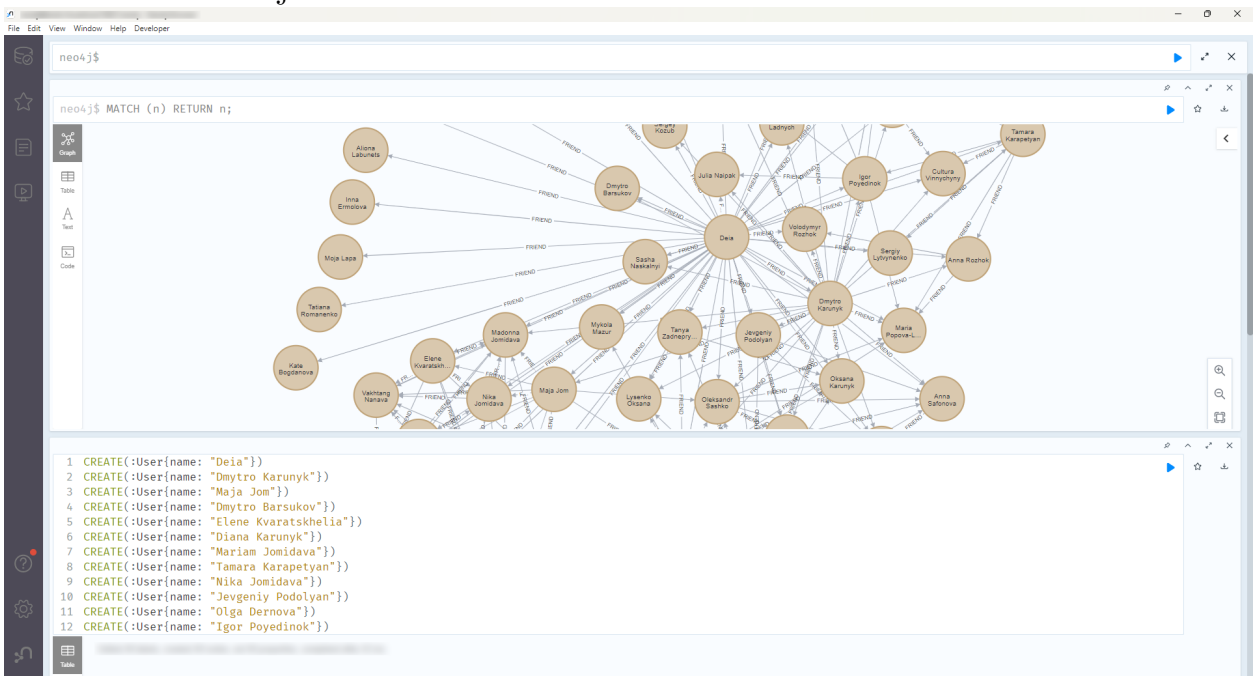
ID	Ім'я	Вік	Місто
0	Dmytro Karunyk	27	Харків
1	Maja Jom	53	Харків
2	Dmytro Barsukov	47	Харків
3	Elene Kvaratskhelia	12	Тбілісі
4	Diana Karunyk	52	Харків
5	Mariam Jomidava	16	Тбілісі
6	Tamara Karapetyan	43	Вінниця
7	Nika Jomidava	23	Тбілісі
8	Jevgeniy Podolyan	36	Харків
9	Olga Dernova	36	Вінниця
10	Igor Poyedinok	24	Харків
11	Volodymyr Anatoliyovych	49	Харків
12	Manoni Jomidava	32	Тбілісі
13	Tanya Zadnepryanskaya	32	Харків
14	Anna Rozhok	43	Вінниця
15	Oleksandr Sashko	50	Харків
16	Manuchar Jomidava	34	Тбілісі
17	Oksana Karunyk	44	Харків
18	Cultura Vinnychyny	56	Вінниця
19	Olga Mykolaivna	32	Вінниця
20	Mykola Mazur	62	Вінниця
21	Sergiy Lytvynenko	23	Харків
22	Vakhtang Nanava	64	Тбілісі
23	Maria Popova-Lelukh	34	Харків
24	Lysenko Oksana	26	Харків
25	Dmytro Markian	24	Харків
26	Ludmyla Karunyk	52	Харків
27	Natalia Monakova	22	Харків

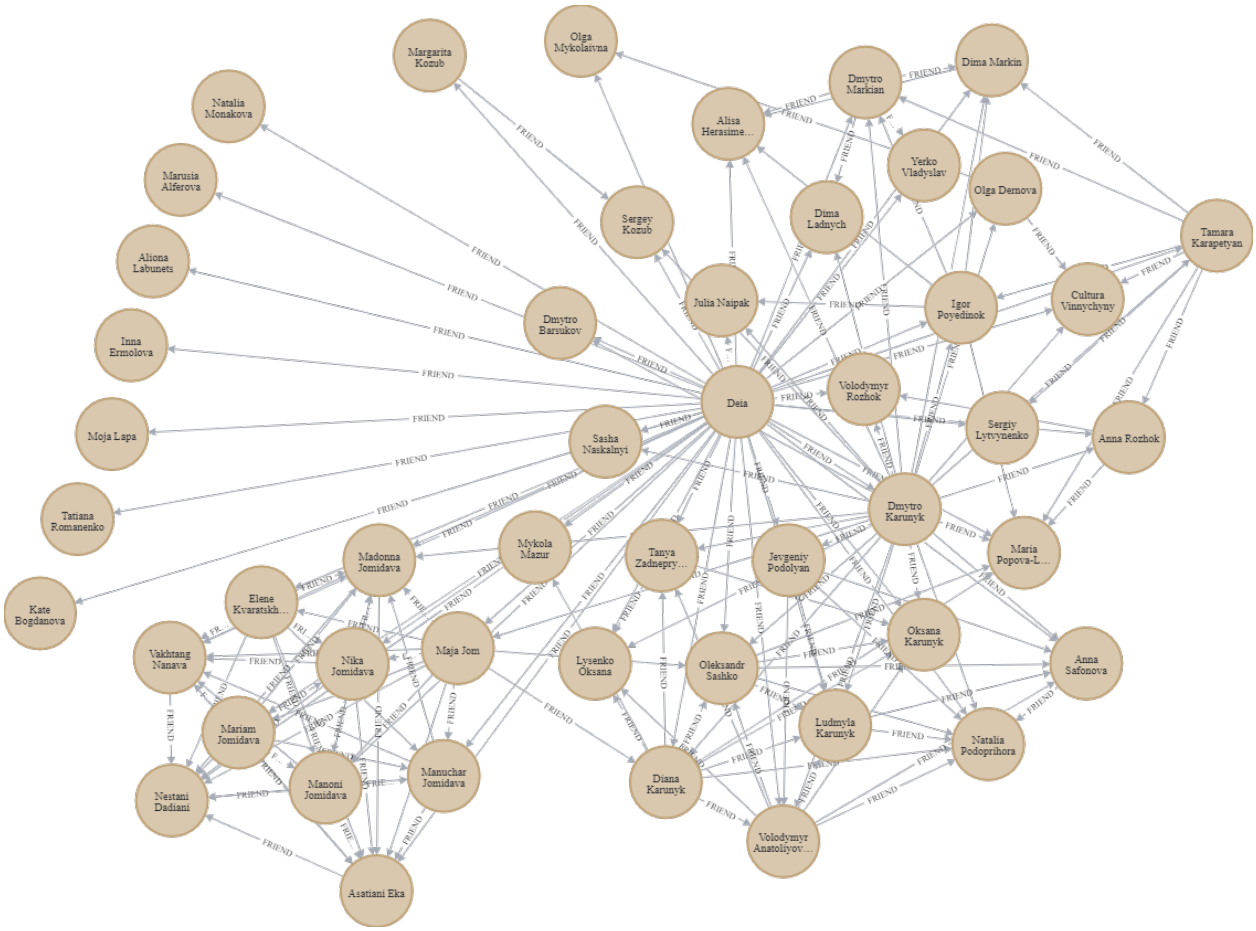
28	Anna Safonova	41	Харків
29	Dima Markin	24	Харків
30	Alisa Herasimenko	19	Київ
31	Sasha Naskalnyi	29	Вінниця
32	Aliona Labunets	28	Київ
33	Margarita Kozub	22	Харків
34	Kate Bogdanova	29	Харків
35	Marusia Alferova	23	Харків
36	Tatiana Romanenko	25	Вінниця
37	Inna Ermolova	27	Харків
38	Yerko Vladyslav	24	Харків
39	Sergey Kozub	48	Харків
40	Madonna Jomidava	37	Тбілісі
41	Moja Lapa	25	Вінниця
42	Volodymyr Rozhok	45	Вінниця
43	Dima Ladnych	21	Харків
44	Julia Naipak	36	Харків
45	Asatiani Eka	34	Тбілісі
46	Natalia Podoprihora	56	Харків
47	Nestani Dadiani	59	Тбілісі

Тепер складаю матрицю суміжності, що ілюструє зв'язки між друзями цільового користувача. Матриця є квадратною, розміру 48x48, симетричною, на головній діагоналі знаходяться нулі. Одиниці ілюструють наявність зв'язку, нулі — його відсутність.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0 Dmytro Karunyk	0	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	0	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0
1 Maja Jom	1	0	0	1	1	1	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
2 Dmytro Barsukov	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 Elene Kvaratskhelia	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
4 Diana Karunyk	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
5 Mariam Jomidava	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
6 Tamara Karapetyan	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 Nika Jomidava	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
8 Jevgeniy Podolyan	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9 Olga Dernova	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 Igor Poyedinok	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
11 Volodymyr Anatoliyovych	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12 Manoni Jomidava	0	1	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1
13 Tanya Zadneprvanskaya	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14 Anna Rozhok	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15 Oleksandr Sashko	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16 Manuchar Jomidava	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
17 Oksana Karunyk	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18 Cultura Vinnychyny	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19 Olga Mykolaivna	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20 Mykola Mazur	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21 Sergiy Lytvynenko	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22 Vakhtang Nanava	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
23 Maria Popova-Lelukh	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24 Lysenko Oksana	1	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25 Dmytro Markian	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
26 Ludmyla Karunyk	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
27 Natalia Monakova	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28 Anna Safonova	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
29 Dima Markin	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30 Alisa Herasimenko	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31 Sasha Naskalnyi	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32 Aliona Labunets	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33 Margarita Kozub	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
34 Kate Bogdanova	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35 Marusia Alferova	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36 Tatiana Romanenko	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37 Inna Ermolova	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38 Yerko Vladyslav	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39 Sergey Kozub	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40 Madonna Jomidava	1	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
41 Moja Lapa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42 Volodymyr Rozhok	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43 Dima Ladnych	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44 Julia Naipak	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45 Asatiani Eka	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
46 Natalia Podoprihora	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47 Nestani Dadiani	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Для наочності побудую граф, спираючись на матрицю суміжності у графовій системі Neo4j.





Граф побудовано, усі необхідні параметри зібрані. Таким чином, можу перейти до наступного, дуже важливого, етапу дослідження — програмування алгоритму. Із результатом автоматизації обчислень можна ознайомитися у наступному розділі.

Розділ 3. Програмування алгоритму

Нижче наведений проєкт програмної реалізації алгоритму із коментарями у необхідних місцях. Проєкт реалізовано мовою Java version 1.8.0-261 в інтегрованому середовищі розробки Intellij IDEA.

Варто звернути увагу на структуру проєкту:

1. Клас AbstractMath: для необхідних математичних обчислень, що зустрічатимуться під час роботи алгоритму, наприклад, обчислення моди кластеру.
2. Клас nominativePeregrine: для методів, що реалізовують 4 методи передбачення номінативних атрибутів.
3. Клас parametersPeregrine: для вхідних параметрів графу, наприклад, матриці параметрів.
4. Клас valuesPeregrine: для методів, що реалізовують 5 методів передбачення кількісних атрибутів.
5. Клас runPeregrine: основний клас для виклику усіх необхідних методів для обчислень.

Нижче наведена реалізація кожного із класів:

Перший лістинг.

Лістинг 3.1: AbstractMath

```
package org.main;

import static java.lang.StrictMath.pow;
import java.util.ArrayList;
import java.util.List;

public class AbstractMath {

  public static Integer mode(List<Integer> data) {
    int mode = 0, maxCount = 0;
    for (int i = 0; i < data.size(); i++) {
      int count = 0;
      for (int j = 0; j < data.size(); ++j) {
        if (data.get(i) == data.get(j))
          count++;
      }
      if (count > maxCount) {
        maxCount = count;
        mode = data.get(i);
      }
    }
    return mode;
  }
  public static Integer mode(int[] data) {
    int mode = 0, maxCount = 0;
    for (int i = 0; i < data.length; i++) {
      int count = 0;
      for (int j = 0; j < data.length; j++) {
        if (data[i] == data[j])
          count++;
      }
      if (count > maxCount) {
        maxCount = count;
```

```
        mode = data[i];
      }
    }
    return mode;
  }
  public static float average(List<Integer> data, int[][] matrix
      , int indexParameter) {
    float average = 0;
    for (int i = 0; i < data.size(); i++) average += matrix[
        indexParameter][i];
    return (float) average / data.size();
  }
  public static float average(int[] data, int[][] matrix, int
      indexParameter) {
    float average = 0;
    for (int i = 0; i < data.length; i++) average += matrix[
        indexParameter][i];
    return (float) average / data.length;
  }
  public static Integer totalMode(List<Integer> data) {
    Integer sumModes = 0;
    for (Integer i : data) sumModes += i;
    return sumModes;
  }
  public static float clusterDispersion(List<Integer> data) {
    float avg = 0;
    float dispersion = 0;
    for (Integer i : data) avg += i;
    avg = avg / data.size();
    for (Integer i : data) {
      float diff = (float) i - avg;
      dispersion += (float) pow(diff, 2);
    }
    return dispersion;
```

```
}
public static float clusterDispersion(int[] data) {
  float avg = 0;
  float dispersion = 0;
  for (int i : data) avg += i;
  avg = avg / data.length;
  for (int i : data) {
    float diff = (float) i - avg;
    dispersion += (float) pow(diff, 2);
  }
  return dispersion;
}
public static int[] clusterLeastDispersion(int[][] data) {
  List<int[]> dataProperSize = new ArrayList<>();
  for (int i = 0; i < data.length; i++) {
    if (data[i].length > 2) dataProperSize.add(data[i]);
  }
  List<Float> dispersions = new ArrayList<>();
  for (int i = 0; i < dataProperSize.size(); i++) {
    dispersions.add(clusterDispersion(dataProperSize.get(i)));
  }
  Float minDispersion = dispersions.get(0);
  int minDispersionIndex = 0;
  for (int i = 1; i < dispersions.size(); i++) {
    if (dispersions.get(i) < minDispersion) minDispersionIndex
        = i;
  }
  return data[minDispersionIndex];
}
public static float nominativeDispersion(int[] cluster) {
  int count = 0;
  int mode = mode(cluster);
  for (int i = 0; i < cluster.length; i++) {
    if (cluster[i] == mode) count++;
```

```
    }
    return (float) count / cluster.length;
  }
}
```

Тепер другий лістинг.

Лістинг 3.2: nominativePeregrine

```
package org.main;

import static org.main.AbstractMath.*;
import java.util.ArrayList;
import java.util.List;

public class nominativePeregrine {

  public static String value1MethodNom (int[][] matrixParameters
     , int indexParameter, String[] description) {
    return description[mode(matrixParameters[indexParameter])];
  }
  public static String value2MethodNom (int[][] matrixParameters
     , int[][] clusters, int indexParameter, String[]
     description) {
    List<Integer> modes = new ArrayList<>();
    for (int i = 0; i < clusters.length; i++) {
      if (clusters[i].length >= 3) {
        List<Integer> data = new ArrayList<>();
        for (int j = 0; j < clusters[i].length; j++) {
          data.add(matrixParameters[indexParameter][clusters[i][
             j]]);
        }
        modes.add(mode(data));
      }
    }
    return description[mode(modes)];
```

```
}
public static String value3MethodNom(int[][] matrixParameters,
    int[][] clusters, int indexParameter, String[] description
   ) {
  int maxClusterSize = clusters[0].length;
  int maxClusterIndex = 0;
  List<Integer> data = new ArrayList<>();
  for (int i = 1; i < clusters.length; i++) {
    if (clusters[i].length > maxClusterSize) maxClusterIndex =
        i;
  }
  for (int i = 0; i < clusters[maxClusterIndex].length; i++) {
    data.add(matrixParameters[indexParameter][clusters[
       maxClusterIndex][i]]);
  }
  return description[mode(data)];
}
public static String value4MethodNom(int[][] matrixParameters,
    int[][] clusters, int indexParameter, String[] description
   ) {
  List<Float> dispersions = new ArrayList<>();
  for (int i = 0; i < clusters.length; i++) {
    dispersions.add(nominativeDispersion(clusters[i]));
  }
  float minDispersion = dispersions.get(0);
  int minDispIndex = 0;
  for (int i = 0; i < dispersions.size(); i++){
    if (dispersions.get(i) < minDispersion){
      minDispIndex = i;
      minDispersion = dispersions.get(i);
    }
  }
  List<Integer> data = new ArrayList<>();
  for (int i = 0; i < clusters[minDispIndex].length; i++) {
```

```
        data.add(matrixParameters[indexParameter][clusters[
            minDispIndex][i]]);
      }
      return description[mode(data)];
    }
}
```

I, нарешті, третій.

Лістинг 3.3: parametersPeregrine

```
package org.main;

public class parametersPeregrine {
  public static final int[][][] matrixParameters = new int[][][] {
      {27, 53, 47, 12, 52, 16, 43, 23, 36, 36, 24, 49, 32, 32,
         43, 50, 34, 44, 56, 32, 62, 23, 64, 34, 26, 24, 52, 22,
          41, 24, 19, 29, 28, 22, 29, 23, 25, 27, 24, 48, 37,
         25, 45, 21, 36, 34, 56, 59},
      {0, 0, 0, 1, 0, 1, 2, 1, 0, 2, 0, 0, 1, 0, 2, 0, 1, 0, 2,
         2, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 2, 3, 0, 0, 0, 2,
         0, 0, 0, 1, 2, 2, 0, 0, 1, 0, 1}
  };
  public static final String[] nominativeDescript = new String[]
      {
    "Kharkiv", "Tbilisi", "Vinnytsia", "Kyiv"
  };
  public static final int[][][] clusters = new int[][][] {
      {2},
      {27},
      {32},
      {34},
      {35},
      {36},
      {37},
      {41},
```

```
        {1, 3, 5, 7, 12, 16, 22, 40, 45, 47},
        {0, 4, 6, 8, 9, 10, 11, 13, 14, 15, 17, 18, 19, 20, 21,
           23, 24, 25, 26, 28, 29, 30, 31, 33, 38, 39, 42, 43, 44,
            46},
    };
}
```

I, нарешті, третій.

Лістинг 3.4: valuesPeregrine

```
package org.main;

import static org.main.AbstractMath.average;
import static org.main.AbstractMath.clusterDispersion;
import static org.main.AbstractMath.clusterLeastDispersion;
import static org.main.AbstractMath.mode;
import static org.main.AbstractMath.totalMode;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.List;

public class valuesPeregrine {

    public static String value1Method (int[][] matrixParameters,
        int indexParameter) {
        int totalValue = 0;
        for (int i : matrixParameters[indexParameter]) totalValue +=
            i;
        return String.valueOf(new BigDecimal(totalValue).divide(new
            BigDecimal(matrixParameters[0].length), 0,
             RoundingMode.HALF_DOWN));
    }
    public static String value2Method (int[][] matrixParameters,
        int[][] clusters, int indexParameter) {
```

```
    List<Integer> modes = new ArrayList<>();
    for (int i = 0; i < clusters.length; i++) {
      if (clusters[i].length >= 3) {
        List<Integer> data = new ArrayList<>();
        for (int j = 0; j < clusters[i].length; j++) {
          data.add(matrixParameters[indexParameter][clusters[i][
              j]]);
        }
        modes.add(mode(data));
      }
    }
    return String.valueOf(new BigDecimal(totalMode(modes)).
       divide(new BigDecimal(modes.size()), 0, RoundingMode.
       HALF_DOWN));
  }
  public static String value3Method(int[][] matrixParameters,
     int[][] clusters, int indexParameter) {
    int maxClusterSize = clusters[0].length;
    int maxClusterIndex = 0;
    List<Integer> data = new ArrayList<>();
    for (int i = 1; i < clusters.length; i++) {
      if (clusters[i].length > maxClusterSize) maxClusterIndex =
          i;
    }
    for (int i = 0; i < clusters[maxClusterIndex].length; i++) {
      data.add(matrixParameters[indexParameter][clusters[
         maxClusterIndex][i]]);
    }
    return String.valueOf(new BigDecimal(mode(data)));
  }
  public static String value4Method(int[][] matrixParameters,
     int[][] clusters, int indexParameter) {
    int[] clusterMinDispersion = clusterLeastDispersion(clusters
       );
```

```
    List<Integer> data = new ArrayList<>();
    for (int i = 0; i < clusterMinDispersion.length; i++) {
      data.add(matrixParameters[indexParameter][
          clusterMinDispersion[i]]);
    }
    return String.valueOf(new BigDecimal(mode(data)));
  }
  public static String value5Method(int[][] matrixParameters,
     int[][] clusters, int indexParameter) {
    List<Float> dispersions = new ArrayList<>();
    List<Float> averages = new ArrayList<>();
    Float prognozedValue = (float) 0;
    float totalDispersion = 0;
    for (int[] c : clusters){
      dispersions.add(clusterDispersion(c));
      totalDispersion += clusterDispersion(c);

      averages.add(average(c, matrixParameters, indexParameter))
          ;
    }
    for (int i = 0; i < dispersions.size(); i++) {
      dispersions.set(i, dispersions.get(i)/totalDispersion);
    }
    for (int i = 0; i < dispersions.size(); i++) {
      prognozedValue += averages.get(i) * dispersions.get(i);
    }
    return String.valueOf(new BigDecimal(prognozedValue).
        intValue());
  }
}
```

І, нарешті, третій.

Лістинг 3.5: runPeregrine

```
package org.main;
```

```
import static org.main.nominativePeregrine.*;
import static org.main.parametersPeregrine.*;
import static org.main.valuesPeregrine.*;


public class runPeregrine {
  public static void main(String[] args) {
    System.out.println("Age predicted");
    System.out.println("Method 1: " + value1Method(
       matrixParameters, 0));
    System.out.println("Method 2: " + value2Method(
       matrixParameters, clusters,0));
    System.out.println("Method 3: " + value3Method(
       matrixParameters, clusters,0));
    System.out.println("Method 4: " + value4Method(
       matrixParameters, clusters, 0));
    System.out.println("Method 5: " + value5Method(
       matrixParameters, clusters, 0));
    System.out.println("City predicted");
    System.out.println("Method 1: " + value1MethodNom(
       matrixParameters, 1, nominativeDescript));
    System.out.println("Method 2: " + value2MethodNom(
       matrixParameters, clusters,1, nominativeDescript));
    System.out.println("Method 3: " + value3MethodNom(
       matrixParameters, clusters,1, nominativeDescript));
    System.out.println("Method 4: " + value4MethodNom(
       matrixParameters, clusters, 1, nominativeDescript));
  }
}
```

Розділ 4. Запуск програми. Аналіз результатів

У результаті відпрацювання програми було отримано наступні результати: передбачений вік користувача п'ятьма методами, а також передбачене місто проживання чотирма методами.

Лістинг 4.1: Output

```
Age predicted
Method 1: 35
Method 2: 29
Method 3: 24
Method 4: 47
Method 5: 36
City predicted
Method 1: Kharkiv
Method 2: Tbilisi
Method 3: Kharkiv
Method 4: Kharkiv
```

Проаналізую результати, що я отримала, визначивши відхилення від фактичного віку для кількісного атрибуту. Для номінативного атрибуту вкажу, чи співпав отриманий результат із фактичним значенням цільового користувача.

Аналіз точності передбачення атрибуту "Вік":

Метод	Поточне значення	Результат	Відхилення
1	22	35	13
2	22	29	7
3	22	24	2
4	22	47	25
5	22	36	14

Отримала такі результати, спираючись на які, роблю висновок, що для кількісного атрибуту найбільш точним методом передбачення є третій метод — визначення за модою найбільшого кластера.

Аналіз точності передбачення атрибуту "Місто":

Метод	Поточне значення	Результат	Співпадіння
1	Харків	Харків	true
2	Харків	Тбілісі	false
3	Харків	Харків	true
4	Харків	Харків	true

Я отримала такі результати, отже, 3 із 4 методів дали результат, що співпадає із фактичним.

Таким чином, я виконала поставлені перед собою задачі, підсумки дослідження підбиті у наступному розділі кваліфікаційної роботи.

Висновки

Кваліфікаційна робота була присвячена проблемі визначення прихованих атрибутів користувачів соціальних мереж. Для ілюстрації дослідження було зафіксовано соціальну мережу Facebook і конкретного користувача Каруник Нестані Дею Володимирівну.

На основі інформації про друзів цільового користувача було складено матрицю суміжності та матрицю із атрибутами користувачів двох типів: кількісного та номінативного. У кваліфікаційній роботі наведено 5 методів передбачення атрибутів для кількісних атрибутів і адаптації 4 з них для номінативних атрибутів:

1. Середнє значення всіх вершин (з адаптацією)
2. Середнє значення мод кластерів (з адаптацією)
3. Мода найбільшого кластера (з адаптацією)
4. Середнє значення кластера із найменшими викидами (з адаптацією)
5. Коефіцієнти впливу

Наступним етапом було запрограмовано всі п'ять методів для кількісних атрибутів і 4 для номінативних, розглянуто код програми, досліджено ефективність.

У результаті мною були отримані такі результати:

Для атрибуту "Вік":

Метод	Поточне значення	Результат	Відхилення
1	22	35	13
2	22	29	7
3	22	24	2
4	22	47	25
5	22	36	14

Для атрибуту "Місто":

Метод	Поточне значення	Результат	Співпадіння
1	Харків	Харків	true
2	Харків	Тбілісі	false
3	Харків	Харків	true
4	Харків	Харків	true

Найбільш точним для кількісного атрибуту виявився метод визначення за модою найбільшого кластера. Для номінативного атрибуту точний результат дали всі методи, крім визначення за модою мод кластерів.

На початку дослідження переді мною були поставлені задачі розв'язати основні проблеми. Це були проблема адаптації методів для номінативних атрибутів, автоматизації відпрацювання алгоритму програмно. Проблеми були розв'язані, що дає змогу піти у моєму дослідженні далі: не обмежуватися кількісними атрибутами, а також збільшити об'єми вхідних даних і працювати над покращенням їх швидкості, точності та ефективності передбачення. Це визначає широкі перспективи розвитку мого дослідження, якими я обов'язково планую зайнятися.

Список використаних джерел

[1] https://www.javatpoint.com/java-tutorial

[2] R. J. Wilson. Introduction to Graph Theory, 2007.

[3] Кузьменко І.М. Теорія графів. Київ. КПІ ім. Ігоря Сікорського, 2020.

[4] Jure Leskovec, Julian J. Mcauley Learning to Discover Social Circles in Ego Networks // Advances in Neural Information Processing Systems 25. 2012. P. 548 -556.

[5] Мазуренко В.В., Штовба С.Д. Огляд моделей аналізу соціальних мереж. Вінниця. ВНТУ, 2015.