МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

«Програмний модуль аналізу скарг на фінансові послуги компаній»

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 41

__ Козак В. В.
(прізвище та ініціали)

Керівник__ Кіктєв М. О.
(прізвище та ініціали)
Кандидат технічних наук, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол №_4__від_05.06.2023р.
зав. кафедри_____доц. Іларіонов О.Є.

Київ - 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри
інтелектуальних
технологій

Іларіонов О.Є.

_____

"___"_____20__ р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Козаку Владиславу Васильовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Програмний модуль аналізу скарг на фінансові послуги компаній

затверджена протоколом засідання кафедри від « 11 » листопада 2022 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2023 року

3. Вихідні дані до проекту (роботи)

Програмний застосунок, що надає змогу користувачу кластеризувати дані про скарги користувачів на фінансові послуги компаній та отримати розмічені файли для зручного подальшого аналізу та використання при формуванні стратегії боротьби з проблеми в середині цих продуктів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Належить зробити аналіз предметної області, проаналізувати можливі шляхи вирішення проблеми та актуальність проблеми, знайти відповідні дані для виконання роботи та провести аналіз та обробку цих даних, розробити функціональну модель та провести її декомпозицію для більш детального розуміння процесу розробки, розробити життєвий цикл розробки системи та архітектуру інформаційної системи, проаналізувати існуючі дослідження в даній темі, визначити математичні підходи, що будуть використані при вирішенні задачі, визначити всі типи вимог до програмного застосунку, продемонструвати приклад роботи розробленого застосунку та проаналізувати результати його роботи.

_____

_____

_____

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Необхідно зробити презентацію для демонстрації проробленої роботи, в якій будеть коротко описані всі перелічені в 4 пункті питання, також в ході презентації наочно продемонструвати роботу розробленого програмного застосунку та функціоналу, що той надає користувачу.

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Вступ	Кіктєв	,17.02.2023	, 17.02.2023
Розділ 1	Кіктєв	, 28.02.2023	, 28.02.2023
Розділ 2	Кіктєв	,29.03.2023	, 29.03.2023
Розділ 3	Кіктєв	,27.04.2023	,27.04.2023
Висновок	Кіктєв	,16.05.2023	,16.05.2023
Додатки	Кіктєв	,01.06.2023	,01.06.2023

7. Дата видачі завдання 16 лютого 2023 року

Керівник_____ _____/Кіктєв М.О./

Завдання прийняв до виконання /Козак В.В./

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
	І процентування	16.02.2023 – 01.03.2023	
	ІІ процентування	01.04.2023 – 10.04.2023	
	ІІІ процентування	01.05.2023 – 10.05.2023	
	Передзахист	19.05.2023 – 28.05.2023	
	Перевірка готових дипломних робіт на плагіат.	30.05.2023 – 06.06.2023	Після перевірки на плагіат правки у роботу вносити не можна.
	Здача готових дипломних робіт на кафедру.	07.06.2023 – 11.06.2023	● підписана керівником та студентом кваліфікаційна робота. ● підписаний відгук керівника ● підписана рецензія ● архів з програмним продуктом ● презентація доповіді

Студент-дипломник __ __/ Козак В. В./

Керівник випускної кваліфікаційної роботи _____ ___/ Кіктєв. М. О. /

Анотація

Козак Владислав Васильович виконав випускну кваліфікаційну роботу на тему «Програмний модуль аналізу скарг на фінансові послуги компаній» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз математичних методів кластеризації, розроблено інформаційне та програмне забезпечення, що виконує дозволяє проводити зручну кластеризацію даних, бачити візуальне представлення результатів та зберігати в зручну структуру.

Ключові слова: кластер, кластеризація, параметр, змінна, відстань, скарга, проблема, продукт.

Summary

Kozak Vladyslav Vasyliovych completed the graduation qualification work on the topic "Software module for the analysis of complaints on financial services of companies" in the specialty 122 - "Computer Science".

In the final qualification work, an analysis of mathematical methods of clustering was carried out, information and software was developed, which allows you to carry out convenient clustering of data, see a visual representation of the results and store them in a convenient structure.

Key words: cluster, clustering, parameter, variable, distance, complaint, problem, product.

Зміст

ВСТУП

Задача визначення класів скарг на фінансові послуги компаній, є дуже важливою на теперішній момент, так як дозволяє сильно зменшити втрати заробітку для компанії і не допустити банкрутства. В даній роботі, буде розглянуто набір даних зі скаргами на фінансові послуги та проведено аналіз, кластеризацію, щоб спробувати розробити стратегію запобігання втрати клієнтів. Самі скарги представляють з себе просто набір слів. Інтуїтивно можна сказати, що просто проблеми які виникають частіше всіх, є основними і мають перший пріоритет на виправлення, але задача є більш комплексною, так як скарги стосуються не просто невдоволення користувачів, а проблем з продуктами компанії, тому простою статистикою точно не обійдешся. Кластеризація якраз покликана виправити цю проблему, вона здатна відповісти на питання, які проблеми в яких ситуаціях частіше всього зустрічаються, за рахунок гнучкого оперування часовими проміжками надходження проблем, можна робити висновки про строки виправлення проблем. Робота покликана створити програмний застосунок, який може визначати центри цих проблем, базуючись на великих наборах реально існуючих даних, що були зібрані різноманітними компаніями. На поточний момент реальних робіт на тему аналізу скарг на фінансові послуги дуже мало, комерційні компанії не афішують результати своїх досліджень цієї теми та використовують свої наробки виключно в своїх закритих цілях. Однією з цілей цієї роботи, є спроба показати можливі підходи до аналізу скарг користувачів, що в свою чергу може призвести до покращення підходів в аналізі великих наборів даних, що мають дуже великий вплив на суспільство в загальному, бо фінансова сфера затримок чи помилок не любить.

Розділ 1. АНАЛІТИЧНИЙ ОГЛЯД, ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальна характеристика предметної області

Темою даної роботи буде дослідження скарг на фінансові послуги, вважаю доречним взяти саме дані про великі американські компанії, так як спектр можливих послуг є набагато більшим ніж у наших компаній, які в силу купи різних чинників, ще не мають такої бази різноманітних продуктів. Проте вважаю цю роботу корисною насамперед для нашої країни так як дослідження проблем конкурентів, своєчасне їх усунення та усвідомлення всіх ризиків, що вони за собою ведуть, в подальшому надасть прекрасну можливість та час на розробку стратегії та алгоритму дій, задля збереження положення компанії на ринку.

Аналіз скарг щодо фінансових послуг є критично важливим аспектом забезпечення виконання постачальниками фінансових послуг своїх юридичних зобов'язань і надання високоякісних послуг своїм клієнтам. Аналіз скарг може допомогти виявити проблемні питання та призвести до покращення надання послуг, що може підвищити задоволеність клієнтів і сприяти довірі до фінансової системи. У цьому розділі буде розглянуто ключові питання, пов'язані з аналізом скарг на фінансові послуги.

Важливим питанням є роль контролюючих органів у аналізі скарг. Регуляторні органи відповідають за нагляд за постачальниками фінансових послуг і за дотриманням ними відповідних законів і правил. Скарги є одним із ключових джерел інформації, які регулярно можуть використовувати для виявлення потенційних проблем і вжиття відповідних заходів. Однак регулятори повинні мати ефективні системи розгляду скарг, щоб збирати, аналізувати та реагувати на скарги своєчасно та ефективно.

Інше питання – важливість аналізу скарг для виявлення системних проблем. Скарги можуть надати цінну інформацію про моделі поведінки чи патерни, які можуть завдати шкоди споживачам. Аналізуючи скарги, постачальники фінансових послуг можуть визначити сфери, які потрібно покращити, і вжити заходів для

виправлення, щоб запобігти повторенню подібних проблем у майбутньому. Це також може допомогти зменшити ризик судових позовів і шкоди репутації.

Ефективний аналіз скарг також вимагає використання відповідних інструментів і методів аналізу даних. Постачальники фінансових послуг можуть використовувати великі дані та алгоритми машинного навчання для виявлення моделей і тенденцій у скаргах клієнтів. Це може допомогти виявити основні причини проблем і розробити більш цілеспрямовані рішення. Наприклад, якщо певний продукт або послуга викликає велику кількість скарг, постачальники можуть використовувати аналіз даних, щоб визначити основні проблеми та внести зміни в продукт або послугу для їх вирішення.

Підсумовуючи, аналіз скарг щодо фінансових послуг має вирішальне значення для покращення надання послуг, підвищення рівня задоволеності клієнтів і підтримки довіри до фінансової системи. Ефективний аналіз вимагає використання відповідної нормативно-правової бази, ефективних систем обробки скарг, інструментів і методів аналізу даних. Аналізуючи скарги, постачальники фінансових послуг можуть визначити сфери, які потрібно покращити, і вжити заходів для виправлення, щоб запобігти виникненню подібних проблем у майбутньому.

1.2 Аналіз актуальності досліджень в обраній області

Корисно було б зазначити актуальність дослідження даної теми на поточний момент. Я вважаю, що тема кластерного аналізу скарг на фінансові послуги дуже актуальна, особливо в сучасну епоху цифрових технологій, коли споживачі мають кілька каналів, щоб висловити свої скарги та думки щодо фінансових послуг.

Кластерний аналіз скарг може допомогти фінансовим установам визначити найпоширеніші та часті скарги та основні проблеми, які їх викликають. Аналізуючи скарги споживачів, пов’язані з банківськими послугами, такими як депозитні рахунки, позики, кредитні картки та інші, фінансові установи можуть отримати уявлення про уподобання клієнтів і проблемні точки. Ця інформація може допомогти

їм розробити цілеспрямовані стратегії для розгляду скарг клієнтів і підвищення їхньої задоволеності.

Крім того, у зв'язку зі збільшенням уваги до клієнтського досвіду та відповідності нормативним вимогам, фінансові установи повинні серйозно ставитися до скарг клієнтів і ефективно їх розглядати. Кластерний аналіз скарг може допомогти їм визначити закономірності та тенденції, а також дати змогу приймати керовані даними рішення для покращення своїх продуктів і послуг.

Таким чином, кластерний аналіз скарг на фінансові послуги є актуальним, оскільки він надає фінансовим установам цінну інформацію про скарги клієнтів, дозволяючи їм розробляти цільові стратегії для підвищення рівня задоволеності та утримання клієнтів, покращення їхньої репутації та дотримання нормативних вимог.

1.3 Огляд існуючих досліджень обраної предметної області

Аналізуючи предметну область, важливим буде дослідити інші роботи в даній темі. Загалом зараз аналіз скарг на фінансові послуги є добре відомим напрямком досліджень, якому в останні роки приділяється значна увага. Він передбачає застосування різноманітних методів інтелектуального аналізу даних і машинного навчання для виявлення закономірностей і тенденцій у даних скарг, а також для отримання цінної інформації, яка може стати основою для прийняття рішень і підвищити рівень задоволеності клієнтів.

Однією з недавніх робіт на цю тему є «Clustering Bank Customer Complaints on Social Media for Analytical CRM via Multi-objective Particle Swarm Optimization»[14] — це дослідницька стаття, опублікована в Journal of Ambient Intelligence and Humanized Computing у 2018 році. Дослідження провели Махді Бахтіарванд і Мар'ям Есмаїлі.

У дослідженні пропонується новий підхід до кластеризації скарг клієнтів банку в соціальних мережах за допомогою алгоритму оптимізації роєм частинок з різними об'єктами (MOPSO). Мета дослідження полягає в тому, щоб визначити

основні теми та питання, які хвилюють клієнтів банків, а також надати банковим структурам інформацію про те, як покращити рівень обслуговування та задоволеності клієнтів.

Автори застосували запропонований метод до набору даних про скарги клієнтів на платформі соціальних мереж Twitter і оцінили продуктивність алгоритму, використовуючи різні показники кластеризації, такі як чистота, ентропія та оцінка F1. Дослідження показало, що підхід кластеризації на основі MOPSO перевершує інші алгоритми кластеризації з точки зору якості та ефективності кластеризації.

Це дослідження надає корисну основу для аналізу скарг клієнтів у соціальних мережах і покращення управління взаємовідносинами з клієнтами в банківській галузі.

Іншою помітною роботою є «Customer Complaints Analysis Using Text Mining and Outcome-Driven Innovation Method for Market-Oriented Product Development»[12] — це дослідницька стаття або дослідження, яке зосереджується на використанні методів інтелектуального аналізу тексту та інновацій, орієнтованих на результат (ODI) для аналізу скарг клієнтів і розвитку ринку -орієнтовані продукти.

Ця праця має на меті допомогти компаніям удосконалити процес розробки продукту шляхом розуміння скарг і відгуків клієнтів. Він поєднує два методи: інтелектуальний аналіз тексту та інновації, орієнтовані на результат. Інтелектуальний аналіз тексту – це техніка, яка використовується для вилучення ідей і шаблонів із неструктурованих даних, таких як скарги клієнтів. З іншого боку, інновації, орієнтовані на результат, — це метод, який використовується для визначення бажаних результатів або результатів, які клієнти хочуть отримати від продукту.

В роботі ці методи використовуються для аналізу скарг клієнтів, пов’язаних із конкретним продуктом, визначення основних потреб і бажань клієнтів і розробки нового продукту, який відповідає цим потребам. Результатом є ринково-орієнтований продукт, розроблений для задоволення потреб і бажань клієнтів і підвищення загальної задоволеності клієнтів.

В цьому дослідженні підкреслює важливість прислухатися до відгуків клієнтів і використовувати інноваційні методи для розробки орієнтованих на ринок продуктів, які відповідають їхнім потребам і бажанням.

В 2018 році вийшла ще одна цікава робота «Data Analysis of Consumer Complaints in Banking Industry using Hybrid Clustering»[13] — це дослідницька стаття або дослідження, яке зосереджується на аналізі скарг споживачів у банківській галузі за допомогою методів гібридної кластеризації.

Дане дослідження має на меті допомогти банкам і фінансовим установам визначити основні причини скарг клієнтів і розробити стратегії для покращення обслуговування клієнтів. Для досягнення цієї мети в дослідженні поєднуються дві методики кластеризації: кластеризація K-середніх та ієрархічна кластеризація.

Кластеризація K-середніх — це метод, який використовується для поділу точок даних на групи або кластери на основі їх подібності. З іншого боку, ієрархічна кластеризація — це метод, який використовується для створення ієрархії кластерів шляхом ітеративного злиття або поділу кластерів на основі їх подібності.

Дослідження використовує ці два методи для групування скарг споживачів, пов’язаних із банківськими послугами, такими як депозитні рахунки, позики, кредитні картки та інші. Об’єднуючи схожі скарги разом, дослідження може виявити найпоширеніші та часті скарги, а також основні проблеми, які їх викликають.

В даній роботі також аналізують настрої скарг споживачів за допомогою методів обробки природної мови (NLP), щоб визначити рівень незадоволеності клієнтів. Завдяки поєднанню аналізу настроїв із методами кластеризації це дослідження може надати банкам повне розуміння скарг і відгуків клієнтів.

В цій роботі підкреслюється важливість аналізу даних і методів кластеризації для виявлення основних причин скарг клієнтів і покращення обслуговування клієнтів у банківській галузі. Підхід гібридної кластеризації, який використовується в дослідженні, може допомогти банкам розробити цільові стратегії для розгляду скарг клієнтів і підвищення рівня задоволеності клієнтів.

Загалом ці та інші дослідження підкреслюють важливість аналізу даних про скарги клієнтів у сфері фінансових послуг, а також потенційні переваги використання передових методів інтелектуального аналізу даних і машинного навчання для отримання цінної інформації та інформування для прийняття рішень.

1.4 Мета дослідження предметної області

На кінець метою кластерного аналізу скарг щодо фінансових послуг є виявлення шаблонів і груп скарг клієнтів, пов’язаних із фінансовими послугами, такими як банківські послуги, кредити, кредитні картки та інші. Аналіз має на меті виявити основні проблеми та першопричини скарг клієнтів, що може допомогти фінансовим установам:

Підвищення рівня задоволеності клієнтів: визначаючи найпоширеніші та часті скарги, фінансові установи можуть розробити цільові стратегії для їх вирішення та підвищення рівня задоволеності клієнтів.

Розширення пропозицій продуктів і послуг: кластерний аналіз скарг може надати розуміння уподобань клієнтів і проблемних точок, що може допомогти фінансовим установам розробити нові продукти та послуги, які краще відповідають потребам клієнтів.

Зменшення операційних витрат: визначивши основні причини скарг, фінансові установи можуть оптимізувати свої процеси та операції, зменшивши витрати, пов’язані з вирішенням скарг клієнтів.

Забезпечення відповідності нормативними вимогами: Регуляторні органи часто вимагають від фінансових установ звітувати та аналізувати скарги клієнтів. Кластерний аналіз скарг може допомогти фінансовим установам дотримуватися нормативних вимог і уникнути штрафних санкцій.

Остаточна мета кластерного аналізу скарг на фінансові послуги полягає в тому, щоб надати фінансовим установам цінну інформацію про скарги та відгуки

клієнтів, дозволяючи їм приймати керовані даними рішення для підвищення рівня задоволеності клієнтів, підвищення їх репутації та дотримання нормативних вимог.

1.5 Постановка задачі

Проблема кластерного аналізу скарг на фінансові послуги передбачає виявлення закономірностей у скаргах, які отримує організація фінансових послуг від своїх клієнтів. Мета полягає в тому, щоб згрупувати скарги в кластери або категорії на основі подібності їх змісту та контексту. Цей процес може допомогти організації зрозуміти природу скарг, визначити основні причини та вжити відповідних заходів для їх усунення.

Щоб сформулювати задачу кластерного аналізу скарг на фінансові послуги, можна почати з таких питань:

- Які основні категорії скарг отримує організація?
- Які загальні теми чи проблеми виникають у цих скаргах?
- Які основні причини цих скарг, як-от погане обслуговування клієнтів, помилки виставлення рахунків або проблеми з продуктом?
- Як можна згрупувати скарги в кластери або категорії на основі цих подібностей?

Проблему кластерного аналізу скарг на фінансові послуги можна сформулювати так: маючи набір даних скарг, отриманих організацією фінансових послуг, метою є використання алгоритмів кластеризації для ідентифікації груп подібних скарг на основі їх змісту та контексту. Потім ці кластери можна використовувати для визначення загальних тем і проблем у скаргах, а також основних причин. Це може допомогти організації покращити свої продукти та послуги, вирішити проблеми клієнтів і запобігти майбутнім скаргам.

Проблема аналізу скарг на фінансові послуги може бути вирішена за допомогою різних математичних методів і прийомів, зокрема:

- Алгоритми кластеризації: алгоритми кластеризації використовуються для групування подібних скарг на основі їхніх характеристик і атрибутів, таких як ключові слова, теми та почуття. Приклади алгоритмів кластеризації включають кластеризацію K-середніх, ієрархічну кластеризацію та кластеризацію на основі щільності.

- Методи обробки природної мови (NLP): методи NLP використовуються для отримання значущої інформації з неструктурованих даних, таких як текстові дані у скаргах клієнтів. Техніки НЛП включають техніки очищення тексту, токенізації, стемінгу та позначення частини мови.

- Аналіз настроїв. Аналіз настроїв — це техніка, яка визначає емоційний тон скарг клієнтів, наприклад позитивний, негативний або нейтральний. Аналіз настрою можна виконати за допомогою алгоритмів машинного навчання або систем на основі правил.

- Моделювання тем: моделювання тем – це техніка, яка використовується для визначення тем або тем, які часто обговорюються в скаргах клієнтів. Тематичне моделювання можна виконувати за допомогою імовірнісних моделей, таких як прихований розподіл Діріхле (LDA).

- Аналіз мережі. Аналіз мережі – це техніка, яка використовується для виявлення взаємозв’язків між скаргами, наприклад ключовими словами чи темами, що зустрічаються одночасно. Аналіз мережі можна використовувати для визначення кластерів пов’язаних скарг і розуміння загальної структури даних скарги.

Ці математичні методи та прийоми можна комбінувати та застосовувати різними способами для вирішення проблеми аналізу скарг щодо фінансових послуг. Наприклад, алгоритми кластеризації можуть бути використані для групування подібних скарг, а потім для кожного кластера можна виконати аналіз настроїв, щоб визначити емоційний тон скарг. Потім можна застосувати тематичне моделювання для визначення загальних тем або тем у кожному кластері. Нарешті, мережевий

аналіз можна використовувати для виявлення зв’язків між кластерами та розуміння загальної структури даних скарги.

В даній роботі буде детально розглянуто саме перший варіант вирішення проблеми за допомогою кластерного аналізу, буде використано декілька алгоритмів кластеризації, що будуть визначені після детального розгляду даних, на основі якого і буде сформована множина алгоритмів, що можливо використати для вирішення даної задачі. Після чого буде проведено детальне порівняння результатів з подальшим удосконаленням обраних підходів для покращення результатів роботи програмного застосунку.

Наступним кроком, треба визначити якою має бути вхідна інформація програмного застосунку, що реалізує кластерний аналіз обраної предметної області. На мою думку вхідна інформація, необхідна для програми, яка реалізує кластерний аналіз скарг щодо фінансових послуг, включатиме наступне:

1. Дані про скарги клієнтів: це включатиме тексти скарг клієнтів, які можуть надходити з різних джерел, таких як журнали кол-центру, повідомлення електронною поштою, платформи соціальних мереж і журнали онлайн-чату.
2. Інформація про клієнта: це включатиме основну інформацію про клієнта, як-от ім’я, адресу електронної пошти, номер телефону та інформацію про обліковий запис.
3. Інформація про фінансовий продукт або послугу: це включатиме інформацію про конкретний фінансовий продукт або послугу, пов’язану з кожною скаргою, як-от кредитні картки, позики, іпотечні кредити чи інвестиції.
4. Інформація про мітку часу: це включатиме час і дату кожної скарги, що дозволить програмі аналізувати скарги з часом і визначати тенденції та закономірності.
5. Інформація про категоризацію: це включатиме будь-яку існуючу категоризацію або маркування скарг, які має фінансова установа, наприклад тип продукту, тип послуги або рівень серйозності.

6. Галузева та нормативна інформація: це включатиме будь-які відповідні галузеві чи нормативні стандарти, вказівки чи вимоги, яких повинна дотримуватися фінансова установа, як-от правила боротьби з відмиванням грошей, закони про конфіденційність даних або закони про захист прав споживачів.

7. Будь-які додаткові відповідні дані: це включатиме будь-які інші дані, які можуть мати відношення до аналізу скарг клієнтів, наприклад демографічні дані, дані про місцезнаходження або дані про транзакції.

Конкретна вхідна інформація, необхідна для програми, може відрізнятися залежно від потреб і цілей фінансової установи та конкретного аналітичного підходу, який використовується. Однак наведена вище інформація буде основною інформацією, необхідною для ефективного аналізу скарг клієнтів у сфері фінансових послуг. Єдиний пункт з переліку, який я б зазначив як опціональний, це інформація про клієнта, так як ці дані не є публічними і використовувати їх в дослідницькій роботі як мінімум не етично, а в реальності ж, публічні датасети надані компаніями чи іншими установами, не будуть включати в себе розгорнуту інформацію про клієнта, як от номер телефону, чи адреса електронної пошти, тому цей пункт є важливим для комерційних рішень, що мають на меті враховувати всю можливу інформацію про клієнтів.

Вихідною інформацією застосунку в даному випадку буде сам результат навчання моделі, згідно налаштованих параметрів, сформованих центроїдів чи будь-чого іншого, на основі чого працює алгоритм, можна буде робити висновки про загальне положення справ з скаргами. Також результатом буде розмічений файл з мітками кластерів та дрібніші файли з даними по конкретним кластерам. Надалі, маючи навчену модель, можна на основі її обрахунку провести аналіз саме кластерів, тому результатом роботи можна вважати файл, з детальною аналітикою обрахованих кластерів, що буде надходити до менеджерів чи адміністратора, що буде на основі отриманих даних, розробляти стратегії запобігання та вирішення проблем.

1.6 Вимоги до програмного застосунку

Не зайвим буде сформувати вимоги до програмного застосунку. Ось кілька прикладів системних, функціональних і нефункціональних вимог до програми, яка реалізує кластерний аналіз скарг щодо фінансових послуг:

Системні вимоги:

- Програма має бути розроблена для обробки великої кількості скарг клієнтів, пов’язаних із фінансовими послугами.
- Програма повинна мати можливість використовувати дані з багатьох джерел, таких як кол-центри, журнали онлайн-чату та платформи соціальних мереж.
- Програма має бути розроблена таким чином, щоб підтримувати масштабованість, щоб враховувати збільшення обсягів даних у міру зростання кількості клієнтів і скарг.

Функціональні вимоги:

- Програма повинна мати можливість групувати схожі скарги разом на основі їхніх характеристик, таких як ключові слова, теми та настрої.
- Програма повинна мати змогу працювати окремо з даними по кожній компанії окремо, використовуючи всі доступні поля.
- Програма повинна мати змогу виводити графіки частот по знайденим кластерам.
- Програма має підтримувати вибір компанії та алгоритму обробки даних по обраній компанії.
- Програма має надавати можливість користувачеві досліджувати, функцію похибки, задля визначення оптимальної кількості кластерів та виводити графіки користувачеві, задля зручної обробки інформації.

- Необхідно забезпечити можливість зміни теми застосунку, задля комфортної роботи в будь-який час доби.
- Необхідно забезпечити функцію збереження результатів в зрозумілу структуру для подальшого аналізу та обробки.
- Програма повинна мати можливість виконувати аналіз настроїв щодо скарг клієнтів, щоб визначити емоційний тон скарг.
- Додаток повинен мати можливість виконувати тематичне моделювання для визначення загальних тем або тем у кожному кластері скарг.

Нефункціональні вимоги:

- Додаток має бути зручним і легким у навігації для аналітиків і зацікавлених сторін.
- Програма має бути безпечною та захищати конфіденційні дані клієнтів.
- Додаток має бути надійним і доступним 24/7, щоб забезпечити оперативне реагування на скарги.
- Програма має бути розроблена таким чином, щоб її можна було легко підтримувати та оновлювати, підтримувати постійні вдосконалення та оновлення.

Розділ 2. ПРОЕКТНІ РІШЕННЯ ПРИ СТВОРЕННІ ЗАСТОСУНКУ

2.1 Дерево функції застосунку для кластеризації

Задля кращого розуміння того, як саме має виглядати майбутній за стосунок, початковою задачею буде визначити який саме необхідний набір функцій має бути присутнім. Дерево функцій є гарним вибором для зрозумілого загального представлення цього набору.

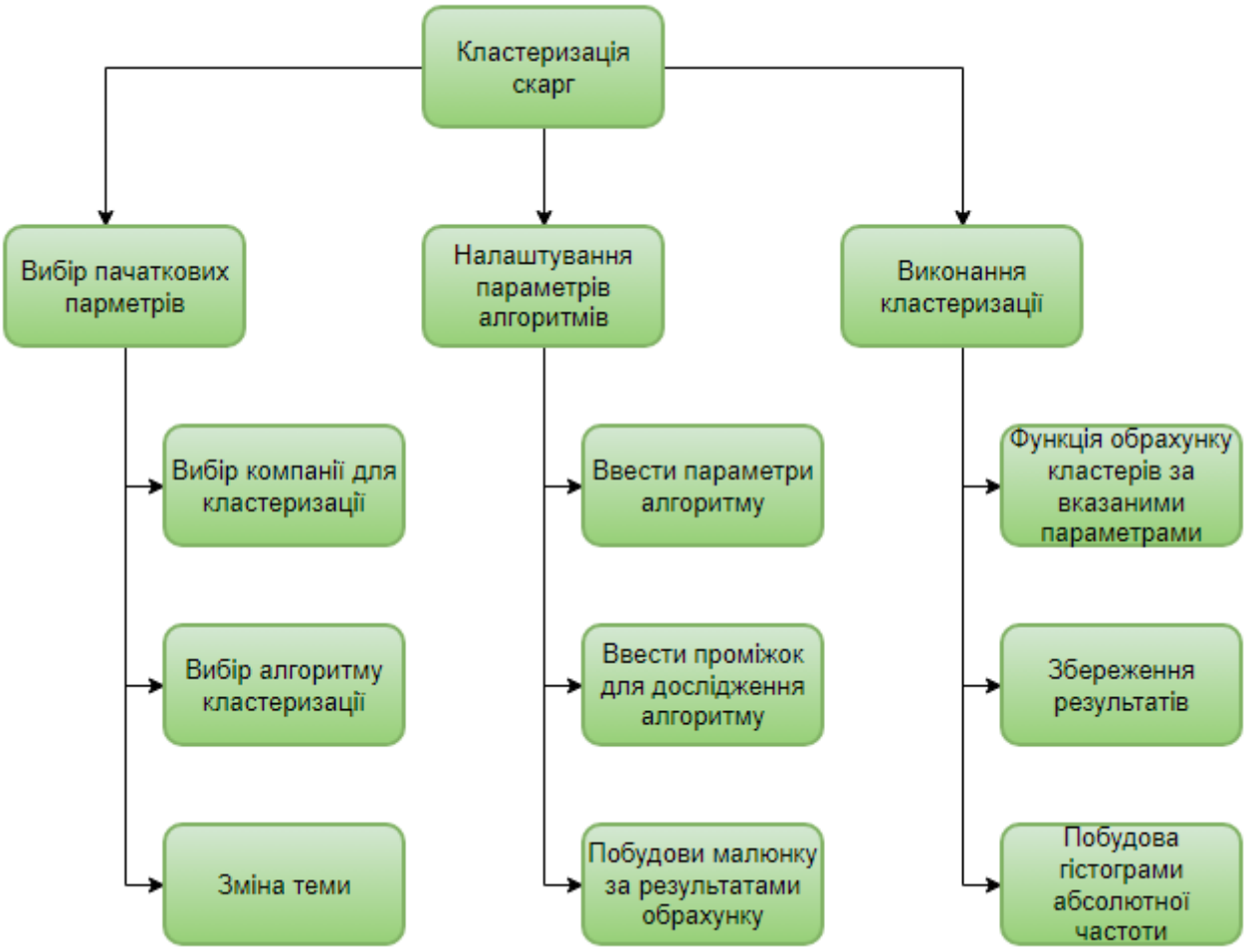

Рисунок 2.1 - Дерево функцій

Необхідні функції за стосунку були розділи на наступні класи:

1. Функції для введення початкових параметрів: вибір компанії дані про яку необхідно кластеризувати, вибір алгоритму кластеризації, вибір теми за стосунку.

2. Налаштування параметрів обраного алгоритму: ввести параметри алгоритму, ввести межі дослідження алгоритму, побудувати графіки за результатами обрахунків.

3. Виконання кластеризації: обрахунок кластерів, збереження результатів, побудова частотних гістограм.

Представлене дерево функцій є поверхневим представленням необхідного функціоналу для комфортної та повноцінної роботи застосунку. Зокрема самі функції будуть містити купу різних реалізацій під кожен обраний алгоритм зі своєю купою тонкощів.

2.2 Функціональна модель системи

Функціональна модель служить схемою для розробки та впровадження моделі кластеризації для скарг на фінансові послуги. У ній описано ключові функції та процеси, задіяні в кластеризації скарг, щоб отримати цінну інформацію та підвищити рівень задоволеності клієнтів у галузі фінансових послуг.

У цьому розділі буде представлено вичерпний огляд функціональної моделі, яка охоплює основні дії та компоненти, необхідні для ефективного проведення кластеризації. Функціональна модель діє як міст між цілями високого рівня системи та детальними аспектами впровадження.

Рисунок 2.2 - Функціональна модель

Для нашої задачі функціональна модель нотації IDEF0 буде мати вигляд як запропоновано на рисунку 2.2. Це загальний вид моделі, що потребує декомпозиції та виокремлення підпроцесів, що необхідно реалізувати.

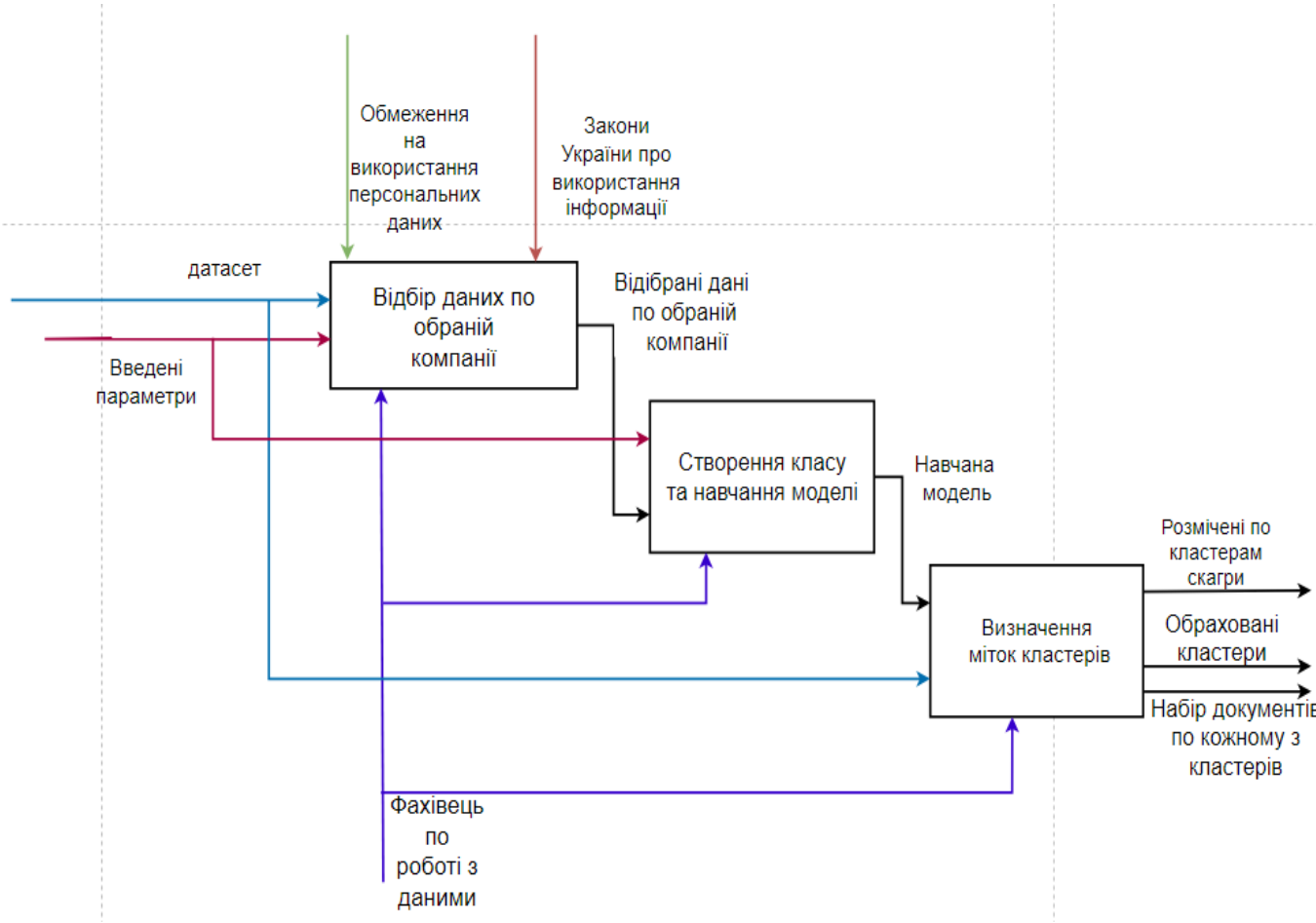

Рисунок 2.3 - Декомпозиція первинної функціональної моделі

Декомпозиція даної моделі представлена на рисунку 2.3, з нього видно, які саме процеси передують отриманню результатів.

2.3 Життєвий цикл розробки системи

Розробка будь-якої система має свій життєвий цикл. Для системи кластеризації скарг на фінансові послуги, життєвий цикл розробки, буде включати наступні кроки:

1. Визначення вимог: на цьому етапі визначаються вимоги до системи шляхом розуміння потреб, цілей і обсягу аналізу кластеризації скарг на фінансові послуги.

2. Аналіз: Вимоги детально аналізуються, а існуючі бізнес-процеси, джерела даних і обмеження досліджуються для визначення функціональних і нефункціональних специфікацій системи.

3. Розробка: Система розроблена на основі проектних специфікацій. Це передбачає реалізацію інтерфейсу користувача, створення механізму кластеризації та інтеграцію функцій керування даними.

4. Тестування: розроблена система ретельно тестується, щоб переконатися, що вона функціонує належним чином. Це включає модульне тестування, інтеграційне тестування та тестування системи для виявлення та усунення будь-яких дефектів або проблем.

5. Розгортання: система розгортається в реальному середовищі, що робить її доступною для користувачів для кластеризації скарг щодо фінансових послуг. Цей етап включає установку системи, її налаштування та забезпечення її належного функціонування.

6. Оцінка: розгорнута система оцінюється для оцінки її продуктивності, точності та зручності використання. Цей етап передбачає збір відгуків від користувачів, вимірювання системних показників і визначення областей для вдосконалення.

7. Технічне обслуговування: система регулярно підтримується й оновлюється для вирішення будь-яких проблем, врахування відгуків користувачів і адаптації до мінливих вимог. Це включає виправлення помилок, оптимізацію продуктивності та вдосконалення можливостей аналізу кластеризації.

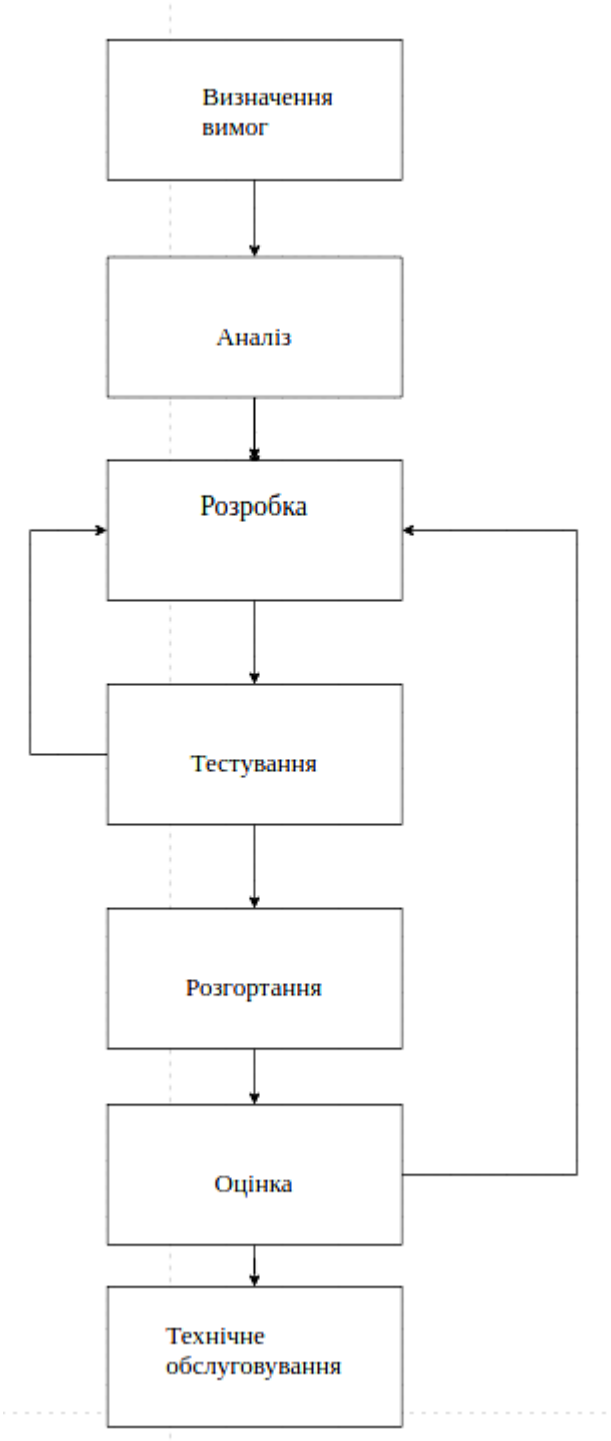

Рисунок 2.4 - Діаграма життєвого циклу розробки системи

Зв’язки в даній діаграмі життєвого циклу можна легко трактувати, через, що і сам життєвий цикл є простим і зрозумілим. Згідно з описом кроків, лінійний прохід до низу означає ідеальну реалізацію з першого разу, на практиці ж це навряд чи можливо, тому у випадку не пройденої стадії тестування, система відправляється на доопрацювання, а саме на стадію розробки, якщо ж всі тести було пройдено,

відбувається розгортання, після чого ця стадія оцінюється, щоб визначити, чи потрібне доопрацювання даних результатів, якщо ми задоволенні отриманими оцінками, то тепер система перейде, на стадію технічного обслуговування.

Дотримуючись цієї діаграми життєвого циклу, розробкою та еволюцією системи кластеризації скарг на фінансові послуги можна забезпечити чітко визначений і структурований підхід до проекту.

2.4 Визначення математичних методів вирішення задачі

2.4.1 Загальний огляд математичних методів кластеризації

Для вирішення задачі кластеризації існує величезна купа алгоритмів, але кожен з них працює з специфічними даними, там в силу логіки своєї роботи, має обмеження в плані використання його для кластеризації. Умовно алгоритм EM, що як по мені є одним із найкращих рішень в багатьох ситуаціях, але його не вийде використати в нашій задачі, так як він працює по принципу сумішей Гауса, тобто набору Гаусівських розподілів для кожної змінної, завдяки чому, кластери можуть накладатися один на одного, що і трапляється частіше всього, не буває ж лише білого чи чорного, здебільшого речі в житті якось перетинаються. Але думаю з самої назви алгоритму, думаю зрозуміло, що використовує він нормальний розподіл, а це функція для неперервних чисел, а наші дані здебільшого взагалі мають рядковий тип, тому якщо якийсь розподіл і підійде тут, то тільки дискретний та і то, для деякий змінних ми вже навіть побудували частотну гістограму, тому використання алгоритмів, що працюють тільки з числовими змінними, для нас неможливе.

Для початку ми би могли подивитися на результати кластеризації цих, досить великих даних, за допомогою простих алгоритмів, тим паче, bias variance tradeoffer, ніхто не відміняв і поступово переходити від алгоритмів з високим показником одного параметру, до іншого, є досить адекватною ідеєю, так можна аналізувати і стан даних на основі результативності алгоритмів.

З перелічених в першому розділі способів вирішення задачі кластеризації, я буду користуватися математичними алгоритмами кластеризації. Серед їх великої кількості треба обрати множину тих, що ми можемо використати для вирішення нашої задачі з наявними даними.

В нашій задачі не вийде використати будь-який алгоритм, тому я б відзначив такі алгоритми, від простих, до складних.

1. Алгоритму типу “k-means”, що побудовані по принципу визначення відстані між точками даних та зсуву центроїдів.
2. Ієрархічні алгоритми кластеризації.
3. Алгоритми, принцип яких полягає у виявленні щільних областей даних і групованні точок даних, які знаходяться на певній відстані одна від одної.
4. Удосконалені алгоритми, що об’єднують в собі різні підходи.

Серед кожного типу є різні представники, але серед алгоритмів першого типу я б зосередив увагу на “K-Mode”, одна з варіацій “k-means” використання якого буде пояснено далі проте треба буде вирішити питання його використання з нашими даними.

Серед ієрархічних алгоритмів є багато варіацій але основа в них одна, щоб адекватно використати їх з нашими даними, буде доречно додати вирахування матриці Гауера, для прикладу з таким типом перетворення гарно працює така варіація як “Agglomerative Clustering” .

Останнім типом є не сам тип, а удосконалення базових концепцій. В нашій задачі це виглядає цікаво, так як даних ми маємо реально багато. Серед них також є удосконалення першого типу алгоритмів, що ми і використаємо для порівняння результатів кластеризації та інших підхід, що базується на побудові кіл з заданим радіусом, де точки, які увійшли в цей радіус, є представниками кластеру, відмінність від першого типу, полягає у тому, що тут немає центроїдів , тут об’єднуються саме купки точок, що в подальшому буде дуже легко візуалізувати.

2.4.2 Опис удосконалення “k-means” для категоріальних даних, “K-Mode”

Одним з простих алгоритмів кластеризації, є “k-means”, його логіка роботи, є досить простою та зрозумілою, він намагатися мінімізувати функцію похибки, яка представляє з себе суму різниць квадратів відстаней до центроїдів, обрахованих алгоритмом, алгоритм є дуже простим і центроїди змінюють свою позицію після кожної ітерації, потім вираховується середнє арифметичне координат всіх точок кластера і центроїд зміщується.

Дійсно простий алгоритм, але в нашому випадку ми не маємо ніяких координат, у нас рядки, а не числа.

На поточний момент, ми знаємо, що для використання простих алгоритмів на кшталт “k-means”, нам треба визначити 2 операції, визначення відстані між двома точками та перерахунок центроїдів.

Одним із різновидів алгоритму k-means” для категоріальних даних, є “K-Mode”, його різниця з “k-means” полягає у тому, що він для визначення відстані, використовує не евклідову відстань, а відстань Хемінга, яка працює по принципу порівняння значень, якщо збігаються, то відстань по відповідній координаті 0, якщо ні, то 1, в результаті береться сума, і так отримується відстань між точками.

$$d(x, y) = \sum_{j=1}^{m} \delta(x_j, y_j)$$

$$\delta(x_j, y_j) = \begin{cases} 0 & \text{if } x_j = y_j \\ 1 & \text{if } x_j \neq y_j \end{cases}$$

Рисунок 2.18 – Формула відстані Хемінга

Для визначення нових центроїдів, замість середнього арифметичного, використовується, як не важко здогадатися з назви, мода, найпопулярніше значення серед усіх точок, що були позначені як частина кластера.

2.4.3 Метод ліктя на основі функції міжкластерної відстані

При використанні такого алгоритму як “k-modes”, одним із параметрів який задається при ініціалізації алгоритму, є кількість кластерів, так як же оптимально визначити це число.

Одним із методів визначення оптимальної кількості кластерів, є метод ліктя, він полягає у тому, що функція похибки завжди падає з збільшенням кількості кластерів, так як значення функції похибки рівне 0, буде досягнуто тільки в тому випадку, коли кількість кластерів буде рівною кількості точок, бо коли існують кластери відстань до яких 0, то до них і буде визначено точку, тому зі збільшенням кількості кластерів, результат буде покращуватися, таким чином, можна просто слідкувати за зменшенням функції похибки, на початкових етапах швидкість її падіння буде дуже великою, але починаючи з якоїсь кількості ітерацій, швидкість зменшиться, і на графіці функції похибки від кількості ітерацій, утвориться лікоть, в честь чого і названо метод.

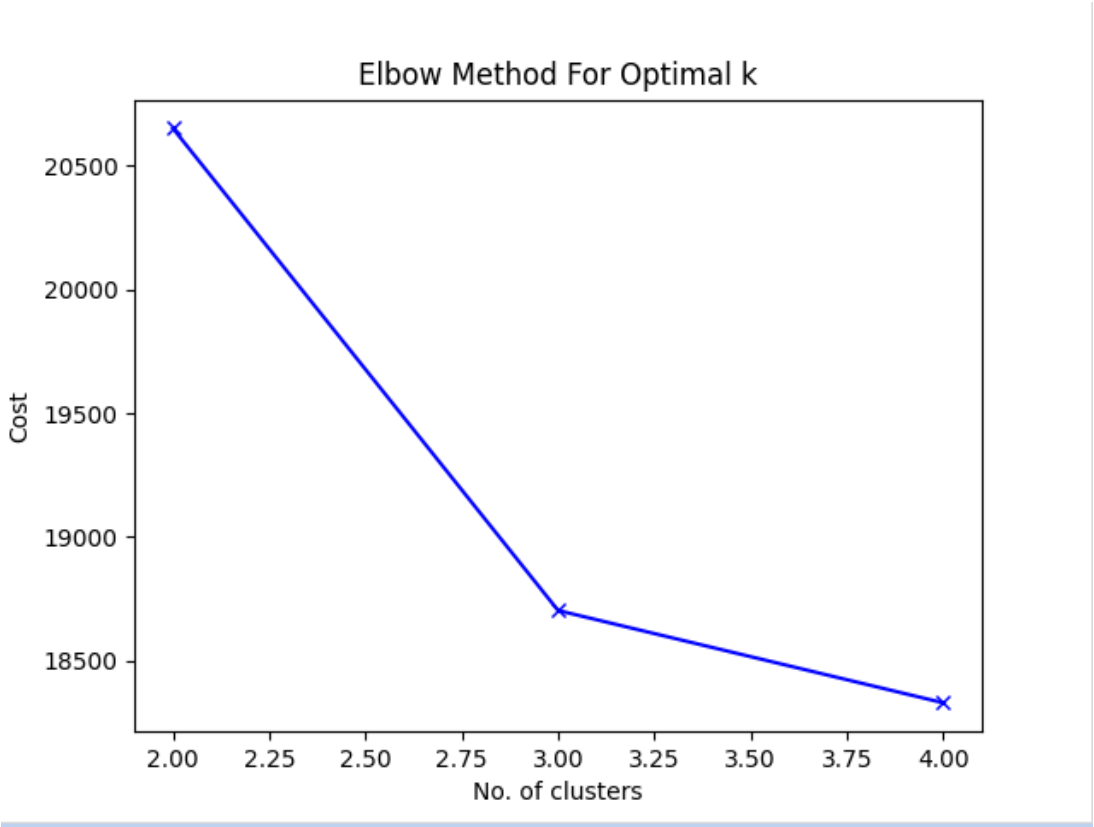

Рисунок 2.19 – Приклад ліктя

На малюнку зображено графік залежності функції похибки від кількості ітерації, тут зображена дуже мала кількість кластерів, але зменшення швидкості падіння функції похибки дуже добре помітно між 2 та 3 кластерами та 3, 4. Згідно цього малюнку, доцільно було б використати саме 3 кластери, але можливо на більшій кількості кластерів, результати будуть ще краще, тому в таких ситуаціях, коли ми маємо справу з таким великим набором даних, краще більш детально дослідити графік зменшення функції похибки.

2.4.4 Оцінка силуету кластеризації як метод визначення оптимальних значень параметрів алгоритму

Також одним з варіантів визначення оптимальної кількості кластерів є використання оцінки силуету кластеризації. Це оцінка, що знаходиться в межах від -1 до 1 та показує наскільки кожна точка відповідає кластеру до якого вона була визначена, підхід був запропонований бельгійським статистиком Пітером Руссеу в 1987 році. Обрахувавши оцінку відповідності кластеру для кожного спостереження, ми можемо знайти середнє значення відповідності, на основі якого і робити висновки про оптимальні значення параметрів.

Математично оцінка силуету обраховується як:

1. Для кожної точки даних у наборі даних:

Обчислити середню різницю (відстань) між точкою даних і всіма іншими точками даних у межах одного кластера. Це позначається як "a(i)".

$$a(i) = \frac{1}{|C_I|-1} \sum_{j \in C_I, i \neq j} \quad d(i,j) \tag{2.1}$$

Де C_I, це кластер до якого належить спостереження i, $|C_I|$- це кількість спостережень, що належать кластеру, функція *d(i, j)* — це відстань між спостереженнями i та j в кластері C_I

2. Для кожної точки даних у наборі даних:

Обчислити середню різницю між точкою даних і всіма точками даних у кожному з інших кластерів. Обирається кластер із мінімальною середньою несхожістю. Це позначається як "b(i)".

$$b(i) = \min_{J \neq I} \frac{1}{|C_I|} \sum_{j \in C_I} \quad d(i,j) \tag{2.2}$$

3. Для кожної точки даних у наборі даних:

Розраховуємо силуетний коефіцієнт, що позначається як «s(i)», за формулою:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{2.3}$$

4. Обчислити загальний коефіцієнт силуету для всього набору даних, взявши середнє значення всіх коефіцієнтів силуету:

$$\underline{s}(k) = \frac{\sum_{i=1}^{k} \quad s(i)}{} \tag{2.4}$$

Згідно з описаним математичним алгоритмом, легко помітити, що самі значення будуть варіюватися від 1 до -1, де 1 є найкращим значенням, до того ж мінімальне значення кількості кластерів для обрахування цієї оцінки, є 2, так як при розрахунку приймають значення, щонайменше два кластери.

Цей коефіцієнт, стане нам у нагоді, при визначенні оптимальних значень кластерів алгоритмів, що не працюють по принципу центроїдів, а мають в своїй основі інші механізми, що не дозволяє використовувати середню віру відстані в кластерах, також коефіцієнт силуету, прекрасно показує зміни результатів роботи алгоритму при зміні будь-якого параметру, що в нашому випадку є важливою характеристикою.

2.4.5 Ієрархічні алгоритми кластеризації. AgglomerativeClustering

Ієрархічна кластеризація — це сімейство алгоритмів кластеризації, які будують ієрархію вкладених кластерів шляхом багаторазового злиття малих кластерів у більші (агломераційні) або розбиття великих кластерів на менші

(розділові). Результат ієрархічної кластеризації зазвичай візуалізується як дендрограма, яка показує ієрархічні зв’язки між кластерами.

За рахунок можливості побудови такої дендрограми, ми можемо побачити як з кожною ітерацією, кількість кластерів стає більшою у випадку розділових алгоритмів, чи зменшується, коли використано агломераційні алгоритми.

В свою чергу, постають нові питання, одне з яких визначення відстані між кластерами Відстань між кластерами можна обчислити за допомогою різних вимірювань відстані, таких як евклідова відстань, манхеттенська відстань або кореляційна відстань. Існують також різні критерії зв’язку, які визначають, як відстані між кластерами об’єднуються для формування нової відстані. Ось три найпоширеніші критерії зв’язку:

1. Одинарний зв’язок: відстань між двома найближчими точками в двох кластерах.
2. Повний зв’язок: відстань між двома найдальшими точками у двох кластерах.
3. Середній зв’язок: середня відстань між усіма парами точок у двох кластерах.

Ієрархічна кластеризація має перевагу в тому, що вона здатна обробляти кластери довільної форми та не вимагає апріорного визначення кількості кластерів. Однак цей алгоритм може бути чутливим до викидів і може бути обчислювально дорогим для великих наборів даних.

Так як для роботи алгоритму необхідно визначити як саме рахувати відстань між точками, для подальшої роботи алгоритму, то саме вирішення цієї проблеми і є основним при використанні цього алгоритму. Для повноцінної роботи ми маємо вирішити цю проблему, так як ми не маємо чисел, відстань між якими ми б могли визначити в звичайному евклідовому просторі.

2.4.6 Вирішення проблеми визначення відстані між точками. Обґрунтування використання відстані Гауера

Відстань Гауера — це метрика відстані, яка зазвичай використовується для кластеризації або аналізу подібності наборів даних, що містять змішані типи даних, наприклад числові, категоріальні та двійкові змінні. Він названий на честь Джона Гауера, який вперше запропонував метрику в 1971 році.

Відстань Гауера – це зважена відстань, яка враховує різні масштаби та типи даних змінних у наборі даних. Відстань між двома спостереженнями обчислюється як зважена сума відстаней між їхніми змінними, де ваги визначаються на основі типу даних кожної змінної.

Дану метрику визначення відстані доречно використовувати в задачах кластеризації категоріальних даних з кількох причин:

1. Обробляє змішані типи даних: відстань Гауера призначена для обробки змішаних типів даних, у тому числі категорійних даних, з якими може бути важко працювати з іншими показниками відстані. Беручи до уваги різні типи даних і масштаби змінних, відстань Гауера забезпечує більш точну міру подібності між спостереженнями.

2. Зважена відстань: відстань Гауера — це зважена метрика відстані, що означає, що важливість кожної змінної враховується явно. Для категоріальних змінних вага обернено пропорційна кількості категорій у змінній. Це гарантує, що на відстані між спостереженнями не домінують змінні з багатьма категоріями.

3. Надійність до відсутніх даних: відстань Гауера може обробляти відсутні дані, що часто зустрічається в проблемах кластеризації категоріальних даних. Відсутні дані розглядаються як окрема категорія в розрахунку відстані, і відстань коригується відповідно.

4. Немає потреби у перетворенні даних: відстань Гауера не потребує перетворення чи нормалізації даних, що може зайняти багато часу та внести зміщення в результати. Це полегшує застосування методу до різних наборів даних.

5. Ефективне обчислення: відстань Гауера можна ефективно обчислити для великих наборів даних, що важливо для задач кластеризації, які включають багато спостережень або змінних.

Загалом відстань Гауера є універсальною та надійною метрикою відстані, яка добре підходить для проблем кластеризації, пов'язаних із категоріальними даними. Його здатність обробляти змішані типи даних, ваги, відсутні дані та ефективне обчислення робить його привабливим варіантом для дослідників даних, які працюють з категоріальними даними.

Сама відстань визначається наступним чином. Для набору даних із n спостережень і p змінних відстань Гауера між спостереженнями i та j визначається як:

$$D_{Gower}(x_1, x_2) = 1 - \left(\frac{1}{p} \sum_{j=1}^{p} s_j(x_1, x_2) \right)$$

Де p це кількість змінних, а $s_j(x_1, x_2)$ це функція, що визначається як:

Для двох числових змінних.

$$s_j(x_1, x_2) = 1 - \frac{|y_{1j} - y_{2j}|}{R_j}$$

Для категоріальних змінних визначається як Dice Distance. Щоразу, коли значення рівні, Dice Distance = 0, а коли вони не рівні виконується наступний алгоритм дій.

DiceDistance ➡ NNEQ / (NTT + NNZ)

- N : number of dimensions
- NTT : number of dims in which both values are True
- NTF : number of dims in which the first value is True, second is False
- NFT : number of dims in which the first value is False, second is True
- NFF : number of dims in which both values are False
- NNEQ : number of non-equal dimensions, NNEQ = NTF + NFT
- NNZ : number of nonzero dimensions, NNZ = NTF + NFT + NTT

Відстань Гауера має кілька переваг перед іншими показниками відстані для змішаних типів даних. Він може обробляти відсутні значення та різні типи даних, не вимагаючи перетворення даних, і його можна ефективно обчислювати для великих наборів даних.

2.4.7 Алгоритм DBSCAN. Обґрунтування використання

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — це алгоритм кластеризації, який об’єднує точки, розташовані близько одна до одної в просторі та мають високу щільність. Алгоритм базується на ідеї, що кластери – це області високої щільності, розділені областями низької щільності, і не вимагає попереднього вказівки кількості кластерів.

Алгоритм DBSCAN має два важливі параметри: ε і minPts. Значення ε визначає радіус околиці навколо кожної точки, а minPts визначає мінімальну кількість точок, необхідних для формування кластера. Тому важливо зазначити як саме можна визначити їх оптимальні значення.

Наступні кроки можуть допомогти вам визначити оптимальні значення для ε і minPts:

1. Візуалізуйте дані: побудуйте точки даних на точковій діаграмі, щоб отримати уявлення про розподіл даних. Це може дати вам уявлення про те, наскільки щільні дані та скільки кластерів ви можете очікувати.

2. Використовуйте діапазон значень для eps: почніть із діапазону значень eps і запускайте DBSCAN для кожного значення. Ви можете спробувати значення в діапазоні від дуже малих до дуже великих, залежно від масштабу ваших даних. Ви також можете використовувати логарифмічну шкалу, щоб отримати краще уявлення про діапазон значень, які добре працюють.

3. Побудуйте графік відстані досяжності: для кожної точки в наборі даних побудуйте її відстань досяжності проти її індексу. Відстань досяжності точки вимірює відстань до найближчої точки в кластері. Це може допомогти вам візуалізувати, як щільність даних змінюється з різними значеннями eps. Ви можете використовувати цей графік, щоб визначити хороше значення для eps.

4. Використовуйте оцінку силуету: оцінка силуету вимірює якість кластеризації. Ви можете спробувати різні значення min_samples і обчислити оцінку силуету для кожного значення eps і min_samples. Виберіть комбінацію eps і min_samples, яка дає вам найвищу оцінку силуету.

Загалом, визначення оптимальних параметрів для DBSCAN передбачає поєднання візуалізації, експериментування та перевірки.

Серед переваг тут є те, що алгоритм сам встановлює кількість кластерів, що є дуже важливою його перевагою над іншими алгоритмами, також особливість його роботи, дозволяє описувати неймовірні з точки зору геометричного розміщення кластери.

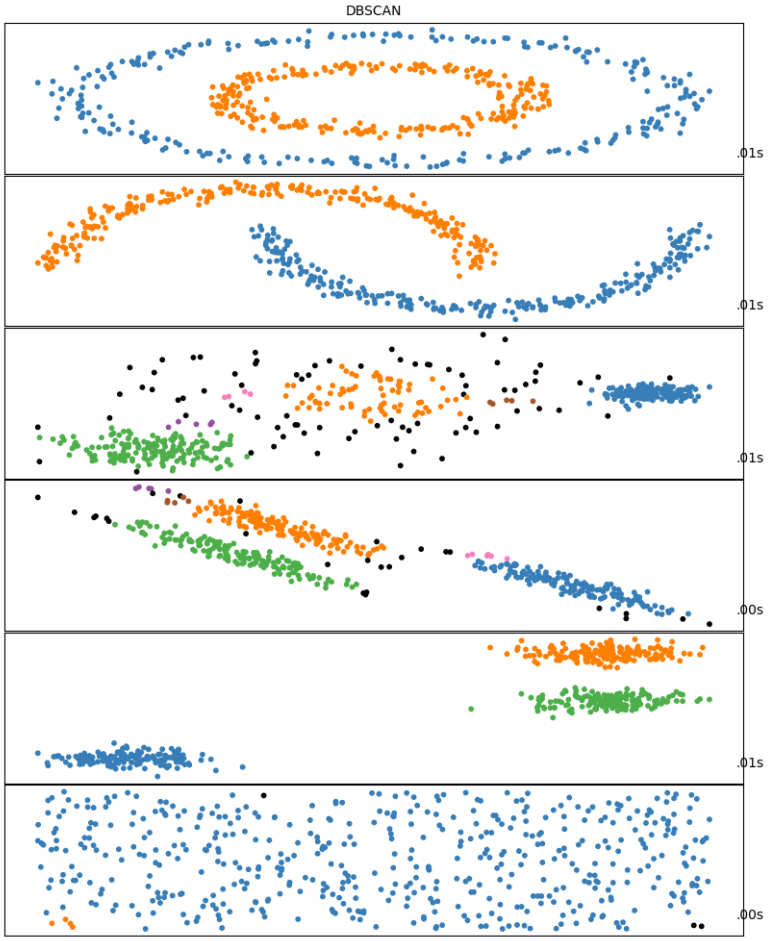

Рисунок 2.20 – Приклад роботи алгоритму DBSCAN при різному розміщенні точок

Даний приклад наочно демонструє, як алгоритм поводить себе в різних ситуаціях, а головне, що для нього не потрібна позиція цих точок, все виконується за рахунок відстані між точками. Також це показує, що отримані кластери можуть бути довільної форми та розміру і можуть відрізнятися за щільністю. Алгоритм особливо корисний для наборів даних зі складною структурою та шумом, і він може виявляти викиди як точки шуму. Однак на продуктивність алгоритму може вплинути вибір параметрів, і він може погано працювати з наборами даних різної щільності.

2.4.8 Алгоритм HDBSCAN. Основні відмінності та переваги над DBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) — це ієрархічний алгоритм кластеризації, який базується на щільності.

Подібно до DBSCAN, HDBSCAN об’єднує точки, які розташовані близько одна до одної в просторі та мають високу щільність, але також призначає оцінку стабільності кожному кластеру, який відображає, наскільки добре кластер підтримується даними. Ця оцінка враховує розмір кластера, щільність і рівень шуму.

Алгоритм HDBSCAN має два важливі параметри: мінімальний розмір кластера та мінімальні вибірки на кластер. Мінімальний розмір кластера визначає найменшу кількість точок, які можуть сформувати кластер, а мінімальна кількість вибірок на кластер визначає мінімальну кількість точок, необхідну для формування кластера. Однак HDBSCAN автоматично визначає оптимальні значення цих параметрів на основі даних за допомогою надійної методики єдиного зв’язку.

HDBSCAN має всі особливості свого пращура та крім цього забезпечує вимірювання стабільності кластерів, що може допомогти ідентифікувати надійні кластери в даних. Однак на продуктивність алгоритму може вплинути вибір метрики відстані та розмір набору даних.

DBSCAN і HDBSCAN — це алгоритми кластеризації, засновані на щільності, але вони відрізняються кількома важливими параметрами:

1. Ієрархічна структура: HDBSCAN — це ієрархічний алгоритм кластеризації, який створює дерево кластерів, тоді як DBSCAN створює плоский розподіл даних на кластери та шумові точки.

2. Стабільність кластера: HDBSCAN призначає оцінку стабільності кожному кластеру, яка відображає, наскільки добре кластер підтримується даними. Ця оцінка враховує розмір кластера, щільність і рівень шуму. Навпаки, DBSCAN такого не робить.

3. Вибір параметрів: HDBSCAN автоматично визначає оптимальні значення двох ключових параметрів, мінімального розміру кластера та мінімальних вибірок на кластер, на основі даних. Це робиться за допомогою техніки під назвою надійне єдине зв’язування, яка поєднує різні кластеризації, отримані зміною

параметрів. DBSCAN вимагає ручного вказівки цих параметрів, що може бути складним для наборів даних із різною щільністю та формою.

4. Продуктивність: HDBSCAN зазвичай повільніше, ніж DBSCAN, особливо для великих наборів даних, оскільки передбачає побудову ієрархії кластерів. Однак HDBSCAN може бути швидшим за DBSCAN для наборів даних із багатьма шумовими точками, оскільки він уникає розгляду шумових точок як кластерів.

5. Обробка шуму: HDBSCAN може обробляти точки шуму ефективніше, ніж DBSCAN, оскільки розрізняє точки шуму та невеликі кластери, які погано підтримуються даними. Навпаки, DBSCAN розглядає всі точки, які не належать до кластерів, як точки шуму.

Загалом головними перевагами HDBSCAN перед DBSCAN є його здатність автоматично визначати оптимальні параметри та його стійкість до шуму та змінної щільності. Однак HDBSCAN є обчислювально дорожчим, ніж DBSCAN, і його ієрархічна структура може не підходити для всіх програм.

2.5 Переваги та перспективність вибраного алгоритмічного підходу

В минулому розділі були детально описані алгоритми, що будуть використовуватися та їх переваги. В свою чергу їх ефективність та доцільність використання для кластеризації скарг на фінансових послуг, можна пояснити наступним переліком причин:

1. Виявлення прихованих закономірностей: скарги на фінансові послуги часто містять складні та неструктуровані текстові дані. Застосовуючи запропоновані алгоритми кластеризації, ми зможемо виявити приховані закономірності та структури в скаргах. Ці шаблони можуть виявити загальні теми, повторювані проблеми або конкретні сфери, які хвилюють користувачів.

2. Виявлення нових проблем: фінансові послуги розвиваються, і з часом можуть виникати нові проблеми. Алгоритми кластеризації можуть допомогти у

виявленні нових проблем шляхом групування скарг, які мають подібні характеристики, але, можливо, не були явно віднесені до категорії нової проблеми. Це допомагає постачальникам фінансових послуг у проактивному вирішенні проблем, що виникають.

3. Ефективне поводження з великими обсягами даних: постачальники фінансових послуг часто отримують значну кількість скарг. Такі алгоритми, як DBSCAN, підходять для великомасштабної кластеризації, оскільки вони можуть ефективно обробляти багатовимірні та великі набори даних. Вони можуть масштабовано обробляти скарги, забезпечуючи основу для аналізу та розгляду великої кількості відгуків користувачів.

4. Обробка шуму та викидів: набори даних скарг можуть містити зашумлені дані або викиди, які є скаргами, які не належать до жодного конкретного кластера. DBSCAN, зокрема, може ефективно справлятися з шумом і викидами. Він ідентифікує кластери на основі щільності, дозволяючи розрізняти значущі кластери та шум/викиди, таким чином забезпечуючи точніше представлення базових шаблонів.

5. Розширене прийняття рішень: алгоритми кластеризації полегшують прийняття рішень на основі даних у сфері фінансових послуг. Групуючи скарги, постачальники фінансових послуг можуть отримати уявлення про настрої користувачів, визначити сфери покращення та визначити пріоритетність дій на основі серйозності та частоти конкретних скарг. Це сприяє підвищенню задоволеності клієнтів і якості обслуговування.

6. Відповідність нормативним вимогам. Скарги на кластеризацію можуть підтримати заходи щодо дотримання нормативних вимог у фінансовому секторі. Аналізуючи згруповані дані про скарги, організації можуть виявити потенційні проблеми з відповідністю, виявити моделі, які можуть порушувати правила, і вжити відповідних заходів для їх швидкого вирішення.

7. Постійне вдосконалення: Алгоритми кластеризації дозволяють проводити постійний аналіз і вдосконалювати процеси управління скаргами.

Відстежуючи та аналізуючи згруповані скарги з часом, постачальники фінансових послуг можуть відстежувати зміни в настроях користувачів, оцінювати ефективність стратегій пом’якшення наслідків та впроваджувати постійні вдосконалення своїх продуктів і послуг.

2.6 Опис файлової структури збереження результатів

Під час роботи з застосунком, користувач повинен мати змогу зберігати результати роботи. Для цього необхідно визначити структуру збереження результатів.

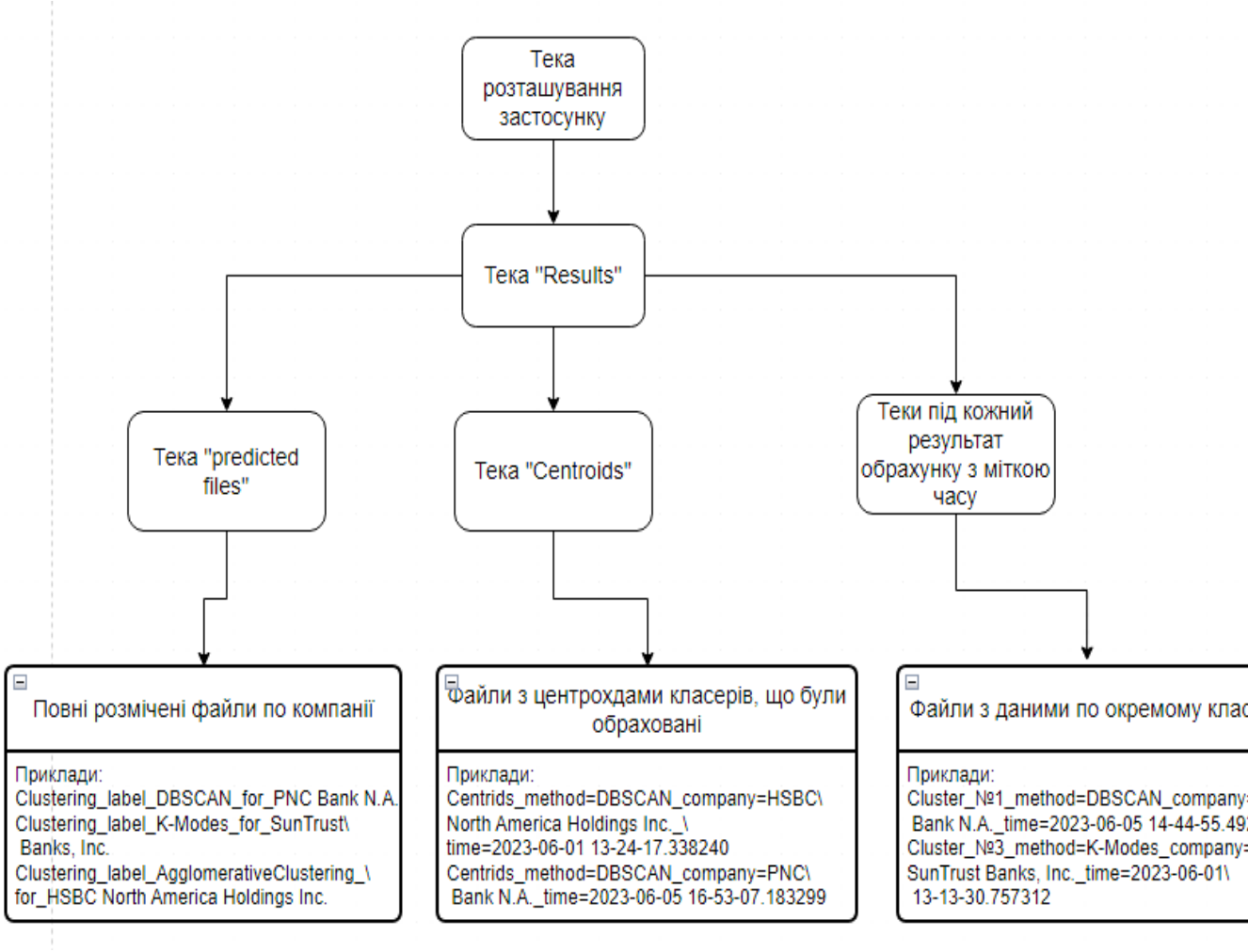

Рисунок 2.21 – Діаграма файлової структури збереження результатів

На рисунку 2.21 показано як саме буде відбуватися збереження результатів. В теці розміщення основного файлу програми, буде створюватись підтека, за умови її

відсутності. Далі будуть існувати дві постійні теки, одна”predicted files”, в який будуть зберігатися повні, оригінальні файли, а саме всі записи про скарги на послуги конкретної компанії. Від цього і шаблон назви файлів, початком назви слугує “Clustering_label_” після чого вказується алгоритм яким буде обраховані дані мітки, потім вказується для якої компанії були визначені мітки, на малюнку показано приклади таких файлів, всі вони мають розширення “.csv”.

Інша базисна підтека з назвою “Centroids”, в ній зберігаються обраховані центроїди за результатами алгоритму. Дані файлі потрібні для біглого погляду на кластер, щоб зрозуміти, які саме скарги наявні в даному кластері, так би мовити модальне значення по кожному кластеру. Приклади кожного файлу показано на рисунку, шаблон дуже схожий, вказується метод обрахунку, компанія та мітка часу, щоб кожен файл точно був унікальний і можна було встановити, де саме знаходиться потрібний результат.

Останніми підтеками є теки, що містять в собі файли, в яких лежать дані, окремо по кожному кластету, щоб можна було більш детально дослідити саме той кластер, що зацікавив користувача. Працює це наступним чином, після збереження результатів, створюється тека зі своєю специфічною назвою, де вказується алгоритм, компанія та мітка часу, тека виходить унікальна за будь-яких умов, після чого вже в неї зберігаються файли за таким шаблоном, вказується номер кластера, компанія, алгоритм та мітка часу, щоб файл випадково не був з тією ж назвою та не замінив собою минулі результати. Така тека створюється під кожний набір файлів.

2.7 Блок-схема алгоритму роботи програмного модулю

Для розуміння того, як повинен функціонувати програмний застосунок, доцільно буде побудувати загальну блок-схему алгоритму.

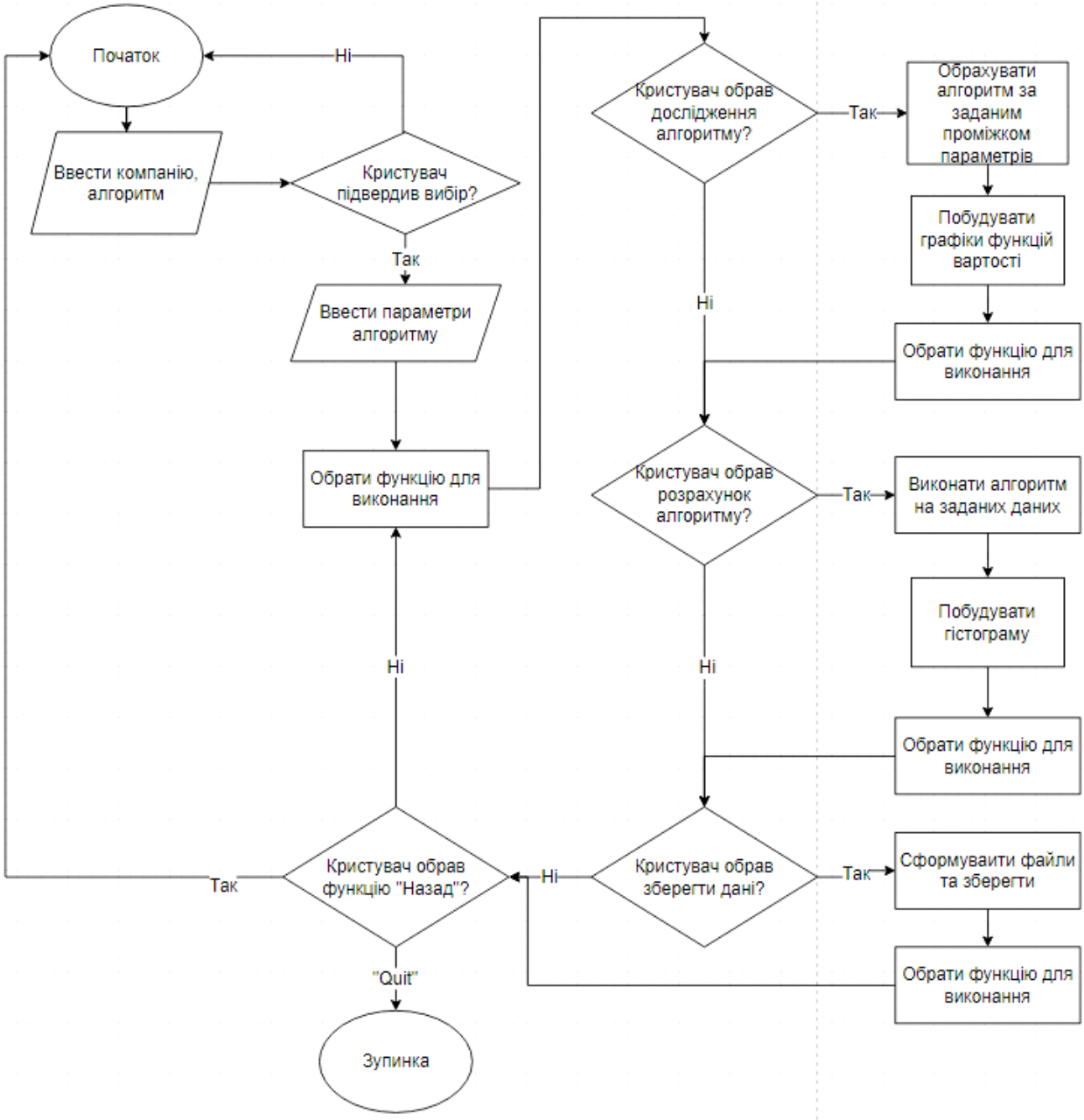

Рисунок 2.22 – Блок-схема алгоритму взаємодії користувача з застосунком

На малюнку продемонстровано блок-схему, яка представляє собою сценарій того, як користувач буде взаємодіяти з програмою і які саме вибори буде мати. З основних функцій, які були описані в підрозділі 2.1, тут представлені всі необхідні для роботи користувача з системою.

Як і зазначалося, користувач на початку зможе обрати дані за яку саме компанію він хоче кластеризувати і як саме він це хоче зробити, обравши алгоритм кластеризації, далі за умови, що користувач підтвердив свій вибір, він переходить до наступного кроку, де йому буде запропоновано ввести параметри обраного

алгоритму, зрозуміло, що для кожного окремого алгоритму, це буде окремий набір параметрів та окреме унікальне вікно. Також користувач зможе провести дослідження ефективності обраного ним алгоритму для конкретного набору даних та дослідити вплив зміни параметрів алгоритму на результати роботи, це буде зручним чином представлено графічно, за допомогою чого і буде реалізовано описані в підрозділах 2.4.3 та 2.4.4 підходи до визначення оптимальних значень параметрів, на блок-схемі ці етапи позначаються як “побудова графіків функцій вартості”, надалі користувач зможе визначивши оптимальні значення параметрів та використати їх для кластеризації обраних даних. Після виконання саме коастеризації, користувач зможе обачити гістограму абсолютних частот по кожному кластеру, на основі якої вирішити, чи влаштовує його таке розбиття та чи є сенс продовжувати роботу з таким розділенням даних. При задовільному відгуку користувача, він зможе зберегти обраховані дані у визначеному форматі збереження файлів, описаному в розділі 2.7. Після чого користувач може просто закрити програму, чи наприклад спробувати інший алгоритм чи використати дані іншої компанії.

2.8 Проектування прототипу графічного інтерфейсу

Інтерфейс користувача повинен бути заздалегідь продуманим, В випадку програмного застосунку, для кластериазції скарг на фінансові послуги компаній, щоб не перенавантажувати користувача, за стосунок буде мати 2 шаблони екрани, з якими він буде мати змогу взаємодіяти.

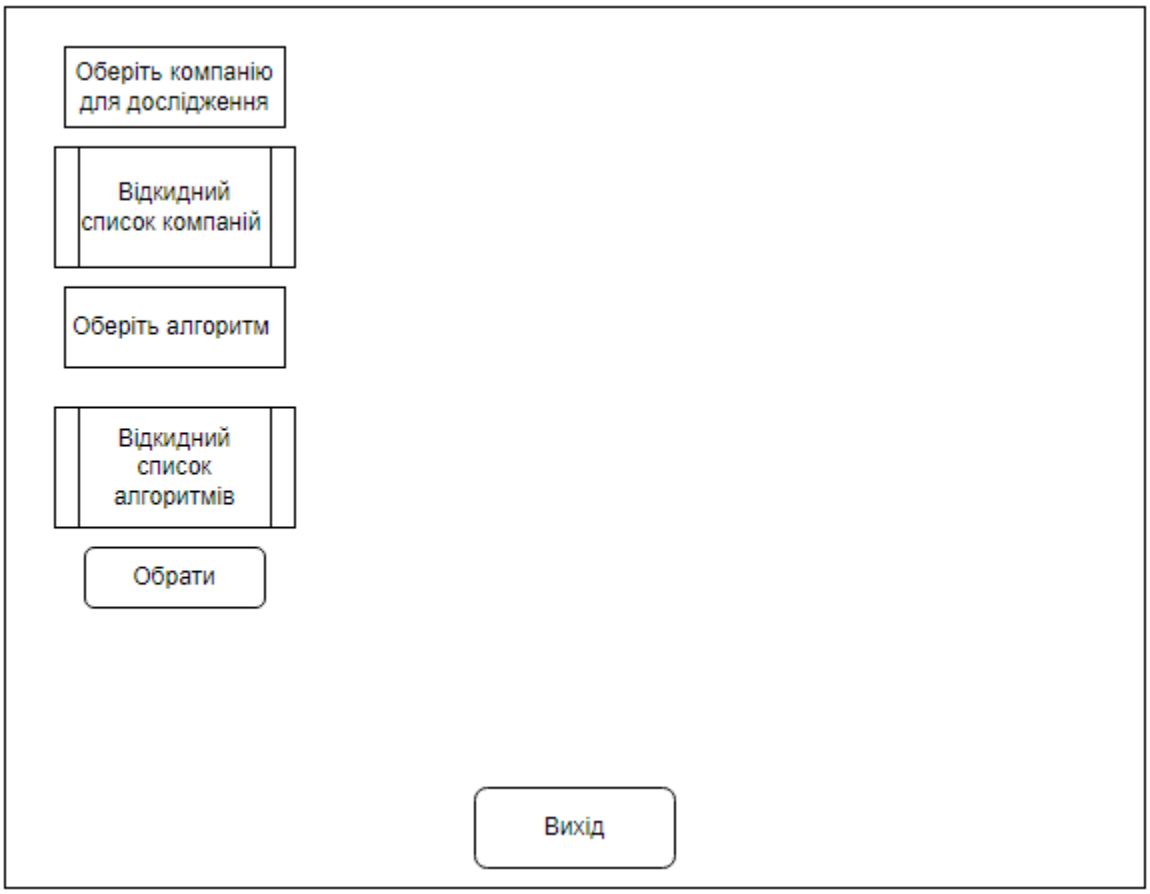

Рисунок 2.23 – Прототип початкового екрану графічного інтерфейсу

На рисунке 2.23 зазначено які можливості буде мати користувач, від початку роботи, програми. На самому початку користувачу буде запропоновано обрати компанію дані про яку він хоче обробити та алгоритм, який він хоче використати для вирішення цієї задачі. Визначившись, користувач зможе натиснути на кнопку “Обрати” та продовжити роботу вже новим екраном.

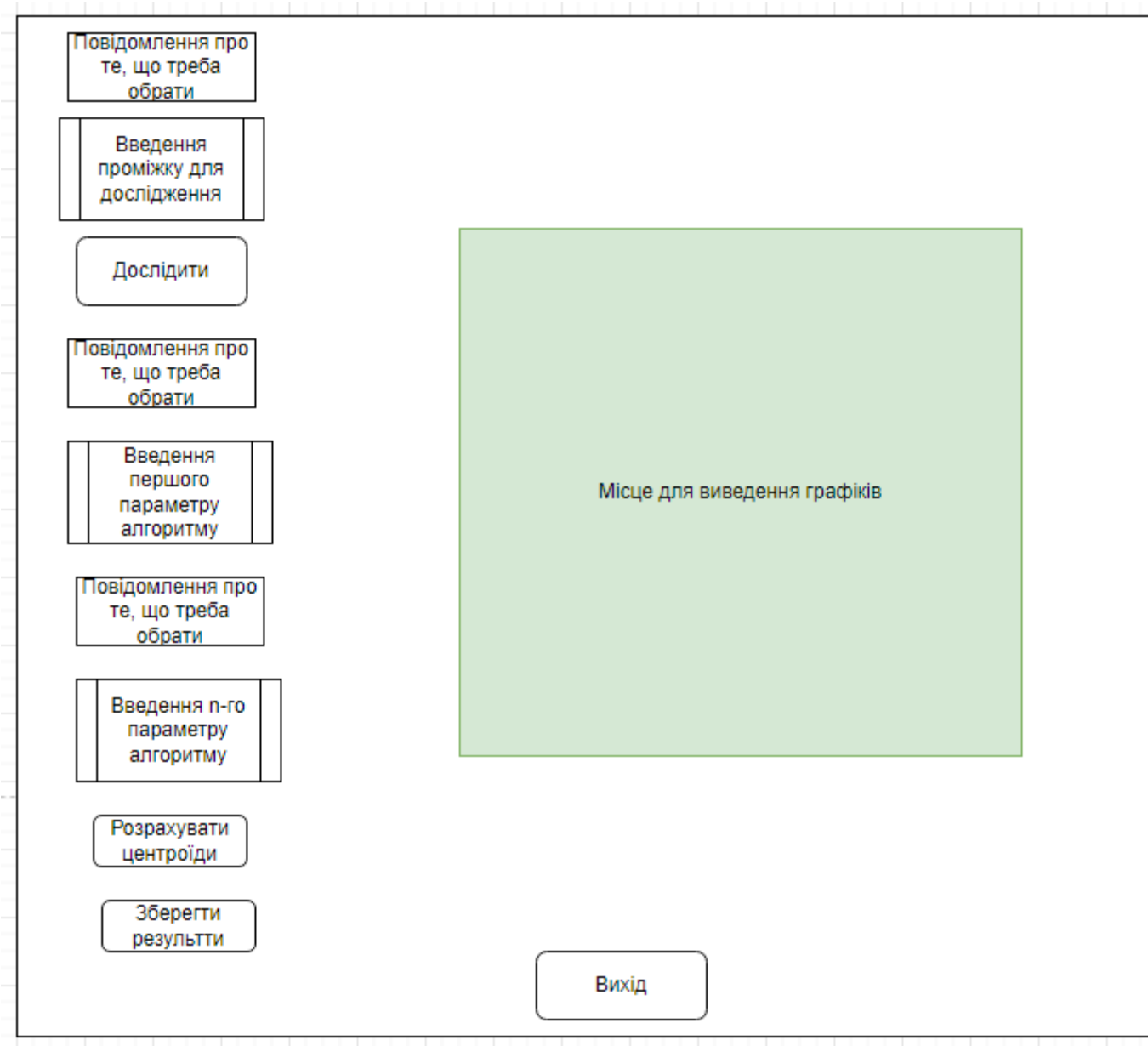

Рисунок 2.24 – Прототип робочого екрану застосунку

Рисунок 2.24 демонструє як буде виглядати екран взаємодії користувача з програмою, коли він визначиться з компанією та алгоритмом. Йому буде запропоновано дослідити якийсь проміжок значень параметрів та порівняти ефективність алгоритму при різних значеннях параметрів. При виконанні цієї функції, користувачеві будуть виведені графіки, в заздалегідь визначену зону , в залежності від алгоритму, осі та значення в цих графіках будуть відрізнятися, але кожен алгоритм буде мати таку функцію. Надалі, за умови коли користувач визначився, за яких саме значень параметрів він хоче отримати результати, він вводить ці значення натискає кнопку “Розрахувати центроїди”, після чого алгоритм

починає своє навчання та зберігається системою. Користувачеві буде виведено стовпчасту діаграму абсолютного розподілу частот по кожному кластеру. Якщо це задовольнить користувача, він зможе натиснути кнопку “Зберегти результати”, яка збереже результати згідно структури, описаної в підрозділі 2.8. Спільною рисою обох екранів є кнопка виходу, користувач, постійно буде мати можливість завершити роботу програми натиснувши кнопку “Вихід”.

2.9 Підсумки розділу

В ході розбору математичних методів вирішення проблеми, було визначено різні типу алгоритмів, та визначено які з них будуть використані для вирішення проблеми кластеризації скарг користувачів, також було усунуто проблеми алгоритмів, що не дозволяють використовувати їх з категоріальними даними. Описано головні переваги використання кожного з обраних алгоритмів. В наступному розділі буде продемонстровано програму, що реалізує використання цих алгоритмів на даних про скарги користувачів на фінансові послуги. Проведено кластерний аналіз результатів та їх розбір.

Розділ 3 РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис структури програми та визначення набору технологій

Задля ефективної перевірки доцільності використання того, чи іншого підходу вирішення задачі, доречним буде створити програмний застосунок, що дозволить зручно та швидко, обирати дані за якими ми хочемо провести кластеризацію, обирати алгоритм для вирішення задачі та бачити візуальне представлення результатів роботи алгоритму. Це дозволить швидко створити необхідну кількість вже оброблених даних, аналізуючи які ми зможемо порівняти отримані результати з теоретичними закономірностями, що були визначені в минулому розділі та ґрунтуючись на яких і проводиться дане дослідження.

Програмний інтерфейс надавати користувачу можливість налаштування параметрів алгоритмів, крім цього необхідно додати поля для вибору алгоритму та даних які ми хочемо кластеризувати. При використанні того чи іншого алгоритму, повинні мати механізм, що дозволить користувачу визначити оптимальні параметри обраного алгоритму на основі методів описаних в підрозділі 2.4.3 та 2.4.4. Остаточною потребою буде можливість збереження результатів в унікальні файли, які потім можна буде використовувати при аналізі чи детальному розборі іншими працівниками, чи штатом аналітиків.

Візуалізацію будемо також роботи на мові Python, за допомогою бібліотек tkinter, networkx, matplotlib. Самі програми будуть реалізовані в вигляді окремих модулів.

Структура програми складається з декількох модулів. Перший – це модуль, написаний на мові Python в якому реалізовано аналіз та обробку початкових даних. Другий – це модуль, містить в собі функції, для створення моделі, її навчання та створення графіку зменшення функції вартості, в ньому використано модулі написані на мові Python, під капотом якої міститься швидша на надійніша мова C++, так як Python дуже погано виділяє пам’ять і враховуючи наші об’єми даних, написання власної реалізації алгоритмів, що були описані в підрозділі 2.5 є

недоречним, використовуючи клас з вже готовими методами які реалізують процес навчання моделі ми значно пришвидшимо виконання програми. Третій модуль містить в собі реалізацію графічного інтерфейсу користувача, для зручного користування програмою та кращого розуміння як нею користуватися, в ньому описані функції для розрахунку кластерів. Останньою частиною програмного застосунку, будемо модуль, що реалізує збереження даних.

Загальний вигляд програми у вигляді модульної структури можно подати наступним чином.

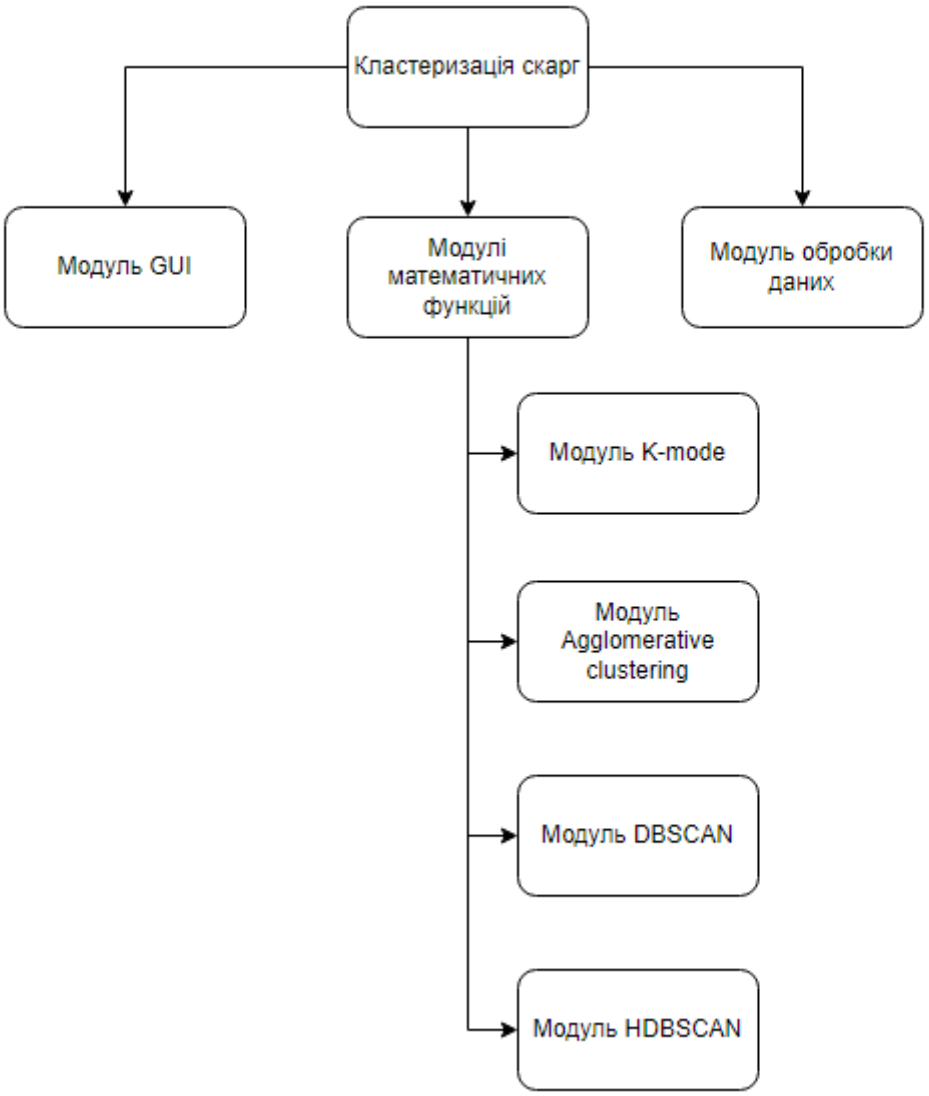

Рисунок 3.1 – Модульна структура програми.

За такої структури, програма зможу відокремлено використовувати тільки

необхідні в даний момент модулі, не перешкоджаючи роботі інших та менше навантажуючи систему.

3.2 Аналіз даних для вирішення задачі

В минулому розділі було зазначено яка саме інформація потрібна для вирішення задачі кластеризації, тому бажано знайти датасет з реальними даними який відповідає зазначеним умовам. Одним із таких, є датасет “Consumer Complaints”. В ньому зібрані дані про скарги клієнтів, на різноманітні послуги, різних компаній.

Для початку доцільно було б подивитися на цей датасет, які поля в ньому є, для подальшого аналізу.

Датасет, включає в себе такі поля:

Таблиця 1 - Назви змінних та їх переклад

Оригінальна назва рядка	Назва українською	Оригінальна назва рядка	Назва українською
Date received	Дата отримання	State	Область
Product	Продукт	ZIP code	Поштовий індекс
Sub-product	Субпродукт	Tags	Теги
Issue	Проблема	Consumer consent provided?	Надано згода споживача?
Sub-issue	Підпроблема	Submitted via	Відправлено через
Consumer complaint narrative	Опис скарги споживача	Date sent to company	Дата відправки до компанії
Company public response	Публічна відповідь компанії	Company response to consumer	Відповідь компанії споживачу
Company	Компанія	Timely response?	Чи була своєчасна відповідь?

В даному датасеті представлені всі вищезазначені необхідні типи інформації.

Типи інформації та поля, що їх надають:

- Дані про скарги клієнтів: 'Issue', 'Sub-issue', 'Consumer complaint narrative'.
- Інформація про клієнта: 'State', 'ZIP code', 'Tags'.
- Інформація про мітку часу: 'Date received', 'Date sent to company'.
- Інформація про категоризацію: 'Product', 'Sub-product'.
- Будь-які додаткові відповідні дані: Всі інші поля.

Таким чином видно, що датасет надає всі необхідні дані для вирішення задачі кластеризації.

Набір даних складається більш ніж з шестисот тисяч різноманітних скарг, тому доцільно було б проаналізувати набір даних для подальшої роботи. Без аналізу в такій задачі точно не обійтися, так як розмірність даних достатньо велика і хотілося б зробити її меншою, також не розуміючи природи змінних, неможливо підібрати не те що оптимальний, навіть просто робочий алгоритм на цих даних.

	Скарга
0	Керування позикою або лізингом
1	Використання дебетової або банківської картки
2	Відкриття, закриття або управління рахунком
3	Внески та вилучення
4	Обслуговування позики, платежі, ескаротний рах...
5	Спроби стягнення боргу, якого не існує
6	Рахунок для оплати
7	Модифікація позики, стягнення, вилучення нерух...
8	Тактика спілкування
9	Річна процентна ставка або процентна ставка
10	Кредитний моніторинг або захист особистості
11	Заявка, посередник або брокер іпотеки
12	Некоректна інформація в кредитному звіті
13	Повернення позики
14	Заборгований рахунок
15	Проблеми, спричинені низькими коштами
16	Неправомірне використання мого кредитного звіту
17	Процес врегулювання та витрати
18	Суперечки щодо рахунку

Рисунок 3.2 - Приклади скарг з датасету

На рисунку 2.5, показані приклади скарг в датасеті, для кращого розуміння вони представлені українською, їх оригінальна мова англійська, тому під час роботи готового застосунку, користувач буде мати справу саме з оригінальними назвами. Під час аналітичного розгляду даних, будуть використані переклади оригінальних назв українською, щоб спростити розуміння сенсу скарг, продуктів і пришвидшити ознайомлення з роботою.

Для аналізу будемо використовувати мову програмування “Python”, так як саме там є зручний інструмент для аналізу таких великих даних як наш датасет, а саме бібліотека pandas.

Першим кроком буде доцільно побачити загальний стан даних, наскільки чистий датасет в плані пропущених значень, можливо якісь з них можна відновити.

	Відсоток пропусків
Дата отримання	0.000000
Продукт	0.000000
Субпродукт	29.556008
Проблема	0.000000
Підпроблема	59.757112
Опис скарги споживача	82.895267
Публічна відповідь компанії	78.348131
Компанія	0.000000
Область	0.791085
Поштовий індекс	0.793918
Теги	85.873802
Надано згода споживача?	68.960391
Відправлено через	0.000000
Дата відправки до компанії	0.000000
Відповідь компанії споживачу	0.000000
Чи була своєчасна відповідь?	0.000000
Споживач заперечує?	6.176428
Ідентифікатор скарги	0.000000

Рисунок 3.3 – Відсоток пропусків змінних в датасеті

Самі голі цифри мало чого говорять нам про загальний стан даних, тому хотілося б побачити відсоткове співвідношення кількості пропусків по кожній змінній.

	Кількість пропусків
Дата отримання	0
Продукт	0
Субпродукт	198202
Проблема	0
Підпроблема	400730
Опис скарги споживача	555894
Публічна відповідь компанії	525401
Компанія	0
Область	5305
Поштовий індекс	5324
Теги	575868
Надано згода споживача?	462447
Відправлено через	0
Дата відправки до компанії	0
Відповідь компанії споживачу	0
Чи була своєчасна відповідь?	0
Споживач заперечує?	41419
Ідентифікатор скарги	0

Рисунок 3.4 – Кількість пропущених даних

З цих результатів, можна сказати, що змінна “Tags”, “Consumer complaint narrative”, “Sub-issue”, “Company public response”, “Consumer consent provided”, це змінні, що мають дуже велику кількість пропусків в собі, тому доцільно буде їх не використовувати, до того ж така зміна як під проблема взагалі не потрібна, бо тільки унікальних значень просто проблем вже цілих 95, але зробити аналіз на основі того, які саме значення пропущені буде важливо.

Спочатку розберемося з єдиною змінною, зміст якої не є інтуїтивно зрозумілим, а саме, “Tags”.

	Унікальні значення змінної Tags
0	Людина похилого віку
1	Військовослужбовець
2	Людина похилого віку, Військовослужбовець

Рисунок 3.5 – Унікальні значення змінної “Tags”

На основі побаченого, можна зрозуміти, що змінна “Tags” має в собі характеристику особи, яка подала скаргу, враховуючи кількість пропусків, категоріальний характер змінної та неінформативність самого значення, можна спокійно викинути її з нашого набору даних, що сильно облегчить нам роботу, бо кількість пропусків цієї змінної більше за 85%, тому її використання, тільки додасть нам ще один простір до і так вже не малої розмірності даних, а саме 18 унікальних змінних, дуже мало алгоритмів, здатні добре працювати з такими розмірностями, тому змінна “Tags” не буде враховуватися при роботі алгоритму.

Змінна “ Consumer complaint narrative”, з назви зрозуміло, що саме показує ця змінна, кількість пропусків якої теж дуже велика, це дає нам розуміння того, що користувачі не горять бажанням залишати додаткову інформації чи якісь уточнення стосовно своєї проблеми, тому при створенні умовної форми для скарг, треба створити якомога більше варіантів, щоб мати краще розуміння проблем користувачів.

Також змінна “ Consumer consent provided” не є цікавою для нас при кластеризації даних за допомогою звичайних математичних засобів вирішення проблеми, так як змінна носить текстовий характер, що дуже ускладнить задачу кластеризації, так як вона не містить умовно невелику кількість унікальних значень, в датасеті наявно більше ста тисяч унікальних значень, що ніяк не годиться для використання її в кластеризаційних моделях, що засновані на звичайних математичних підходах, дана зміна є інформативною та важливою при використанні NLP підходів проте в нашому випадку змінна не є потрібною. Крім того, велика кількість пропусків цієї змінної, взагалі не залишає питання про можливість збереження цієї змінної до моменту запуску моделі.

Така змінна як “Company Id” також не є цікавою для вирішення нашої задачі, так як зараз задача полягає саме в загальному рішенні проблеми та визначенні кластерів проблем, за нею можна розбити наші дані і в подальшому при побудові програмного рішення, порівняти та проаналізувати результати кластеризації для всього датесету та окремо для кожної компанії, що точно дасть цікаві результати.

В додаток до всього раніше зазначеного, змінна “Timely response?” також буде проігнорована нами, через не важливість знань про своєчасність відповіді компанії, зараз нас цікавить саме інформація про скарги. Компанія, штат та зіп-код, також не цікавлять нас, при аналізі скарги, хотілося б розбити проблему на кластери, згідно особливостей скарги, а не компанії чи штату, додам що компанія все ж є важливою, але саме для розділення датасету та аналізу результатів для кожної компанії окремо, все ж таки компанії мають різні продукти, що в свою чергу означає зовсім різні скарги, які логічно розділяться найкращим чином саме за компанією.

В фінальну версію даних увійдуть самі необхідні змінні, що були визначені в ході проведеного аналізу пропусків та доцільності використання змінних, а саме: 'Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via', 'Date sent to company', 'Company response to consumer'. Таким чином, розмірність наших даних, зменшилася з 18 до 7, це значно пришвидшить виконання будь-якого алгоритму, та дозволить краще розбити скарги на кластери, без урахування непотрібної інформації.

	Відсоток пропусків в змінних
Дата отримання	0
Продукт	0
Підпродукт	0
Проблема	0
Надіслано через	0
Дата надіслання до компанії	0
Відповідь компанії споживачу	0

Рисунок 3.6 – Відсоток пропусків змінних в датасеті в оновленому наборі даних

Після проведених дій, наш датасет, з розмірів (670598, 18), має розмірність (472396, 7). Така кількість все ще залишається великою, тому доцільним буде проаналізувати, як можна розділити наші дані, на менші підвибірки, результати в яких можна буде порівняти. Так як в наших даних наявна така змінна як дата, доцільним буде розбити дані саме за неї, поділити їх за роком, можливо навіть зробити розбиття помісячно.

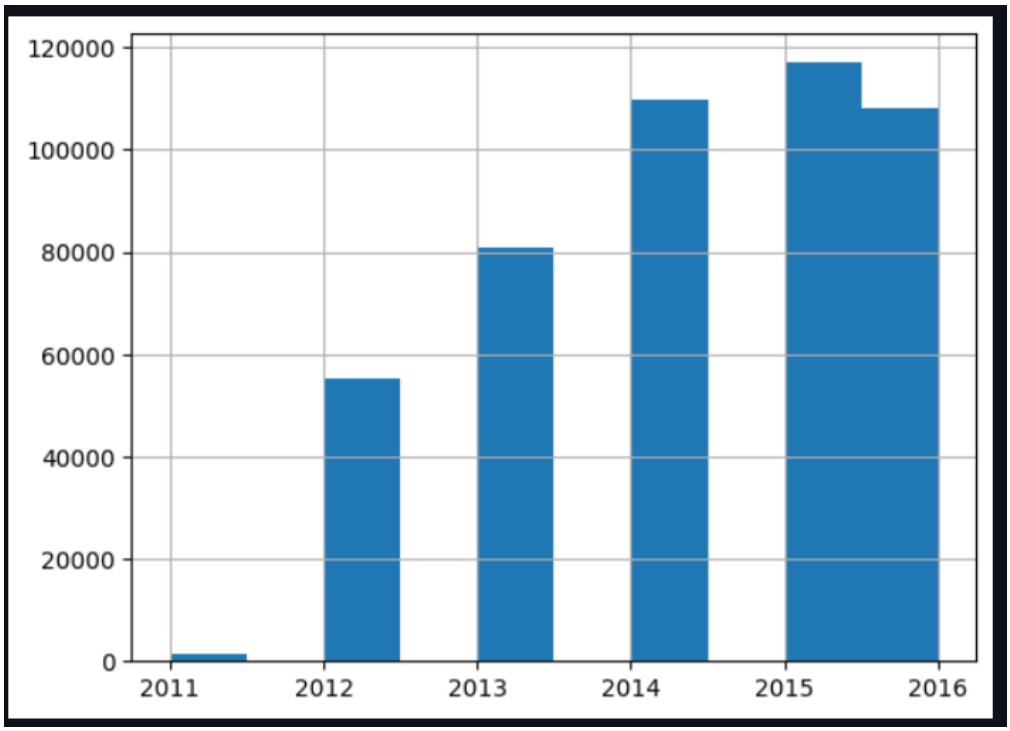

Рисунок 3.7 – Гістограма кількості даних по рокам

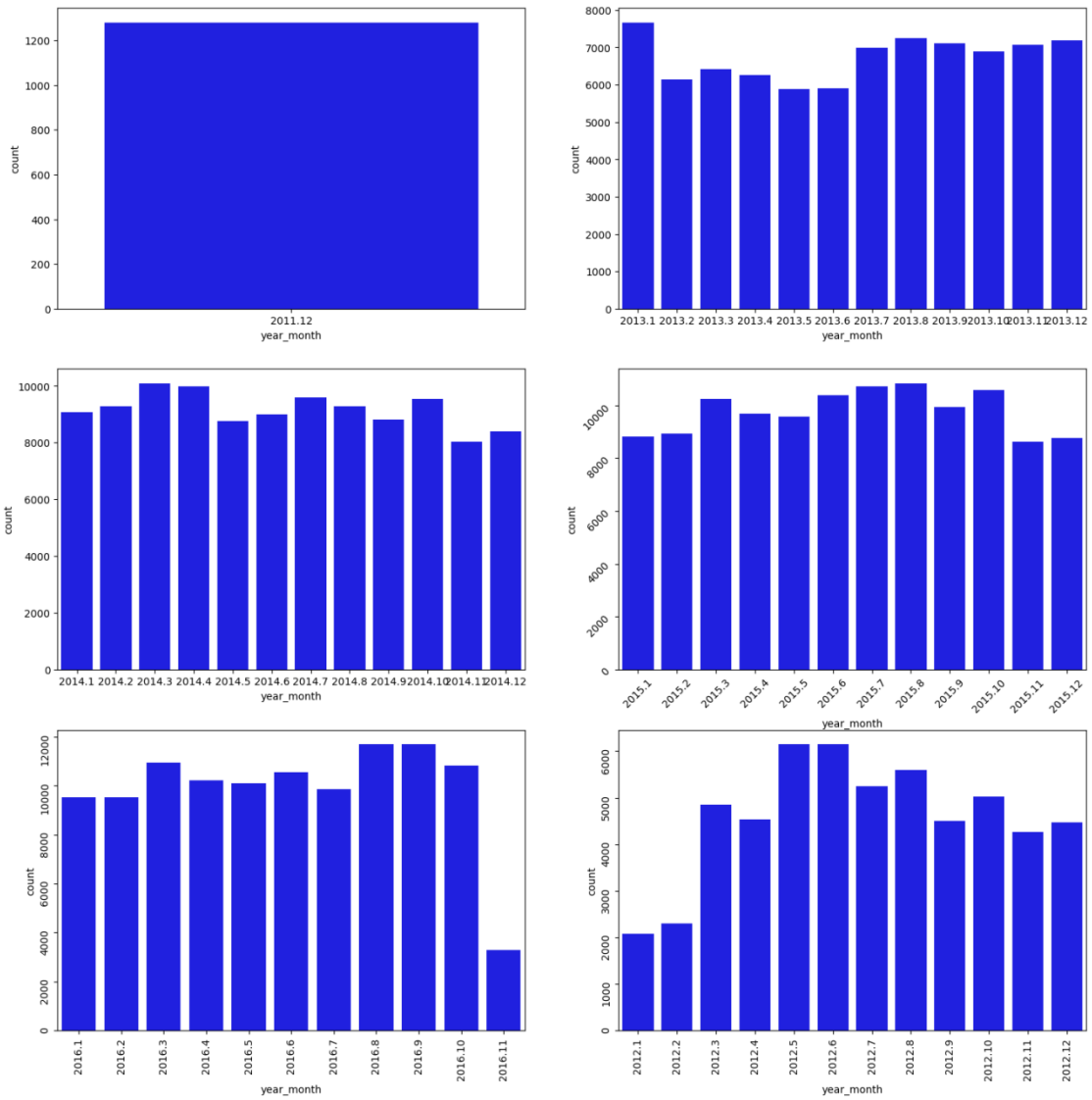

Рисунок 3.8 – Гістограми кількості даних по місяцю в рік

З побудованих гістограм, ми бачимо скільки даних приходить на один місяць в середньому, тому думаю доцільно буде розділити дані на вибірки помісячно, щоб алгоритм швидше та якісніше сходився, також за рахунок цього, можна буде порівняти, наскільки з часом змінюються наші кластери і чи змінюються взагалі, це буде свідчити про те, чи вирішуються компаніями їх проблеми, на які скаржаться їх клієнти. Так як вибірка за 2011 рік складає всього нічого, думаю буде правильним рішенням, викинути його з наших даних.

Також важливим є розбиття за компанією, так як датасет містить майже 4 тисячі різних компаній.

	Кількість унікальних компаній
0	3933

Рисунок 3.9 – Кількість унікальних компаній

Тому логічно, що компанії про які наявно дуже мало записів, не є цікавими для нас, так як їх розбиття на кластери взагалі не має сенсу. Тому цікаво подивитися, на які компанії скільки скарг припадає та в загальному поглянути на доцільність такого розбиття з подальшою обробкою даних на основі аналізу.

Company	
Bank of America	53403
Wells Fargo & Company	45216
JPMorgan Chase & Co.	29564
Ocwen	22911
Citibank	16160
Nationstar Mortgage	14974
Ditech Financial LLC	10474
Navient Solutions, Inc.	10267
U.S. Bancorp	8920
PNC Bank N.A.	7473
Encore Capital Group	6903
Capital One	6182
HSBC North America Holdings Inc.	6128
SunTrust Banks, Inc.	5432
Select Portfolio Servicing, Inc	5190
TD Bank US Holding Company	4175
Portfolio Recovery Associates, Inc.	4148
Synchrony Financial	3548
Citizens Financial Group, Inc.	3415
Seterus, Inc.	3355

Рисунок 3.10 – Топ 20 компаній за кількістю записів

З рисунку явно видно, що тільки топ 20 компаній вже містить істотну частину записів, тому доречним буде дослідити статистичні властивості змінної кількості записів на компанію.

	Опис змінної Company
count	3933.000000
mean	170.505467
std	1963.461690
min	1.000000
25%	2.000000
50%	6.000000
75%	25.000000
max	61720.000000

Рисунок 3.11 – Статистичні властивості змінної

На малюнку явно видно, що 75% записів, містить менше 25 записів в датасеті, це дуже псує отримані статистичні дані, тому вважаю доречним поглянути на значення, що вищі за 75 процентиль.

	Опис змінної Company вищої за 75 процентиль
count	962.000000
mean	678.991684
std	3928.231629
min	26.000000
25%	39.000000
50%	67.000000
75%	178.500000
max	61720.000000

Рисунок 3.12 – Статистичні властивості змінної вищої за 75 процентиль

Кількість записів все ще мала як для 75 процентиля, тому вважаю доречним відібрати тільки ті компаній, в яких більше 1000 записів, так як для подальшого аналізу, тисячі компаній просто нереально проаналізувати, а їх мала кількість записів

взагалі не вказує на високу ефективність про навчанні моделі. Тому для детального аналізу буде взято топ 10 компаній за кількістю записів про них.

Company	
Bank of America	53403
Wells Fargo & Company	45216
JPMorgan Chase & Co.	29564
Ocwen	22911
Citibank	16160
Nationstar Mortgage	14974
Ditech Financial LLC	10474
Navient Solutions, Inc.	10267
U.S. Bancorp	8920
PNC Bank N.A.	7473

Рисунок 3.13 – Топ 10 компаній за кількістю записів

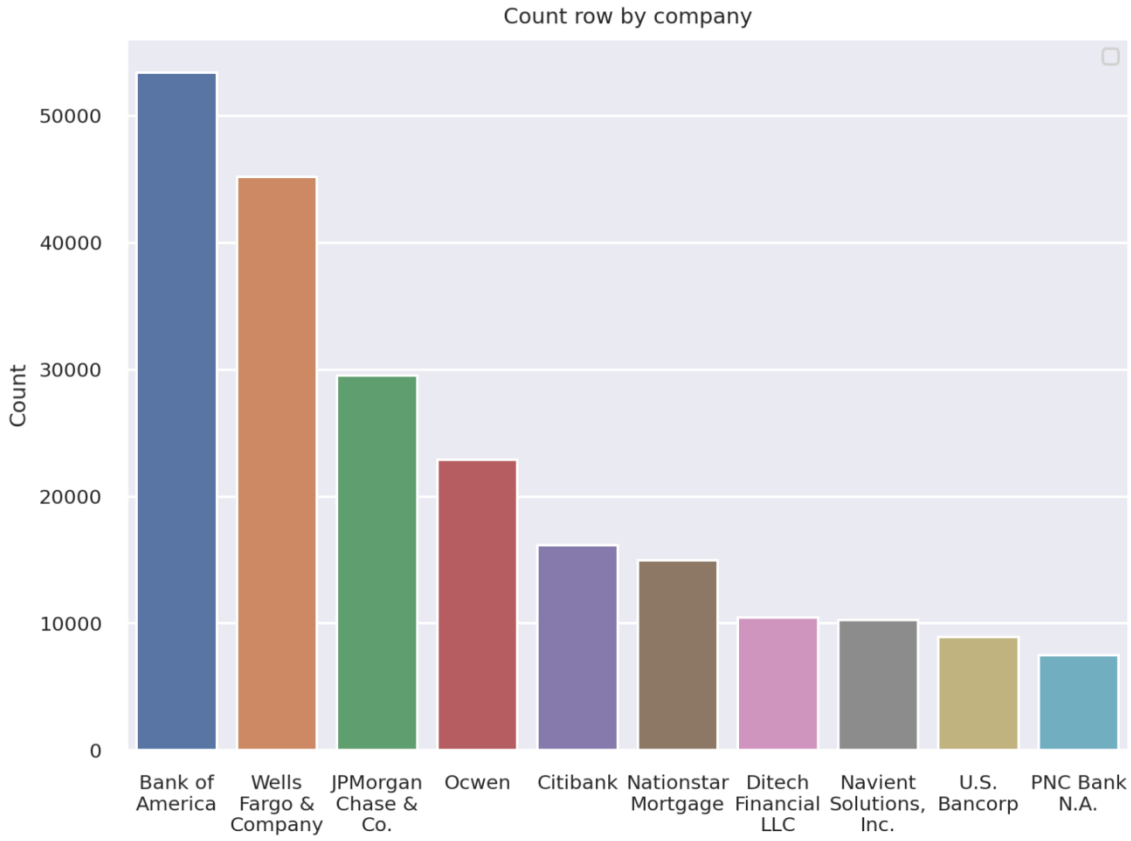

Рисунок 3.14 – Гістограма розподілу топ 10 компаній за кількістю записів

3.3 Тестовий приклад роботи програми

Під час роботи з програмою, користувач буде мати справу з вікном.

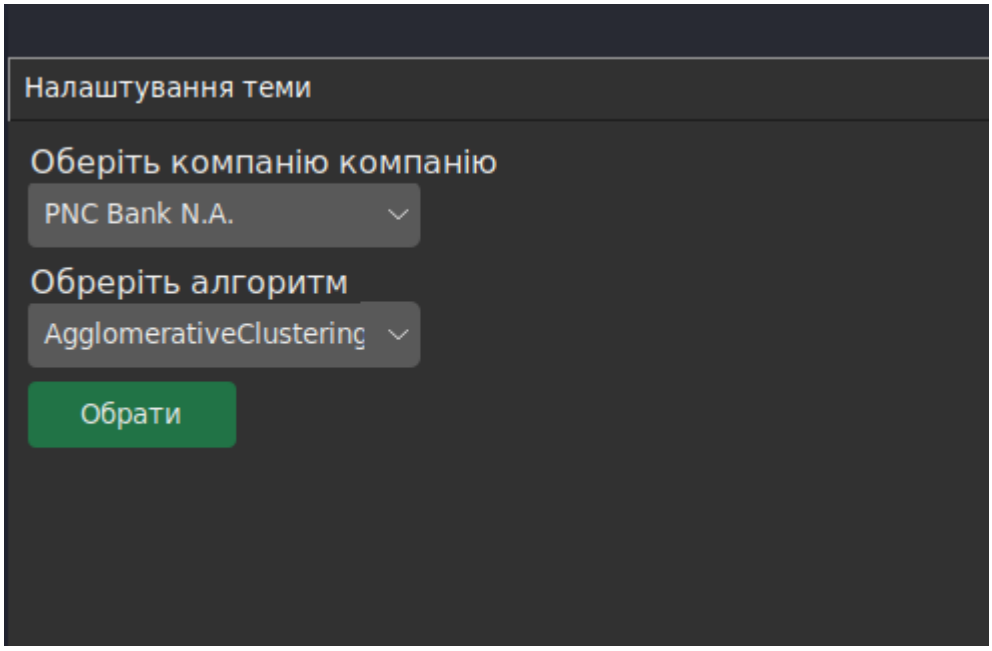

Рисунок 3.15 – Графічний інтерфейс користувача

Спочатку вікно надає можливість користувачу обрати компанію, дані про яку хоче дослідити та алгоритм кластеризації, що хоче використати.

Далі в залежності від обраного алгоритму, програма пропонує користувача задати параметри роботи алгоритму. До прикладу у випадку, коли користувач, в якості алгоритму кластеризації обрав “K-modes”, користувач побачить наступний екран.

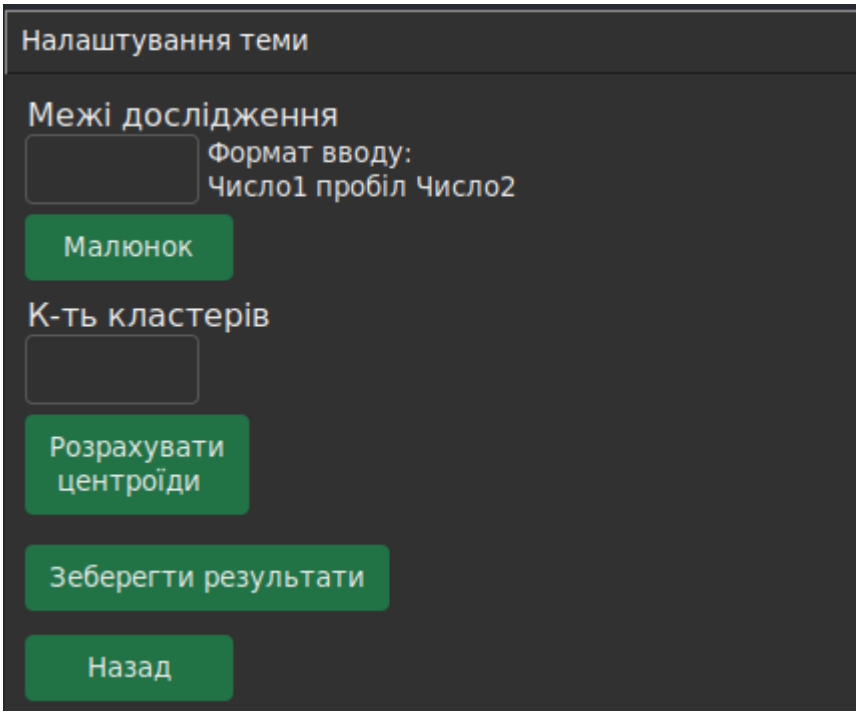

Рисунок 3.16 – Налаштування параметрів “K-modes”

Тут ми маємо можливість обрахувати центроїди алгоритму згідно з обраними даними, зазначивши кількість кластерів, на яку ми хочемо розбити наш набір даних, або використати метод ліктя, описаний в підрозділі 2.4, надавши йому межі дослідження.

Формат вводу даних описаний на малюнку 4.2. При натисканні на кнопку “Малюнок” програма почне навчати моделі від різної кількості кластерів, умовно від 5 до 10 і в результаті роботи, виведе на екран графік залежності функції вартості від кількості кластерів, простіше кажучи, ця функція дозволяє користувачу, побачити графік, щоб за допомогою методу ліктя, визначити оптимальну кількість кластерів.

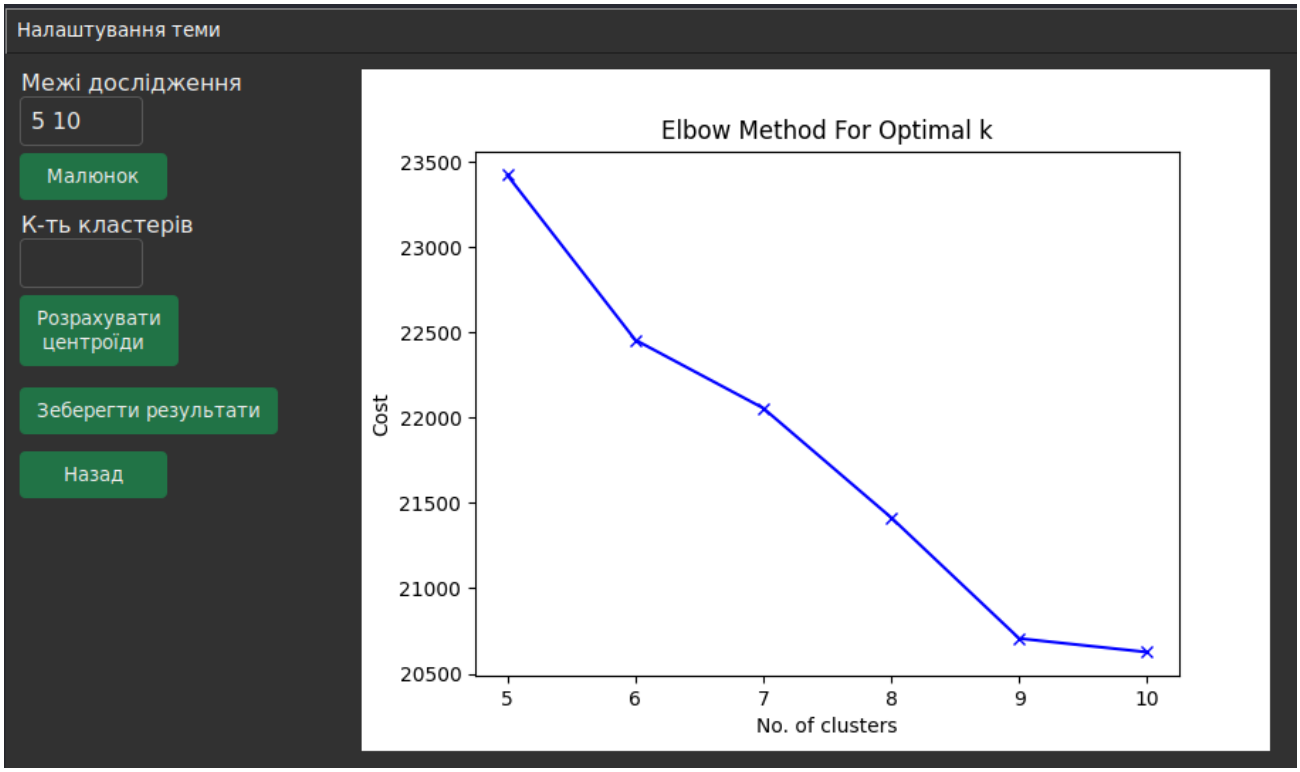

Рисунок 3.17 – Результат виконання функції малюнок

З побудованого графіку, видно, що на проміжку від 5 до 10 кластерів, не має явного моменту перелому функції вартості, але явно видно, що після кількості кластерів рівної 9, швидкість падіння значення функції сильно зменшується, тому з даного малюнку можна зробити висновок, про оптимальність кількості кластерів рівній 9.

Надалі користувач може використати функцію “Розрахувати центроїди” і задати свою фіксовану кількість кластерів та отримати наступні результати.

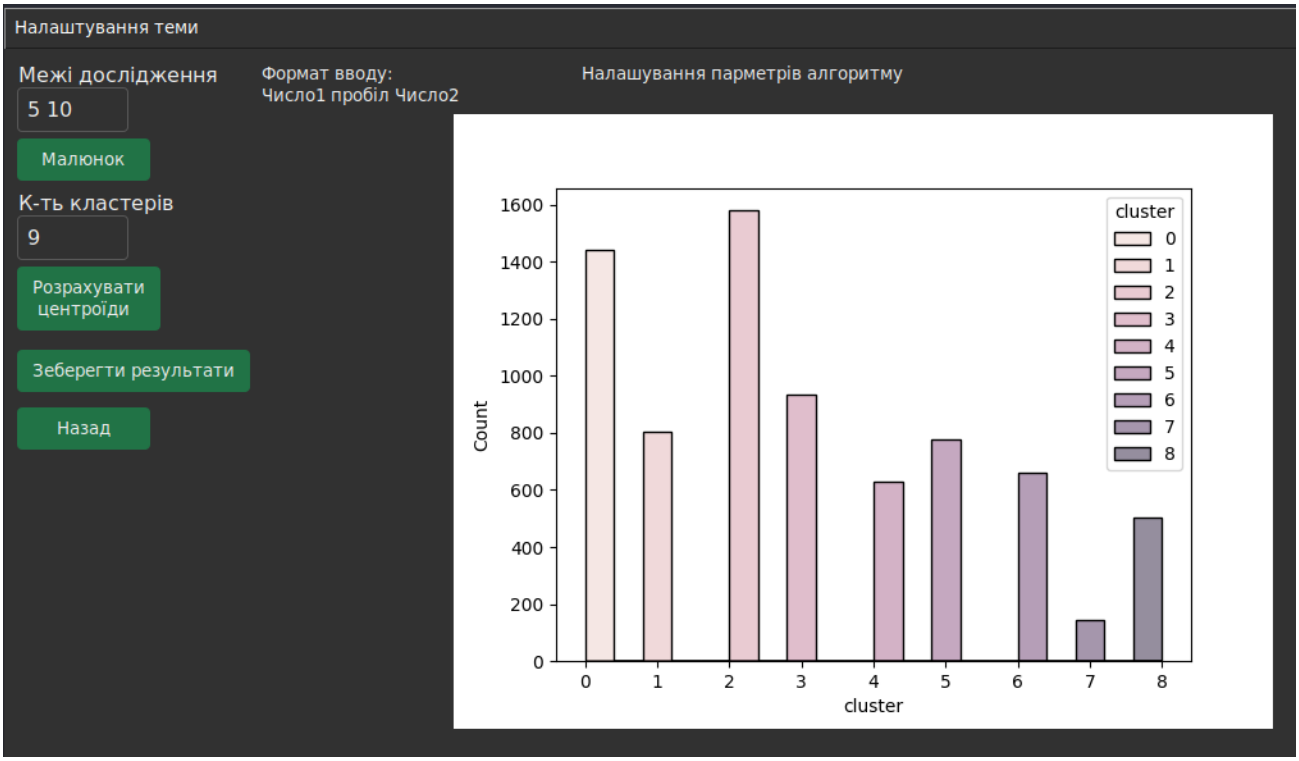

Рисунок 3.18 – Результат роботи функції “Розрахувати центроїди”

Тепер користувач може бачити розподіл кількості спостережень в кожному кластері. Якщо користувач задоволений побаченою картиною, він має змогу зберегти обраховані дані, використавши функцію “Зберегти результати”. Дана функція збереже в окрему теку загальний файл за обраною компанією з мітками кластерів, що відповідають спостереженням, також буде створено окремі файли в кожному з яких будуть скарги тільки з окремого кластеру. Це дозволить зручно аналізувати отримані результати, працівникам підтримки чи аналітикам, щоб визначити значущість кластеризації та робити висновки на основі обрахованих кластерів. Самі файли залежать від обраного алгоритму, кожен алгоритм зберігає свої результати роботи окремо. В свою чергу, збереження результатів, дозволить використовувати їх в подальшому, наприклад для написання нових програм для використання вже обрахованих центроїдів в кластерному аналізі чи аналіз саме розмічених даних.

Аналогічні екрани можна побачити при виконанні інших алгоритмів, до прикладу обравши ієрархічну кластеризацію, користувач матиме змогу побудувати дерево зазначеної ним глибини, щоб визначити оптимальну кількість кластерів, саме

для цього алгоритму.

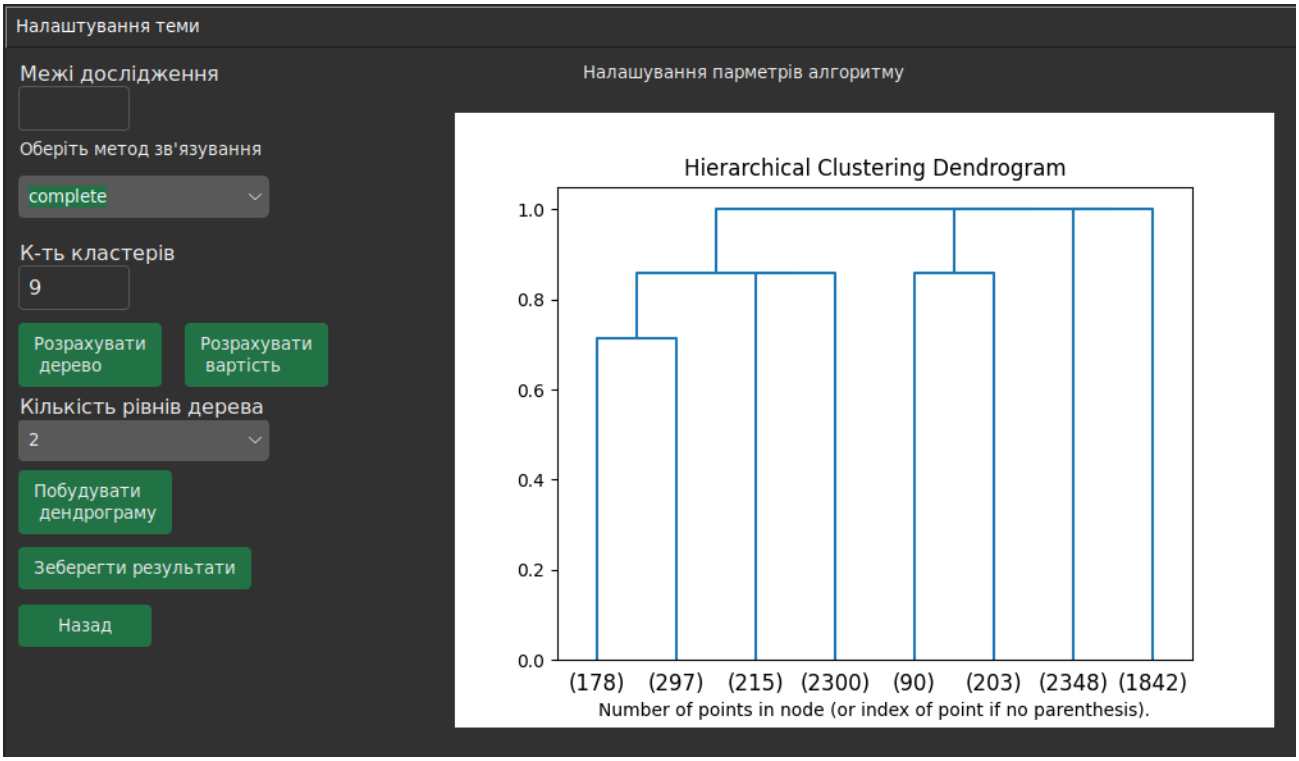

Рисунок 3.19 Дерево розбиття за методом ієрархічної кластеризації, згідно обраних параметрів глибини та зв’язування

До прикладу в цьому прикладі, можемо бачити, скільки та на якому кроці наше дерево має гілок та скільки спостережень належать цим гілкам. За рахунок чого, ми можемо роботи висновки про оптимальну кількість кластерів саме для цього алгоритму, на даному прикладі, розбиття на 4 кластери є вигідним, або ж за бажання виокремити підкластери, можна обрати 8 кластерів, що дозволить детальніше дослідити менші кластери та отримати більш точні результати.

Також є інший підхід визначення оптимальної кількості кластерів, що реалізує метод ліктя та виводить наступні результати.

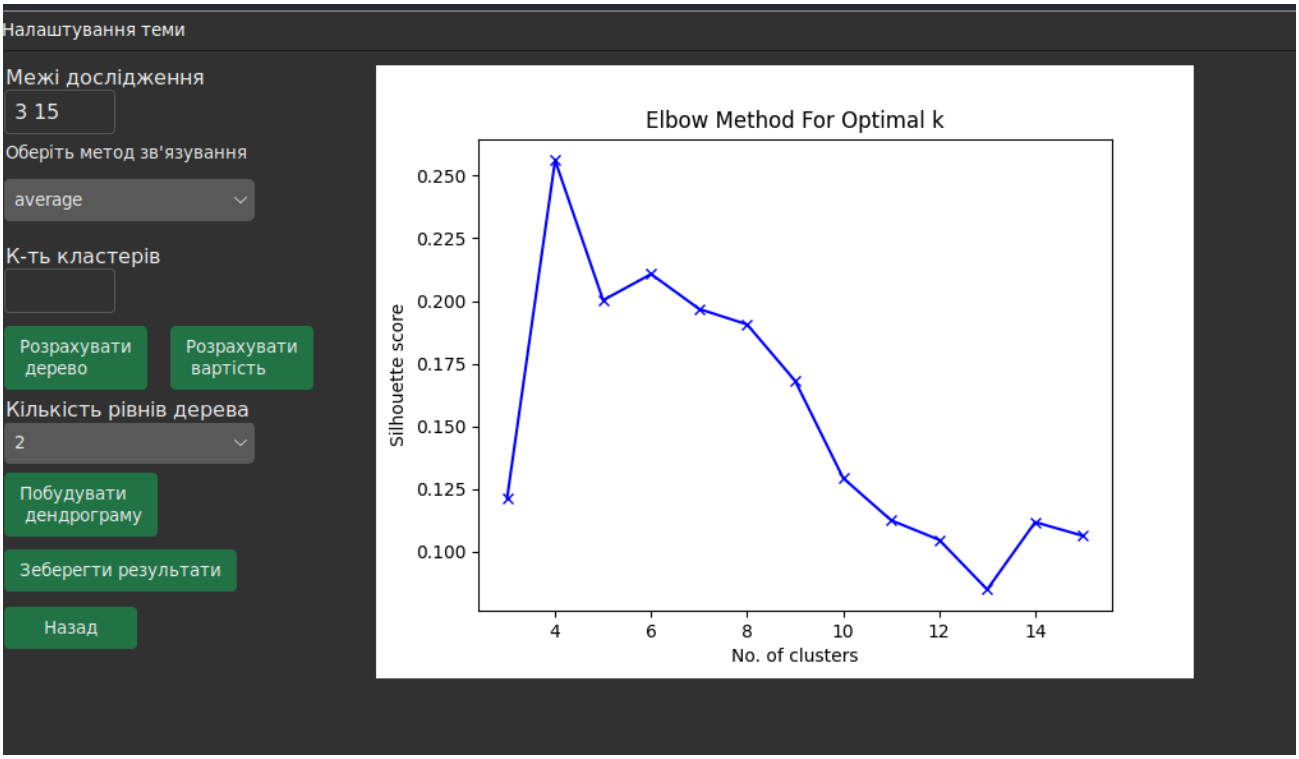

Рисунок 3.20 – Графік залежності коефіцієнту вартості від обраної кількості кластерів

В даному випадку обраховується коефіцієнт силуету кластеризації описаний в підрозділі 2.4.2, згідно логіки його роботи, з графіку можна зробити висновок про оптимальну кількість кластерів у проміжку від 5 до 7.

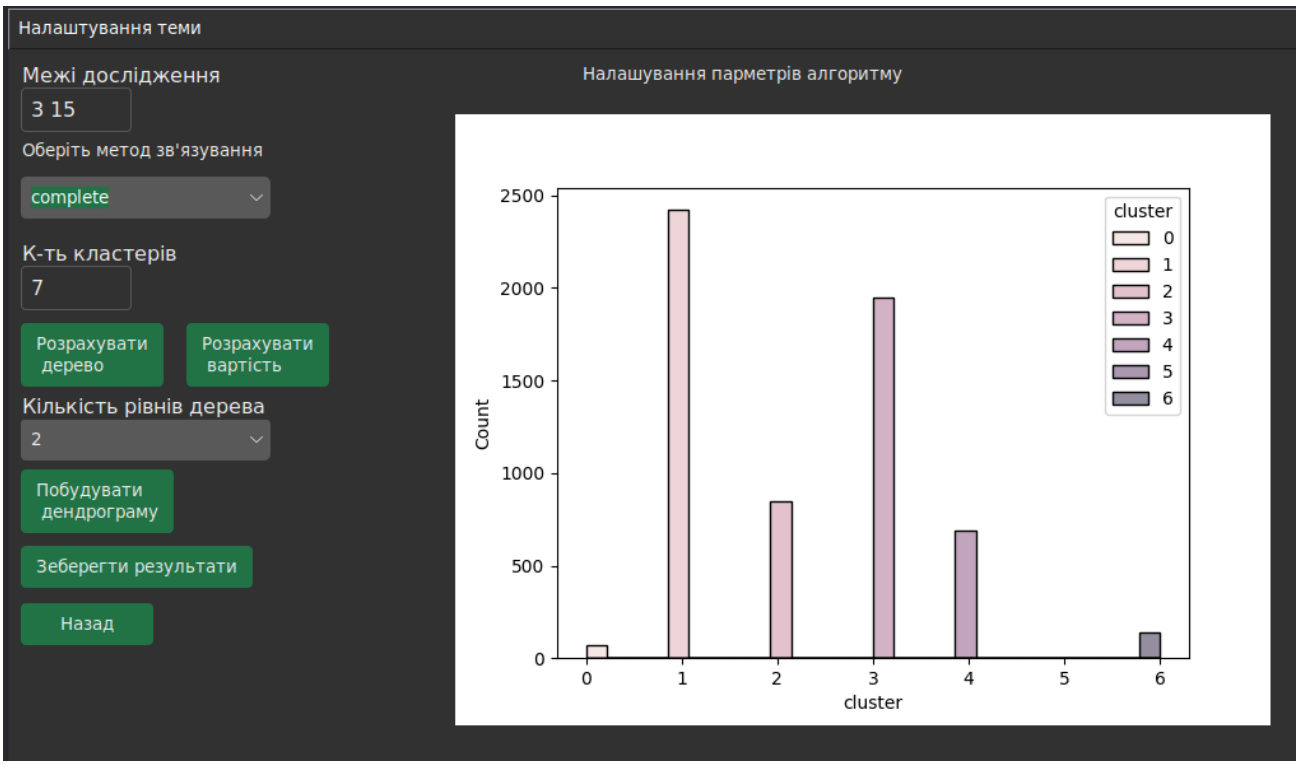

Рисунок 3.21 – Результат ієрархічної кластеризації

Картина аналогічна минулому алгоритму, на екрані продемонстровано гістограму кількості спостережень до номеру кластера. Також зберігаються дані для подальшого аналізу.

Наступним є алгоритм DBSCAN, що має такі параметри як епсілон, що визначає радіус кола, в якому ми оглядаємо наші точки та мінімальна кількість точок для утворення кластеру.

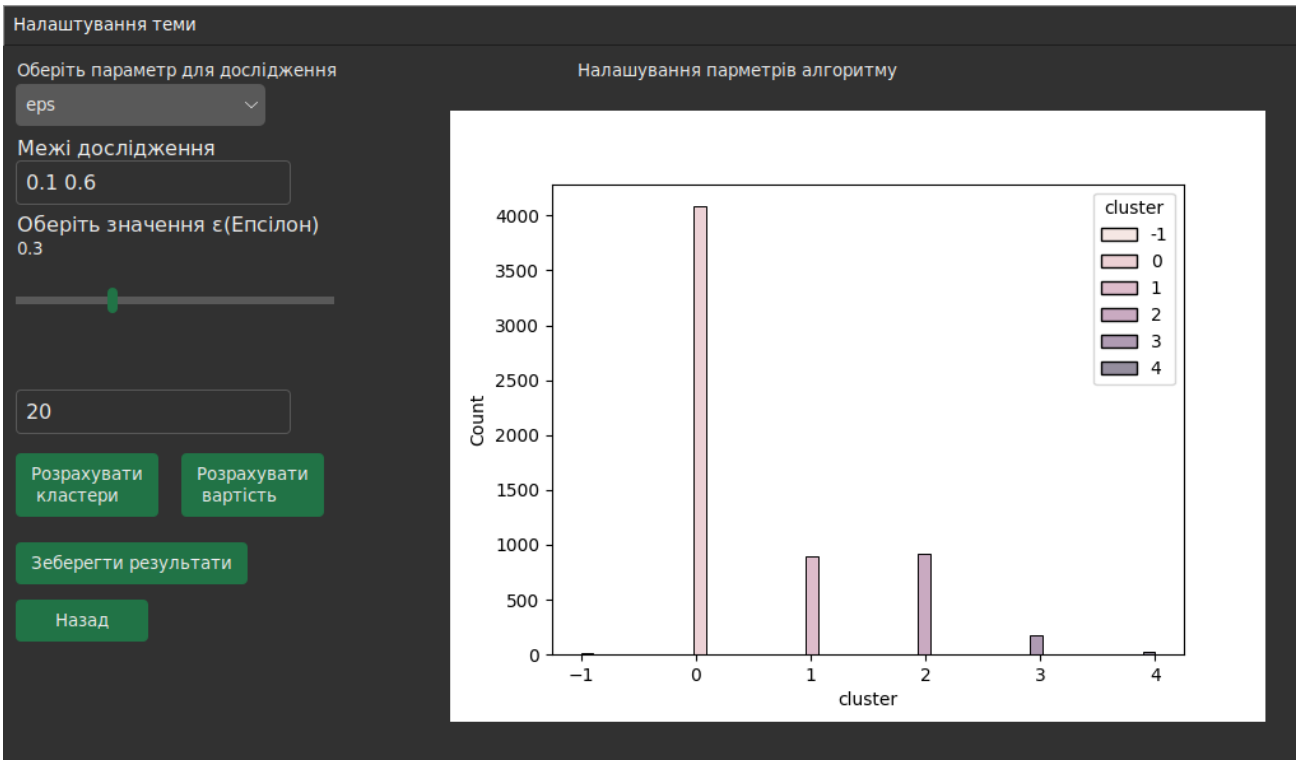

Рисунок 3.22 – Результат виконання алгоритму DBSCAN

В цьому випадку до особливостей додається той факт, що ми не маємо заздалегідь визначеної кількості кластерів, тому вибір параметрів є конче важливим для отримання адекватних результатів, в підрозділі 2.7, присвяченому цьому алгоритму було описано методи визначення оптимальних значень параметрів алгоритму, тому тут важливим є саме досвід використання та аналіз кількість кластерів, що отримується в ході виконання, позаяк правильно візуалізувати наші категоріальні дані великої розмірності ми не можемо.

Саме для визначення цих параметрів, ми можемо дослідити тенденції зміни коефіцієнту силуету, що дозволить зрозуміти в якому напрямку ми хочемо рухатись.

Рисунок 3.23 – Результат дослідження параметру епсілон

Тут ми можемо бачити, що збільшення параметра епсілон, призводить до покращення коефіцієнту та зменшення кількості кластерів, що якраз і є логічним висновком з алгоритму роботу обраного методу, що описаний в підрозділі 2.7. Кількість графіків у випадку дослідження параметрів алгоритму DBSCAN є більшою, так як саме від них залежить кількість кластерів, що знайде алгоритм, а отже ці тенденції теж є вкрай важливими для користувача.

В свою чергу алгоритм HDBSCAN, майже всі свої параметри встановлює власноруч і для нього важливими є мінімальна кількість спостережень для створення кластера та мінімальний розмір кластера, інші параметри він здатен встановити сам з дерева відстаней, що будує власноруч.

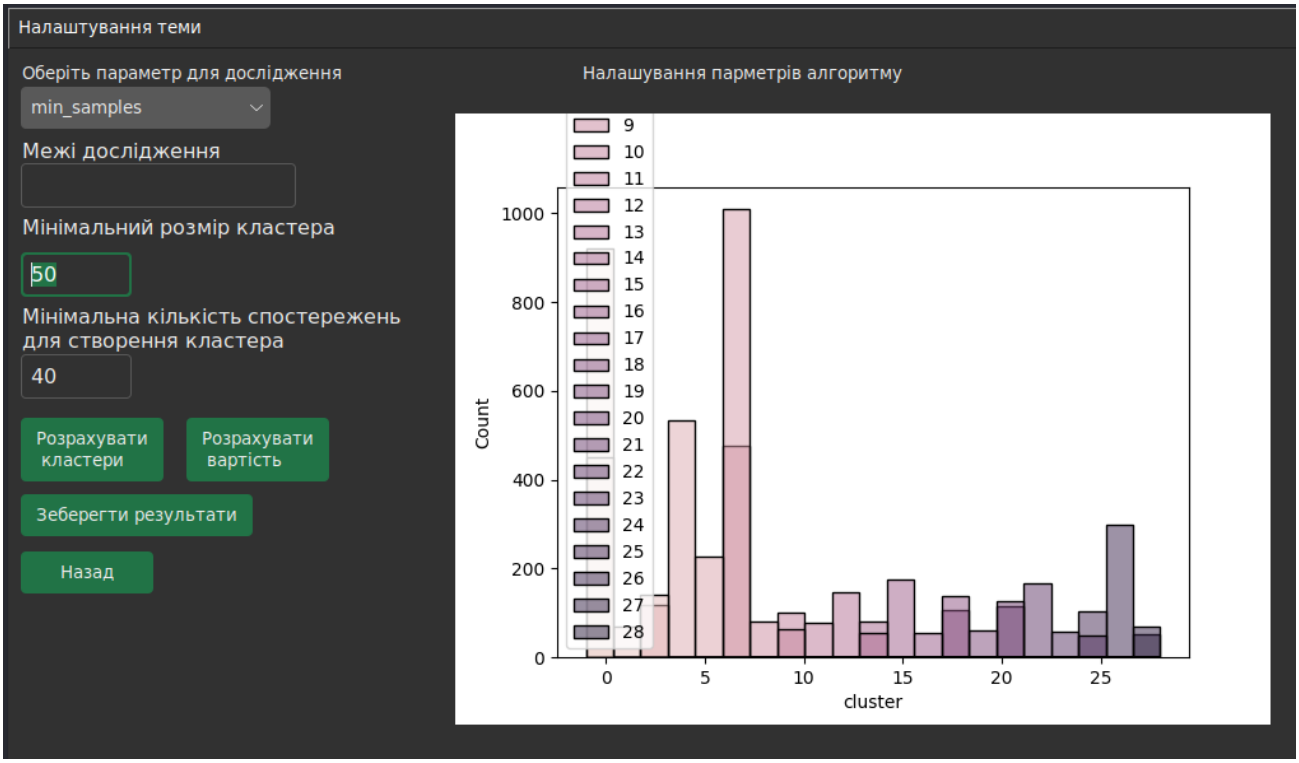

Рисунок 3.24 – Результат виконання алгоритму HDBSCAN

Сам алгоритм має досить заплутаний набір параметрів, але сенс цього є наступним, мінімальний розмір кластеру, сильно зрізає можливості розбиття даних, роблячи умови дуже жорсткими, в свою чергу якщо власноруч зменшити мінімальну кількість спостережень для створення кластера ми отримаємо не такі жорсткі рамки, а саме меншу кількість даних, що будуть визначені алгоритмом як викиди. Сам алгоритм, як можемо бачити з прикладу, знаходить достатньо велику кількість малих кластерів, що є досить корисним на практиці, крім цього самі кластери є дуже схожими за розміром, тож вони більш ніж підходять для комфортного аналізу.

Для встановлення оптимальних значень параметрів та з’ясування тенденцій зміни коефіцієнту силуету, додано функцію дослідження проміжків значень, як і в алгоритмі DBSCAN, користувач бачить не лише зв’язок досліджуваного параметра з коефіцієнтом, а і кількість кластерів, що знаходить алгоритм.

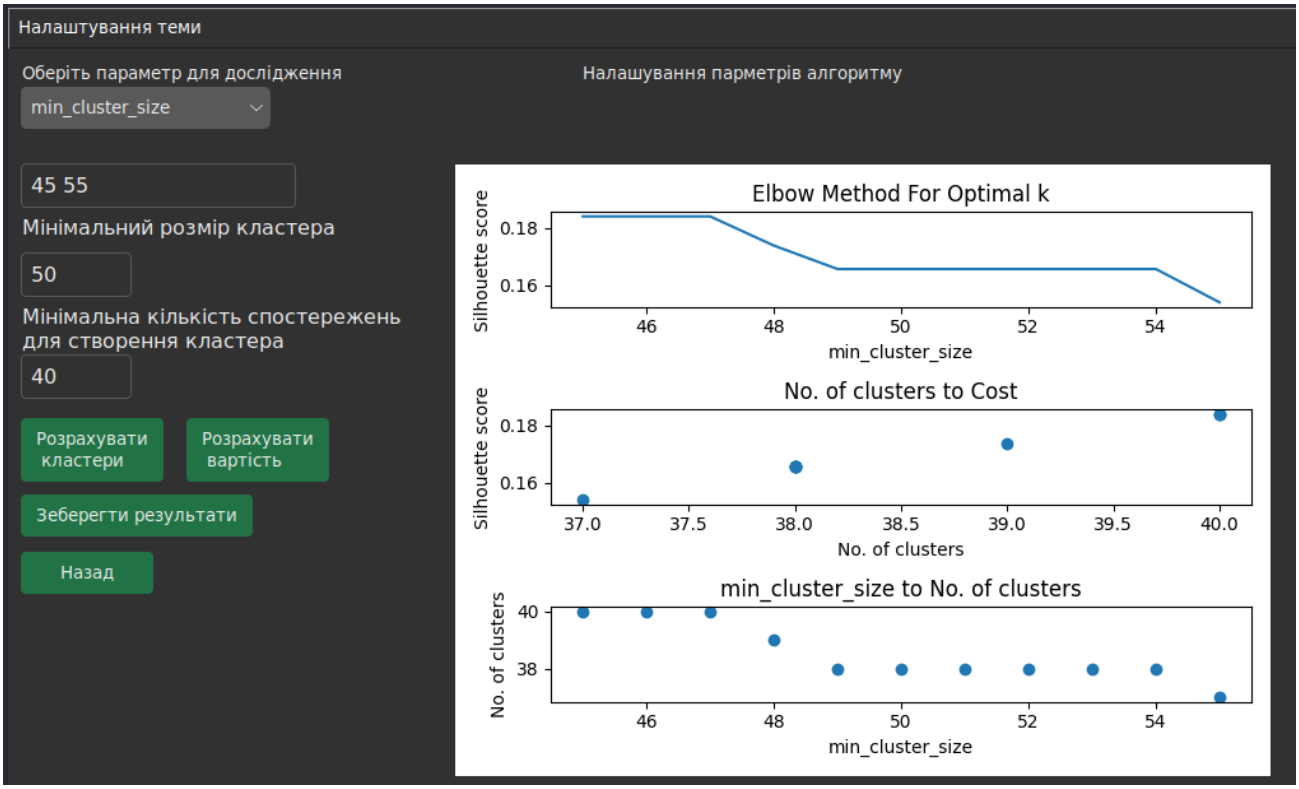

Рисунок 3.25 — Результат дослідження параметра алгоритму HDBSCAN

Ми можемо бачити, як збільшення мінімальної кількості спостережень для створення кластеру, призводить до зменшення кількості кластерів, в свою чергу коефіцієнт силуету теж спадає, але за аналогією з методом ліктя, ми шукаємо саме оптимум тому втрата декількох сотих не є небезпечною.

3.4 Аналіз отриманих результатів

Після виконання кластеризації користувач отримує файли з результатами виконання алгоритмів. Цікаво буде дослідити які саме дані увійшли в кластери і проаналізувати ефективність кластеризації, чи має взагалі це якийсь сенс і чи були вловлені якісь залежності.

Під час тестування роботи програми, було виконано багато спроб, які в результаті надали нам набір файлів аналізу, для прикладу візьмемо результат отриманий алгоритмом HDBSCAN для даних компанії “TD Bank US Holding Company”.

Рисунок 3.26 — Відсоткове співвідношення по кількості даних в кластері

З даного графіку ми бачимо, що дані було розділені на 21 кластер, більшість з яких містить приблизно однакову кількість даних, тільки кластер з міткою 7 має аж 17 з половиною відсотків скарг з наявних. Враховуючи особливості роботи алгоритму HDBSCAN, які було описано в підрозділі 2.4.8, можемо припустити, що кластери з аномально малою кількістю даних, були визначені як викиди, що не схожі на більшість інших записів наданих компанією.

Рисунок 3.27 — Графік кількості унікальних продуктів в кластері

Даний графік в демонструє, що -1 кластер це просто якась намішана купа різних продуктів, як і 0 та 1 кластер, на минулому малюнку ми бачили, що розміри цих кластерів теж є не великими, тому ймовірно це кластери, що є викидами, скарги які не вдалося віднести до чогось конкретного і вони сформували свій, окремий кластер, що містить в собі буквально все. В свою чергу кластери під номерами 8, 17, 18, 19, які містили більше даних, гарно проходять цю перевірку та показують, що для всіх даних, які увійшли до цих кластерів, є спільна риса і вона має унікальну природу. Найбільший кластер під номер 7, взагалі містить тільки один продукт, що вже дає змогу визначити, найбільш проблемний продукт компанії. Але також на графіку видно такий приклад як кластер з міткою 2, він вміщує велику частину даних, але крім цього також має аж 5 унікальних продуктів в собі. Тому доречним буде проаналізувати, як себе поводять інші параметри, все ж таки один продукт може містити багато підпродуктів, а також різні продукти можуть бути об’єднані одним типом скарг на ці самі продукти.

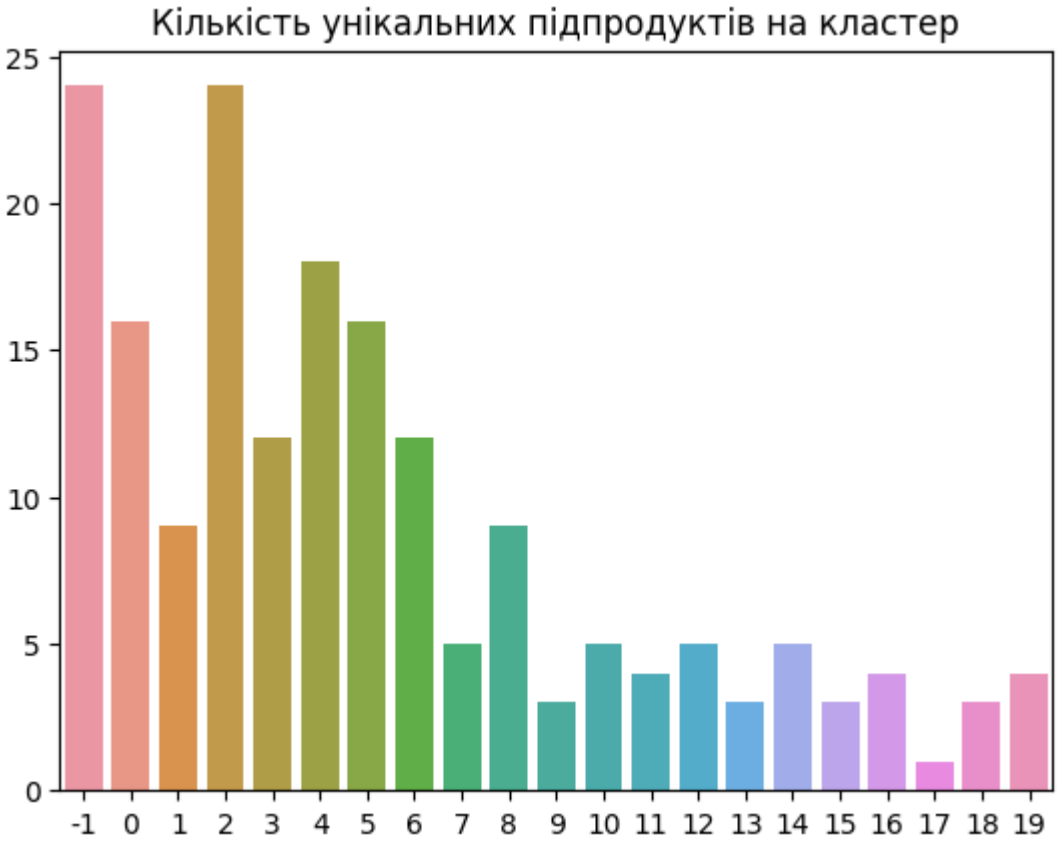

Рисунок 3.28 – Графік кількості унікальних підпродуктів в кластері

Продовжуючи аналіз результатів вже за унікальними підпродуктами в кластері, ми бачимо, що ймовірніше всього 2 кластер, не дивлячись на свої розміри, все ж є викидом, враховуючи як багато різноманітних значень входять до цього кластеру, підозри стосовно кластерів -1 та 0 теж підтверджуються. А от проходять перевірку на міцність все ті ж кластери 7, 8, 17, 18, 19, в них спостерігається тенденція до збереження саме конкретних значень підпродуктів. Тепер важливим є перевірка стану саме змінної, що позначає скарги.

Рисунок 3.29 – Графік кількості унікальних скарг в кластері

Картина на диво схожа до попередніх, знову ті ж самі кластери мають величезне різноманіття значень, що входять до них, а стабільні кластери як і раніше поводять себе стримано та всім своїм виглядом показують, що вони тут мають сенс. Тепер вже точно можна робити висновки про те, як алгоритм відпрацював в даному прикладі, а саме, кластери з -1 по 6, є викидами, на це вказує неймовірне різноманіття унікальних значень параметрів в цих кластерах. Власне викид не є чимось поганим, це такі ж скарги клієнтів, просто ми не можемо виокремити спільні риси серед цих скарг. Натомість ми можемо сконцентрувати свій погляд на такі кластери як 7, що в даному прикладі є взірцем кластеру. Він містить велику кількість даних та поводиться з ними пристойно, кажучи, що є 5 типів скарг, які зустрічаються саме одному конкретному продукті компанії, на який припадає цілих 17 з половиною відсотків всіх наявних скарг наших користувачів, також кластер зазначає в яких саме підпродуктах зустрічаються ці проблеми.

Ці результати вдалося отримати навіть не зазираючи в самі скарги наявні в

цьому кластері, аналогічні висновки легко можна зробити і в інших кластерах. Тепер надавши ці результати аналітику компанії, який знайомий з тим, які саме це продукти, де саме можуть виникати ці проблеми і який загалом є експертом в даній області, він легко зможе зрозуміти, як саме буде краще вчинити з отриманою інформацією, що дуже легко представити в графічному вигляді, який є зручним для розуміння людини та для чого не знадобиться мати якісь додаткові знання чи вміння, щоб власне отримати користь від проробленої роботи.

3.5 Висновки

Підсумовуючи, розробка програми для кластеризації скарг щодо фінансових послуг виявилася цінним інструментом для аналізу та розуміння скарг клієнтів у фінансовій галузі. Додаток успішно реалізував алгоритми кластеризації для групування подібних скарг, виявлення шаблонів і спільних тем.

За допомогою програми постачальники фінансових послуг можуть отримати уявлення про головні больові точки, з якими стикаються їхні клієнти. Здатність ідентифікувати окремі кластери скарг, як-от помилки виставлення рахунків, проблеми з обслуговуванням клієнтів або шахрайські дії, дозволяє організаціям зосередити свої ресурси на вирішенні конкретних проблемних питань. Використовуючи програму, постачальники фінансових послуг можуть покращити процеси обслуговування клієнтів, оптимізувати внутрішні операції та приймати обґрунтовані рішення для підвищення загальної задоволеності клієнтів.

Крім того, зручний інтерфейс програми та інтуїтивно зрозумілі функції роблять її доступною та простою у використанні як для аналітиків, так і для осіб, які приймають рішення. Візуальне представлення тенденцій роботи алгоритмів забезпечує чітке розуміння, чого саме і як прагне досягти користувач. Важливо відзначити, що успіх програми залежить від якості та точності даних скарги. Таким чином, для забезпечення актуальної та достовірної інформації мають бути постійні процеси збору та уточнення даних.

Підсумовуючи, розробка програми для кластеризації скарг щодо фінансових послуг дає змогу організаціям краще розуміти та вирішувати проблеми клієнтів. Використовуючи знання, отримані за допомогою програми, постачальники фінансових послуг можуть покращити свою діяльність, підвищити рівень задоволеності клієнтів і побудувати міцніші відносини зі своєю клієнтурою. Програма є цінним інструментом для сприяння позитивним змінам і надання виняткового досвіду клієнтам у сфері фінансових послуг.

ДОДАТКИ

Програмний код:

```
# import temp
import tkinter as tk
from tkinter import ttk
import sv_ttk
from tkinter.messagebox import showinfo
from alg import get_optK
from alg import get_agg
from alg import make_dendrogram
from alg import get_dbscan
from alg import get_hdbscan
from alg import agg_plot
from alg import dbscan_plot
from alg import hdbscan_plot
import os
import datetime


# import matplotlib
import matplotlib.pyplot as plt
# import PIL
from PIL import ImageTk, Image

import pandas as pd
# import kourse_w


root = tk.Tk()
root.tk.call('source', 'styles/Forest-ttk-theme-master/forest-dark.tcl')
root.tk.call('source', 'styles/Forest-ttk-theme-master/forest-light.tcl')
ttk.Style().theme_use('forest-dark')
# sv_ttk.set_theme("dark")
menubar = tk.Menu(root, background='#313131', foreground='white')

df_result = pd.DataFrame([], columns=['cluster'])

def change_theme_light():
    ttk.Style().theme_use('forest-light')
    menubar.config(background='white', foreground='black')


def change_theme_dark():
    ttk.Style().theme_use('forest-dark')
    menubar.config(background='#313131', foreground='white')


root.config(menu=menubar)
file_menu = tk.Menu(menubar, tearoff=False)
file_menu.add_command(
    label='Використати світлу тему',
    command=change_theme_light
)

file_menu.add_command(
    label='Використати темну тему',
    command=change_theme_dark
)
```

```
menubar.add_cascade(
   label="Налаштування теми",
   menu=file_menu
)

canvas1 = tk.Canvas(root, width=1080, height=720)
canvas1.pack()

# fig = plt.figure()

# matplotlib.use('TkAgg')

# frame = tk.Frame(root)
# frame.place(relwidth=1,relheight=1)

frame = ttk.Frame(root)
frame.place(relwidth=1, relheight=1)

# frame.grid(row=0, column=0, padx=(20, 10), pady=(20, 10), sticky="nsew")

entry_mas = []
label_mas = []
# GlRange=kourse_w.Global_range()
company_list = ['Ditech Financial LLC',
            'Navient Solutions, Inc.',
            'U.S. Bancorp',
            'PNC Bank N.A.',
            'Encore Capital Group',
            'Capital One',
            'HSBC North America Holdings Inc.',
            'SunTrust Banks, Inc.',
            'Select Portfolio Servicing, Inc',
            'TD Bank US Holding Company']
selected_company = tk.StringVar()
combobox_company = ttk.Combobox(frame, textvariable=selected_company)
combobox_company['values'] = company_list
combobox_company['state'] = 'readonly'
combobox_company.set('PNC Bank N.A.')

p_list = [2, 3, 4]
selected_p = tk.IntVar()
combobox_p = ttk.Combobox(frame, textvariable=selected_p)
combobox_p['values'] = p_list
combobox_p['state'] = 'readonly'
combobox_p.set(2)

algorithms = ['K-Modes', 'AgglomerativeClustering', 'DBSCAN', 'HDBSCAN']
selected_algorithm = tk.StringVar()
combobox_algorithms = ttk.Combobox(frame, textvariable=selected_algorithm)
combobox_algorithms['values'] = algorithms
combobox_algorithms['state'] = 'readonly'
combobox_algorithms.set('AgglomerativeClustering')

param_dbscan = ['min_samples', 'eps']
selected_param_dbscan = tk.StringVar()
combobox_param_dbscan = ttk.Combobox(frame, textvariable=selected_param_dbscan)
combobox_param_dbscan['values'] = param_dbscan
combobox_param_dbscan['state'] = 'readonly'
combobox_param_dbscan.set('min_samples')
```

```
param_hdbscan = ['min_samples', 'min_cluster_size']
selected_param_hdbscan = tk.StringVar()
combobox_param_hdbscan = ttk.Combobox(frame, textvariable=selected_param_hdbscan)
combobox_param_hdbscan['values'] = param_hdbscan
combobox_param_hdbscan['state'] = 'readonly'
combobox_param_hdbscan.set('min_samples')

links = ['average', 'single', 'complete']
selected_links = tk.StringVar()
combobox_links = ttk.Combobox(frame, textvariable=selected_links)
combobox_links['values'] = links
combobox_links['state'] = 'readonly'
combobox_links.set('average')
# s = ttk.Style()
# s.theme_use('clam')
p_y = 0
label_error_numeric = ttk.Label(frame, text='')
label_save_message = ttk.Label(frame, text='')
# label_err1.pack()
# label_err2.pack()
label_err3 = ttk.Label(frame, text='')
label_config_par = ttk.Label(frame, text='Налашування парметрів алгоритму')
label_link = ttk.Label(frame, text="Оберіть метод зв'язування")
label_param = ttk.Label(frame, text="Оберіть параметр для дослідження")
label_range_format = ttk.Label(
    frame, text='Формат вводу: Число1 пробіл Число2')
label_claster_count = ttk.Label(frame, text='К-ть кластерів ', font='20')


label_enter_company = ttk.Label(
    frame, text='Оберіть компанію компанію ', font='20')
label_enter_algorithm = ttk.Label(frame, text='Обреріть алгоритм ', font='20')

label_count_p = ttk.Label(frame, text='Кількість рівнів дерева ', font='20')


label3 = ttk.Label(frame, text='Рік ', font='20')
label4 = ttk.Label(frame, text='Місяць ', font='20')
label_range_search = ttk.Label(frame, text='Межі дослідження', font='20')
# label1.place(x=10,y=50)

entry_range_search = ttk.Entry(frame, font='30')
entry_enter_range_2 = ttk.Entry(frame, font='20')
entry3 = ttk.Entry(frame, font='30')
entry4 = ttk.Entry(frame, font='30')

entry_cluster_count = ttk.Entry(frame, font='30')


g_value = 0.2
g = tk.DoubleVar(value=g_value)
label_scale = ttk.Label(text=g_value)
# textvariable=g


def change():
    g.set(scale.get())
    # print(round(g.get(), 2))
    label_scale["text"] = str(round(g.get(), 2))
```

```
# lambda event: g.set(scale.get())
scale = ttk.Scale(orient=tk.HORIZONTAL, from_=0, to=1,
                  length=250, variable=g, command=lambda x: change())

# image1 = Image.open("bardejov.jpg")
# image1 = image1.resize((50, 50), Image.ANTIALIAS)
# test = ImageTk.PhotoImage(image1)
label_test = ttk.Label(frame)
# label_test.pack()

def get_mode(df: pd.DataFrame):
    result = pd.DataFrame([], columns=df.columns)
    if('cluster' not in df.columns):
        return result
    counts = []
    for cluster, df_cl in df.groupby('cluster'):
        # result = pd.concat([result, df_cl.mode(axis=0, dropna=False).loc[0]],axis=1, ignore_index=True,
  sort=False)
        result.loc[cluster] = df_cl.mode(axis=0, dropna=False).loc[0]
        # print(result)
        # result = result.append(df_cl.mode(axis=0, dropna=False).loc[0])
        counts.append(df_cl.shape[0])
    result['count_rows'] = counts
    return result

def company_changed(event):
    """ handle the month changed event """
    showinfo(
        title='Результат',
        message=f'Ви обрали {selected_company.get()}!'
    )


def eneter_alg_company():
    global df_result
    df_result = pd.DataFrame([], columns=['cluster'])
    algorithm = combobox_algorithms.get()
    company = combobox_company.get()
    label_test.place_forget()
    entry_range_search.delete(0, tk.END)
    entry_cluster_count.delete(0, tk.END)
    entry_enter_range_2.delete(0, tk.END)
    # entry_range_search.delete(0, tk.END)
    label_enter_company.place_forget()
    combobox_company.place_forget()

    label_enter_algorithm.place_forget()
    combobox_algorithms.place_forget()

    button_enter_company.place_forget()
    if (algorithm == algorithms[0]):
        label_config_par.place(x=450, y=10)

        label_range_search.config(text='Межі дослідження', font='20')
        label_range_search.place(x=10, y=10)

        entry_range_search.place(x=10, y=30, relwidth=0.08)

        label_range_format.config(text='Формат вводу: \nЧисло1 пробіл Число2')
        label_range_format.place(x=100, y=30)
```

```
    label_claster_count.config(text='К-ть кластерів ', font='20')
    label_claster_count.place(x=10, y=110)

    button_get_image_k_means.place(x=10, y=70)
    button_calc_centr.place(x=10, y=170)
    entry_cluster_count.place(x=10, y=130, relwidth=0.08)
    # label_config_par.place(anchor='center')
    button_save.place(x=10, y=235)
    button_back.place(x=10, y=280)

  elif (algorithm == algorithms[1]):
    label_config_par.place(x=450, y=10)

    label_range_search.config(text='Межі дослідження', font='20')
    label_range_search.place(x=10, y=10)

    entry_range_search.place(x=10, y=30, relwidth=0.08)

    label_range_format.config(text='Формат вводу: Число1 пробіл Число2')
    label_range_format.place(x=100, y=30)

    label_link.place(x=10, y=70)

    combobox_links.place(x=10, y=100)

    label_claster_count.config(text='К-ть кластерів ', font='20')
    label_claster_count.place(x=10, y=150)
    entry_cluster_count.place(x=10, y=170, relwidth=0.08)

    # label_range_search.config(text="Метод зв'язування ",font='20')
    # label_range_search.place(x=10,y=10)
    # button_get_image_k_means.place(x=10,y=70)
    button_calc_tree.place(x=10, y=215)
    button_plot_cost.place(x=140, y=215)

    label_count_p.place(x=10, y=270)
    combobox_p.place(x=10, y=290)

    button_make_dendro.place(x=10, y=330)

    button_save.place(x=10, y=390)

    button_back.place(x=10, y=435)
  elif (algorithm == algorithms[2]):
    label_param.place(x=10, y=10)
    combobox_param_dbscan.place(x=10, y=30)

    label_range_search.config(text='Межі дослідження', font='20')
    label_range_search.place(x=10, y=70)
    entry_range_search.place(x=10, y=90, relwidth=0.2)

    label_claster_count.config(
      text='Оберіть значення ε(Епсілон) ', font='20')
    label_claster_count.place(x=10, y=130)
    label_scale.place(x=10, y=150)
    scale.place(x=10, y=190)
    # label_range_format.place(x=200,y=10)

    label_range_format.config(
      text="Мінімальна кількість спостережень \nдля створення кластера ", font='20')
    label_range_format.place(x=10, y=220)
```

```
        entry_cluster_count.place(x=10, y=270, relwidth=0.2)
        button_calc_dbscan.place(x=10, y=320)
        button_plot_cost_dbscan.place(x=140, y=320)
        # combobox_links.place(x=10,y=30)
        # button_get_image_k_means.place(x=10,y=70)
        # button_calc_tree.place(x=10,y=130)
        button_save.place(x=10, y=390)

        button_back.place(x=10, y=435)
        label_config_par.place(x=450, y=10)
        # button_make_dendro.place(x=10, y=270)
        # label_count_p.place(x=10, y=190)
        # combobox_p.place(x=10, y=220)
    elif (algorithm == algorithms[3]):
        label_config_par.place(x=450, y=10)

        label_param.place(x=10, y=10)
        combobox_param_hdbscan.place(x=10, y=30)

        label_range_format.config(text='Межі дослідження', font='20')
        label_range_format.place(x=10, y=70)
        entry_enter_range_2.place(x=10, y=90, relwidth=0.2)

        label_range_search.config(
            text="Мінімальний розмір кластера", font='20')
        label_range_search.place(x=10, y=130)
        entry_cluster_count.place(x=10, y=160, relwidth=0.08)

        label_claster_count.config(
            text='Мінімальна кількість спостережень \nдля створення кластера ', font='20')
        label_claster_count.place(x=10, y=200)
        entry_range_search.place(x=10, y=240, relwidth=0.08)

        button_calc_hdbscan.place(x=10, y=290)
        button_plot_cost_hdbscan.place(x=140, y=290)

        button_save.place(x=10, y=350)

        button_back.place(x=10, y=395)

    else:
        to_back([])
        print('a')
    # print(algorithm, company)


def plot_K():
    global label_test, label_error_numeric, label_save_message, entry_range_search, label_err3, frame,
    label_range_format
    # label_test
    label_test.place_forget()
    str1 = entry_range_search.get().split(' ')
    company = combobox_company.get()
    K = range(10, 11)
    # print(str1)
    # print(str1[0].isnumeric(), str1[1].isnumeric(), )
    if (len(str1) > 1 and str1[0].isnumeric() and str1[1].isnumeric() and int(str1[0]) < int(str1[1])):
        K = range(int(str1[0]), int(str1[1])+1)
        # print(K)
    else:
        label_range_format.place_forget()
```

```
            label_err3.config(
                text='Не привильний формат вводу\nВикористовуємо межі 10-25')
            label_err3.place(x=200, y=10)
            frame.update()

        # print(K)
        cost, _, _ = get_optK(K, company=company)
        fig = plt.figure()
        plt.plot(K, cost, 'bx-')
        print(K, cost)
        plt.xlabel('No. of clusters')
        plt.ylabel('Cost')
        plt.title('Elbow Method For Optimal k')
        fig.savefig('saved_figure.png')
        image1 = Image.open("saved_figure.png")
        # image1 = image1.resize((50, 50), Image.ANTIALIAS)
        temp = ImageTk.PhotoImage(image1)
        label_test2 = ttk.Label(frame, image=temp)
        label_test2.image = temp
        label_test = label_test2
        label_error_numeric.place_forget()
        label_save_message.place_forget()
        label_err3.place_forget()
        label_range_format.place_forget()
        label_test2.place(x=250, y=10)
        # plt.show()


    def calc_centroid():
        global entry_cluster_count, label_test, label_err3, label_test2, label_range_format, df_result
        str_centr = entry_cluster_count.get()
        # str_year = entry3.get()
        # str_month = entry4.get()
        label_test.place_forget()
        company = combobox_company.get()
        # year, month = '2016', '6'
        # label_test2.place_forget()
        if (str_centr.isnumeric()):
            # if(str_year.isnumeric() and int(str_year)>=2012 and int(str_year)<=2016):
            #     year = str_year
            # if(str_month.isnumeric() and int(str_month)<=12 and int(str_year)>=0):
            #     month = str_month
            label_error_numeric.config(text='', font='30')
            K = range(int(str_centr), int(str_centr)+1)
            cost, centroids, df_result = get_optK(K, company=company)
            image1 = Image.open("saved_figure.png")
            # image1 = image1.resize((50, 50), Image.ANTIALIAS)
            temp = ImageTk.PhotoImage(image1)
            label_test2 = ttk.Label(frame, image=temp)
            label_test2.image = temp
            label_test = label_test2
            # label_error_numeric.place_forget()
            # label_save_message.place_forget()
            label_err3.place_forget()
            label_range_format.place_forget()
            label_test2.place(x=350, y=50)
            used_var = ['Product', 'Sub-product', 'Issue',
                        'Submitted via', 'Date sent to company', 'Company response to consumer',
                        'year']
            print(centroids.shape, centroids)
            centroids = pd.DataFrame(centroids)
```

```
    #     columns=['Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via',
    #    'Date sent to company', 'Company response to consumer']
        print(centroids.head())
        # centroids['year'] = [year for i in range(len(centroids))]
        # centroids['month'] = [month for i in range(len(centroids))]
        # centroids.to_csv('Result_centroid_K_means_for_' + company + '.csv')
        # df_result.to_csv('Clustering_label_K_means_for_' + company + '.csv')
        label_error_numeric.place_forget()
        label_err3.place_forget()
        # label_save_message.config(
        #     text=' Файл з результатами обрахунку збережено', font='30')
        # label_save_message.place(x=450, y=550)
        # label_save_message.place(x=200,y=170)
        label_range_format.place(x=200, y=10)

    else:
        label_err3.place_forget()
        label_save_message.place_forget()
        label_range_format.place(x=200, y=10)
        label_error_numeric.config(text=' Введіть число не символ', font='30')
        label_error_numeric.place(x=200, y=170)


def calc_tree():
    global entry_cluster_count, label_test, label_err3, label_test2, label_range_format, df_result
    str_centr = entry_cluster_count.get()
    # str_year = entry3.get()
    # str_month = entry4.get()
    label_test.place_forget()
    company = combobox_company.get()
    str_link = combobox_links.get()

    # year, month = '2016', '6'
    # label_test2.place_forget()
    if (str_centr.isnumeric()):
        # if(str_year.isnumeric() and int(str_year)>=2012 and int(str_year)<=2016):
        #     year = str_year
        # if(str_month.isnumeric() and int(str_month)<=12 and int(str_year)>=0):
        #     month = str_month
        label_error_numeric.config(text='', font='30')
        K = int(str_centr)
        df_result = get_agg(K, company=company, link=str_link)
        image1 = Image.open("saved_figure.png")
        # image1 = image1.resize((50, 50), Image.ANTIALIAS)
        temp = ImageTk.PhotoImage(image1)
        label_test2 = ttk.Label(frame, image=temp)
        label_test2.image = temp
        label_test = label_test2
        # label_error_numeric.place_forget()
        # label_save_message.place_forget()
        label_err3.place_forget()
        label_range_format.place_forget()
        label_test2.place(x=350, y=50)
        # used_var = ['Product', 'Sub-product', 'Issue',
        #             'Submitted via', 'Date sent to company', 'Company response to consumer',
        #             'year']
        # print(centroids.shape, centroids)
        # centroids = pd.DataFrame(centroids, columns=used_var)
    #     columns=['Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via',
    #    'Date sent to company', 'Company response to consumer']
        # print(centroids.head())
```

```
        # centroids['year'] = [year for i in range(len(centroids))]
        # centroids['month'] = [month for i in range(len(centroids))]
        # centroids.to_csv('Result_centroid_K_means_for_' +company +'.csv')
        # df.to_csv('Clustering_label_AG_for_' + company + '.csv')
        label_error_numeric.place_forget()
        label_err3.place_forget()
        # label_save_message.config(
        #     text=' Файл з результатами обрахунку збережено', font='30')
        # label_save_message.place(x=450, y=170)
        # label_range_format.place(x=200,y=10)

    else:
        label_err3.place_forget()
        label_save_message.place_forget()
        label_range_format.place(x=200, y=10)
        label_error_numeric.config(text=' Введіть число не символ', font='30')
        label_error_numeric.place(x=200, y=170)


def make_dendro():
    global entry_cluster_count, label_test, label_err3, label_test2, label_range_format, entry4, entry3
    str_centr = entry_cluster_count.get()
    # str_year = entry3.get()
    # str_month = entry4.get()
    label_test.place_forget()

    label_test.place_forget()
    company = combobox_company.get()
    str_link = combobox_links.get()
    p = combobox_p.get()
    # year, month = '2016', '6'
    # label_test2.place_forget()
    if (str_centr.isnumeric()):
        # if(str_year.isnumeric() and int(str_year)>=2012 and int(str_year)<=2016):
        #     year = str_year
        # if(str_month.isnumeric() and int(str_month)<=12 and int(str_year)>=0):
        #     month = str_month
        label_error_numeric.config(text='', font='30')
        K = int(str_centr)
        # cost, centroids, df = make_dendrogram(K, company = company, link=str_link)
        make_dendrogram(K, company=company, link=str_link, p=int(p))
        image1 = Image.open("saved_figure.png")
        # image1 = image1.resize((50, 50), Image.ANTIALIAS)
        temp = ImageTk.PhotoImage(image1)
        label_test2 = ttk.Label(frame, image=temp)
        label_test2.image = temp
        label_test = label_test2
        # label_error_numeric.place_forget()
        # label_save_message.place_forget()
        label_err3.place_forget()
        label_range_format.place_forget()
        label_test2.place(x=350, y=50)
    #     used_var = ['Product', 'Sub-product', 'Issue',
    #     'Submitted via', 'Date sent to company', 'Company response to consumer',
    #     'year']
    #     # print(centroids.shape, centroids)
    #     centroids = pd.DataFrame(centroids, columns=used_var)
    #     plt.title("Hierarchical Clustering Dendrogram")
    # plot the top three levels of the dendrogram
        # plot_dendrogram(model_single, truncate_mode="level", p=2)
        # plt.xlabel("Number of points in node (or index of point if no parenthesis).")
```

```
        # plt.rcParams["figure.figsize"] = (20,10)
        # plt.show()
    #     columns=['Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via',
    #    'Date sent to company', 'Company response to consumer']
        # print(centroids.head())
        # centroids['year'] = [year for i in range(len(centroids))]
        # centroids['month'] = [month for i in range(len(centroids))]
        # centroids.to_csv('Result_centroid_K_means_for_' +company +'.csv')
        # df.to_csv('Clustering_label_K_means_for_' +company +'.csv')
        label_error_numeric.place_forget()
        label_err3.place_forget()
        # label_save_message.config(text=' Файл з результатами обрахунку збережено',font='30')
        # label_save_message.place(x=200,y=170)
        # label_range_format.place(x=200,y=10)

    else:
        label_err3.place_forget()
        label_save_message.place_forget()
        label_range_format.place(x=200, y=10)
        label_error_numeric.config(text=' Введіть число не символ', font='30')
        label_error_numeric.place(x=200, y=170)


def plot_tree_cost():
    global label_test, label_error_numeric, label_save_message, entry_range_search, label_err3, frame,
   label_range_format
    # label_test
    label_test.place_forget()

    str1 = entry_range_search.get().split(' ')
    company = combobox_company.get()
    link = combobox_links.get()
    K = range(10, 11)
    # print(str1)
    # print(str1[0].isnumeric(), str1[1].isnumeric(), )
    if (len(str1) > 1 and str1[0].isnumeric() and str1[1].isnumeric() and int(str1[0]) < int(str1[1])):
        K = range(int(str1[0]), int(str1[1])+1)
        # print(K)
    else:
        label_range_format.place_forget()
        label_err3.config(
            text='Не привильний формат вводу\nВикористовуємо межі 10-25')
        label_err3.place(x=200, y=10)
        frame.update()

    # print(K)
    cost = agg_plot(K, company=company, link=link)
    fig = plt.figure()
    plt.plot(K, cost, 'bx-')
    # print(K, cost)
    plt.xlabel('No. of clusters')
    plt.ylabel('Silhouette score')
    plt.title('Elbow Method For Optimal k')
    fig.savefig('saved_figure.png')
    image1 = Image.open("saved_figure.png")
    # image1 = image1.resize((50, 50), Image.ANTIALIAS)
    temp = ImageTk.PhotoImage(image1)
    label_test2 = ttk.Label(frame, image=temp)
    label_test2.image = temp
    label_test = label_test2
    label_error_numeric.place_forget()
```

```
    label_save_message.place_forget()
    label_err3.place_forget()
    label_range_format.place_forget()
    label_test2.place(x=300, y=10)
    # plt.show()


def calc_dbscan():
    global entry_cluster_count, label_test, label_err3, label_test2, label_range_format, df_result
    str_minPts = entry_cluster_count.get()
    # str_year = entry3.get()
    # str_month = entry4.get()
    eps = round(scale.get(), 2)
    label_test.place_forget()
    company = combobox_company.get()
    str_link = combobox_links.get()

    # year, month = '2016', '6'
    # label_test2.place_forget()
    if (str_minPts.isnumeric()):
        # if(str_year.isnumeric() and int(str_year)>=2012 and int(str_year)<=2016):
        #     year = str_year
        # if(str_month.isnumeric() and int(str_month)<=12 and int(str_year)>=0):
        #     month = str_month
        label_error_numeric.config(text='', font='30')
        K = int(str_minPts)
        df_result = get_dbscan(company=company, e=eps, min_sample=K)
        image1 = Image.open("saved_figure.png")
        # image1 = image1.resize((50, 50), Image.ANTIALIAS)
        temp = ImageTk.PhotoImage(image1)
        label_test2 = ttk.Label(frame, image=temp)
        label_test2.image = temp
        label_test = label_test2
        # label_error_numeric.place_forget()
        # label_save_message.place_forget()
        label_err3.place_forget()
        label_range_format.place_forget()
        label_test2.place(x=350, y=50)
        # used_var = ['Product', 'Sub-product', 'Issue',
        #             'Submitted via', 'Date sent to company', 'Company response to consumer',
        #             'year']
        # print(centroids.shape, centroids)
        # centroids = pd.DataFrame(centroids, columns=used_var)
    #     columns=['Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via',
    #    'Date sent to company', 'Company response to consumer']
        # print(centroids.head())
        # centroids['year'] = [year for i in range(len(centroids))]
        # centroids['month'] = [month for i in range(len(centroids))]
        # centroids.to_csv('Result_centroid_K_means_for_' +company +'.csv')
        # df.to_csv('Clustering_label_DBSCAN_for_' + company + '.csv')
        label_error_numeric.place_forget()
        label_err3.place_forget()
        # label_save_message.config(
            # text=' Файл з результатами обрахунку збережено', font='30')
        # label_save_message.place(x=450, y=550)
        # label_range_format.place(x=200,y=10)

    else:
        label_err3.place_forget()
        label_save_message.place_forget()
        # label_range_format.place(x=200,y=10)
```

```
        label_error_numeric.config(text=' Введіть число не символ', font='30')
        label_error_numeric.place(x=200, y=150)


def plot_dbscan_cost():
    global label_test, label_error_numeric, label_save_message, entry_range_search, label_err3, frame,
    label_range_format
    # label_test
    label_test.place_forget()

    param = combobox_param_dbscan.get()
    str1 = entry_range_search.get().split(' ')
    company = combobox_company.get()
    label_err3.place_forget()
    K = list(range(10, 11))
    # print(str1)
    # print(str1[0].isnumeric(), str1[1].isnumeric(), )
    cost = []
    number_of_cluster = []
    # print(str1[0], str1[1])
    if (param == 'eps'):
        try:
            diff = float(str1[1]) - float(str1[0])
            # print(diff)
            K = [float(str1[0]) + i*(diff/5) for i in range(5)]
            # print(K)
            cost, number_of_cluster = dbscan_plot(
                K, company=company, param=param)
            # print(float(str1[0]), float(str1[1]))
        except:
            print("no")
            label_err3.config(
                text='Не привильний формат вводу\nВикористовуємо межі 10-25')
            label_err3.place(x=250, y=90)
            frame.update()
            return 0
    elif (param == 'min_samples'):

        if (len(str1) > 1 and str1[0].isnumeric() and str1[1].isnumeric() and int(str1[0]) < int(str1[1])):

            K = list(range(int(str1[0]), int(str1[1])+1))
            cost, number_of_cluster = dbscan_plot(
                K, company=company, param=param)

        # print(K)
        else:
            # label_range_format.place_forget()
            label_err3.config(
                text='Не привильний формат вводу\nВикористовуємо межі 10-25')
            label_err3.place(x=250, y=90)
            frame.update()
            return 0
    # return 0
    # print(K)
    # cost = agg_plot(K, company=company, link=param)
    print(K, number_of_cluster, cost)
    # plt.plot(K, cost)
    # plt.show()
    # f = plt.figure()
    fig, ax = plt.subplots(3)
    ax[0].plot(K, cost)
```

```
    ax[1].scatter(number_of_cluster, cost)
    ax[2].scatter(K, number_of_cluster)
    # # plt.scatter(K, cost, 'bx-')
    # # print(K, cost)

    ax[0].set_xlabel(param)
    ax[0].set_ylabel('Silhouette score')
    ax[0].set_title('Elbow Method For Optimal '+param)

    ax[1].set_xlabel('No. of clusters')
    ax[1].set_ylabel('Silhouette score')
    ax[1].set_title('No. of clusters to Cost')

    ax[2].set_xlabel(param)
    ax[2].set_ylabel('No. of clusters')
    ax[2].set_title(param + ' to No. of clusters')
    fig.tight_layout()
    # fig.tight_layout(pad=5.0)
    # plt.show()
    # plt.xlabel('No. of clusters')
    # plt.ylabel('Cost')
    # plt.title('Elbow Method For Optimal k')
    fig.savefig('saved_figure.png')
    image1 = Image.open("saved_figure.png")
    # image1 = image1.resize((50, 50), Image.ANTIALIAS)
    temp = ImageTk.PhotoImage(image1)
    label_test2=ttk.Label(frame, image=temp)
    label_test2.image=temp
    label_test = label_test2
    label_error_numeric.place_forget()
    label_save_message.place_forget()
    label_err3.place_forget()
    # label_range_format.place_forget()
    label_test2.place(x=350,y=90)
    # plt.show()


def calc_hdbscan():
    global entry_cluster_count, label_test, label_err3, label_test2, label_range_format, df_result
    str_minPts = entry_cluster_count.get()
    min_sample = entry_range_search.get()
    # str_year = entry3.get()
    # label_range_format
    # str_month = entry4.get()
    # eps = round(scale.get(), 2)
    label_test.place_forget()
    company = combobox_company.get()
    # str_link = combobox_links.get()

    # year, month = '2016', '6'
    # label_test2.place_forget()
    if (str_minPts.isnumeric() and min_sample.isnumeric()):
        # if(str_year.isnumeric() and int(str_year)>=2012 and int(str_year)<=2016):
        #     year = str_year
        # if(str_month.isnumeric() and int(str_month)<=12 and int(str_year)>=0):
        #     month = str_month
        label_error_numeric.config(text='', font='30')
        K = int(str_minPts)
        min_sample = int(min_sample)
        df_result = get_hdbscan(company=company,  min_cluster_size=K,
                    min_sample=min_sample)
```

```
        image1 = Image.open("saved_figure.png")
        # image1 = image1.resize((50, 50), Image.ANTIALIAS)
        temp = ImageTk.PhotoImage(image1)
        label_test2 = ttk.Label(frame, image=temp)
        label_test2.image = temp
        label_test = label_test2
        # label_error_numeric.place_forget()
        # label_save_message.place_forget()
        label_err3.place_forget()
        # label_range_format.place_forget()
        label_test2.place(x=350, y=50)
        used_var = ['Product', 'Sub-product', 'Issue',
                    'Submitted via', 'Date sent to company', 'Company response to consumer',
                    'year']
        # print(centroids.shape, centroids)
        # centroids = pd.DataFrame(centroids, columns=used_var)
    #     columns=['Date received', 'Product', 'Sub-product', 'Issue', 'Submitted via',
    #    'Date sent to company', 'Company response to consumer']
        # print(centroids.head())
        # centroids['year'] = [year for i in range(len(centroids))]
        # centroids['month'] = [month for i in range(len(centroids))]
        # centroids.to_csv('Result_centroid_K_means_for_' +company +'.csv')
        # df.to_csv('Clustering_label_DBSCAN_for_' + company + '.csv')
        label_error_numeric.place_forget()
        label_err3.place_forget()
        # label_save_message.config(
        #     text=' Файл з результатами обрахунку збережено', font='30')
        # label_save_message.place(x=450, y=550)
        # label_range_format.place(x=200,y=10)

    else:
        label_err3.place_forget()
        label_save_message.place_forget()
        # label_range_format.place(x=200,y=10)
        label_error_numeric.config(text=' Введіть число не символ', font='30')
        label_error_numeric.place(x=200, y=150)

def plot_hdbscan_cost():
    global label_test, label_error_numeric, label_save_message, entry_range_search, label_err3, frame,
    label_range_format
    # label_test
    label_test.place_forget()

    param = combobox_param_hdbscan.get()
    str1 = entry_enter_range_2.get().split(' ')
    company = combobox_company.get()
    label_err3.place_forget()
    K = list(range(10, 11))
    # print(str1)
    # print(str1[0].isnumeric(), str1[1].isnumeric(), )
    cost = []
    number_of_cluster = []
    # print(str1[0], str1[1])
    if (len(str1) > 1 and str1[0].isnumeric() and str1[1].isnumeric() and int(str1[0]) < int(str1[1])):

        K = list(range(int(str1[0]), int(str1[1])+1))
        cost, number_of_cluster = hdbscan_plot(
            K, company=company, param=param)

    # print(K)
    else:
```

```
    # label_range_format.place_forget()
    label_err3.config(
        text='Не привильний формат вводу\nВикористовуємо межі 10-25')
    label_err3.place(x=250, y=90)
    frame.update()
    return 0
# elif (param == 'min_samples'):

#     if (len(str1) > 1 and str1[0].isnumeric() and str1[1].isnumeric() and int(str1[0]) < int(str1[1])):

#         K = list(range(int(str1[0]), int(str1[1])+1))
#         cost, number_of_cluster = hdbscan_plot(
#             K, company=company, param=param)

#     # print(K)
#     else:
#         # label_range_format.place_forget()
#         label_err3.config(
#             text='Не привильний формат вводу\nВикористовуємо межі 10-25')
#         label_err3.place(x=250, y=90)
#         frame.update()
#         return 0
# return 0
# print(K)
# cost = agg_plot(K, company=company, link=param)
print(K, number_of_cluster, cost)
# plt.plot(K, cost)
# plt.show()
# f = plt.figure()
fig, ax = plt.subplots(3)
ax[0].plot(K, cost)
ax[1].scatter(number_of_cluster, cost)
ax[2].scatter(K, number_of_cluster)
# # plt.scatter(K, cost, 'bx-')
# # print(K, cost)

ax[0].set_xlabel(param)
ax[0].set_ylabel('Silhouette score')
ax[0].set_title('Elbow Method For Optimal '+param)

ax[1].set_xlabel('No. of clusters')
ax[1].set_ylabel('Silhouette score')
ax[1].set_title('No. of clusters to Cost')

ax[2].set_xlabel(param)
ax[2].set_ylabel('No. of clusters')
ax[2].set_title(param + ' to No. of clusters')
fig.tight_layout()
# fig.tight_layout(pad=5.0)
# plt.show()
# plt.xlabel('No. of clusters')
# plt.ylabel('Cost')
# plt.title('Elbow Method For Optimal k')
fig.savefig('saved_figure.png')
image1 = Image.open("saved_figure.png")
# image1 = image1.resize((50, 50), Image.ANTIALIAS)
temp = ImageTk.PhotoImage(image1)
label_test2=ttk.Label(frame, image=temp)
label_test2.image=temp
label_test = label_test2
label_error_numeric.place_forget()
```

```
    label_save_message.place_forget()
    label_err3.place_forget()
    # label_range_format.place_forget()
    label_test2.place(x=350,y=90)
    # plt.show()


def to_back(entytys_alg):
    for i in entytys_alg:
        i.place_forget()

    # del df_result

    label_enter_company.place(x=10, y=10)
    combobox_company.place(x=10, y=30)

    label_enter_algorithm.place(x=10, y=70)
    combobox_algorithms.place(x=10, y=90)

    button_enter_company.place(x=10, y=130)
    button_back.place_forget()
    label_test.place_forget()

def save_results(df: pd.DataFrame, company:str, method:str):
    # print(df.cluster.unique())
    clusters = df.cluster.unique()
    df_centroids = get_mode(df)
    time = str(datetime.datetime.now()).replace(":", "-")
    path_to_folder = os.path.dirname(os.path.abspath(__file__)) + '/'
    print(path_to_folder)
    path = 'Results'
    path_pred = path+'/Predict_files/'
    paht_centr = path + '/Centroids/'
    path_method = path+ '/Clusters_'+method+'_for_'+ company + '_at_'+time
    print(os.path.exists(path_to_folder+path))
    if(not os.path.exists(path_to_folder+path)):
        os.mkdir(path)

    print(path_to_folder+path_pred, os.path.exists(path_to_folder+path_pred))
    if(not os.path.exists(path_to_folder+path_pred)):
        os.mkdir(path_to_folder+path+'/Predict_files')

    df.to_csv(path_to_folder+path+'/Predict_files/Clustering_label_'+method+'_for_' + company + '.csv')

    if(not os.path.exists(path_to_folder+path_method+'/')):
        os.mkdir(path_to_folder+path_method)

    for i in clusters:
        df.loc[df.cluster==i].to_csv(path_method+ '/Cluster_№'+ str(i)+'_method='+method+'_company='+
    company+'_time='+time+'.csv')

    if(not os.path.exists(path_to_folder+paht_centr)):
        os.mkdir(path_to_folder + paht_centr[:-1])

    df_centroids.to_csv(path_to_folder + paht_centr + 'Centrids_method='+method+'_company='+
    company+'_time='+time+\
        '.csv', index=False)



    label_save_message.config(
```

```
        text=' Файли з результатами обрахунків збережено', font='30')
    label_save_message.place(x=450, y=550)
    # df.to_csv()
    # print(df)
    return 0

# combobox_company.bind('<<ComboboxSelected>>', company_changed)

# label2=ttk.Label(frame,text='К-ть ранжувань ',font='20')
# label2.place(x=10,y=70)

# entry2=ttk.Entry(frame,font='30')
# entry2.place(x=10,y=100,relwidth=0.05)




# button1.place(x=10,y=140)

button_get_image_k_means = ttk.Button(
    frame, text='Малюнок', command=plot_K, style="Accent.TButton")


button_calc_centr = ttk.Button(
    frame, text='Розрахувати\n центроїди', command=calc_centroid, style="Accent.TButton")

button_calc_tree = ttk.Button(
    frame, text='Розрахувати\n дерево', command=calc_tree, style="Accent.TButton")
button_plot_cost = ttk.Button(
    frame, text='Розрахувати\n вартість', command=plot_tree_cost, style="Accent.TButton")
button_make_dendro = ttk.Button(
    frame, text='Побудувати\n дендрограму', command=make_dendro, style="Accent.TButton")

button_calc_dbscan = ttk.Button(
    frame, text='Розрахувати\n кластери', command=calc_dbscan, style="Accent.TButton")
button_plot_cost_dbscan = ttk.Button(
    frame, text='Розрахувати\n вартість', command=plot_dbscan_cost, style="Accent.TButton")


button_calc_hdbscan = ttk.Button(
    frame, text='Розрахувати\n кластери', command=calc_hdbscan, style="Accent.TButton")
button_plot_cost_hdbscan = ttk.Button(
    frame, text='Розрахувати\n вартість', command=plot_hdbscan_cost, style="Accent.TButton")

button_enter_company = ttk.Button(
    frame, text='Обрати', command=eneter_alg_company, style="Accent.TButton")

button_save = ttk.Button(
    frame, text='Зеберегти результати', command=lambda: save_results(df_result, combobox_company.get(), \
        combobox_algorithms.get()), style="Accent.TButton")

algs_entity = [label_range_search, entry_range_search, button_get_image_k_means, button_calc_centr,
               label_claster_count, label_range_format, label_error_numeric, entry_cluster_count,
               label_save_message, combobox_links, button_calc_tree, label_config_par, button_make_dendro,
               combobox_p, label_count_p, scale, label_scale, button_calc_dbscan, button_calc_hdbscan, label_link,
               button_plot_cost, combobox_param_dbscan, label_param, button_plot_cost_hdbscan,
    combobox_param_hdbscan, \
                entry_enter_range_2, button_save, button_plot_cost_dbscan, label_test]

button_back = ttk.Button(frame, text='Назад', command=lambda: to_back(
    algs_entity), style="Accent.TButton")
```

```
# generals_entity = [label_enter_company, combobox_company, label_enter_algorithm, combobox_algorithms,
  button_enter_company, button_back]
ttk.Button(root, text="Quit", command=root.quit, style="Accent.TButton").pack()

# label_range_search.place(x=10,y=10)
# label_enter_company.place(x=10,y=230)
label_enter_company.place(x=10, y=10)
combobox_company.place(x=10, y=30)

label_enter_algorithm.place(x=10, y=70)
combobox_algorithms.place(x=10, y=90)

button_enter_company.place(x=10, y=130)


# entry_range_search.place(x=10,y=30,relwidth=0.08)
# button_get_image_k_means.place(x=10,y=70)
# label_claster_count.place(x=10,y=110)
# label3.place(x=10,y=170)
# label4.place(x=70,y=170)
# entry_cluster_count.place(x=10,y=130,relwidth=0.08)
# entry3.place(x=10,y=190,relwidth=0.05)
# # entry_enter_company.place(x=10,y=250,relwidth=0.05)
# entry4.place(x=70,y=190,relwidth=0.05)
# button_calc_centr.place(x=10,y=300)
# label_range_format.place(x=200,y=10)

# combobox_company.place(x=10,y=260)
root.mainloop()

# root.quit()
```

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Brian S. Everitt, Sabine Landau, Morven Leese, Daniel StahlCluster Analysis, 5th Edition, Publisher(s): John Wiley & Sons, 2011 (346 pages).
2. Fluent Python, 2nd Edition, Released April 2022, by Luciano Ramalho, Publisher(s): O'Reilly Media, Inc (275 pages).
3. Pandas documentation .URL:https://pandas.pydata.org/docs/
4. Graphical User Interfaces with Tk. URL:https://docs.python.org/3/library/tk.html
5. K-modes algorithm. URL:https://docs.python.org/3/library/tk.html
6. Dataset for clustering analysys URL:https://data.world/data-society/consumer-complaint-data
7. Аналіз практик урегулювання скарг споживачів URL: http://www.fst-ua.info/ua/analiz-praktyk-urehuliuvannia-skarh-spozhyvachiv-strakhovykh-posluh/
8. Financial Statement Analysis: A Practitioner's Guide by Martin S. Fridson (Author), Fernando Alvarez (Author) , Publisher : Wiley; 4th edition (July 5, 2011) , 400 pages.
9. Financial Planning & Analysis and Performance Management, by Jack Alexander (Author) , Publisher : Wiley; 1st edition (June 13, 2018) , 640 pages .
10. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking by Foster Provost (Author), Tom Fawcett (Author) , Publisher : O'Reilly Media; 1st edition (September 17, 2013) , 413 pages .
11. Big Data: A Revolution That Will Transform How We Live, Work, and Think by Viktor Mayer-Schönberger (Author), Kenneth Cukier (Author) , Publisher : Harper Business; Reprint edition (March 4, 2014) , 272 pages.

12. Customer Complaints Analysis Using Text Mining and Outcome-Driven Innovation Method for Market-Oriented Product Development: https://www.mdpi.com/2071-1050/11/1/40

13. Data Analysis of Consumer Complaints in Banking Industry using Hybrid Clustering: https://www.researchgate.net/publication/328248172_Data_Analysis_of_Consumer_Complaints_in_Banking_Industry_using_Hybrid_Clustering

14. Clustering Bank Customer Complaints on Social Media for Analytical CRM via Multi-objective Particle Swarm Optimization:https://www.researchgate.net/publication/337568304_Clustering_Bank_Customer_Complaints_on_Social_Media_for_Analytical_CRM_via_Multi-objective_Particle_Swarm_Optimization