НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.8

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «Система аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту»

Виконав:
Студент 2 курсу, групи КА-22мп
Симонов Єгор Денисович _____

Науковий керівник:
Професор кафедри ММСА, д.ф.-м.н., проф.,
Макаренко Олександр Сергійович _____

Рецензент:
Професор кафедри ММСА, д.т.н., проф.,
Данилов Валерій Якович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент (*підпис*): _____

Київ - 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Симонову Єгору Денисовичу

1. **Тема дисертації:** «Система аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту», науковий керівник дисертації Макаренко Олександр Сергійович, професор, доктор фізико-математичних наук, затверджені наказом по університету від «08» листопада 2023 р. № 5200-с.
2. **Строк подання студентом дисертації**: _____
3. **Об’єкт дослідження:** моделі кластеризації текстових даних та нейронні мережі.
4. **Предмет дослідження:** взаємозв'язок методів кластеризації та процесу прийняття рішень у системах з використанням штучного інтелекту.
5. **Перелік завдань, які потрібно розробити:**
   1) огляд предметної області та аналіз наявних методів кластеризації, аналіз методів оцінки якості рішень в моделях з використанням штучного інтелекту;
   2) аналіз вхідних даних, попередня обробка та очищення даних, підготовка показників до передачі в моделі кластеризації та нейронну мережу;

3) розробка структури системи та моделей, необхідних для дослідження поставленого завдання;
4) тестування побудованої системи та аналіз отриманих результатів.

6. **Перелік графічного (ілюстративного) матеріалу:**
   1) схеми: архітектура системи та моделі машинного навчання;
   2) таблиці з результатами роботи моделей та описом даних;
   3) рисунки з прикладом роботи програми та описом моделей;
   4) графіки з результатами роботи програми;
   5) презентація.

7. **Орієнтовний перелік публікацій:** Симонов Є.Д., Макаренко О.С., Бідюк П.І. Система аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту, II науково-практична конференція «Системні науки та інформатика», КПІ ім. Ігоря Сікорського, Київ, 4-8 грудня, 2023. С. 207-212.

8. **Дата видачі завдання:** 01.09.2023

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Формулювання теми магістерської дисертації.	01.09.2023 – 08.09.2023	Виконано
2	Огляд літературно-інформаційних джерел за темою роботи.	09.09.2023 – 22.09.2023	Виконано
3	Виділення характеристик і моделей для дослідження. Вибір методів для дослідження.	23.09.2023 – 30.09.2023	Виконано
4	Проектування підходу до вирішення задачі та архітектури програмного продукту.	01.10.2023 – 15.10.2023	Виконано
5	Програмна реалізація створеної архітектури.	16.10.2023 – 29.10.2023	Виконано
6	Розробка інтерфейсу.	30.10.2023 – 06.11.2023	Виконано
7	Опис стартап-проєкту	07.11.2023 – 13.11.2023	Виконано
8	Оформлення пояснювальної записки	14.11.2023 – 31.12.2023	Виконано

Студент _____ Єгор СИМОНОВ

Науковий керівник дисертації _____ Олександр МАКАРЕНКО

РЕФЕРАТ

Магістерська дисертація: 98с., 22 рис., 21 табл., 1 додаток, 46 джерел.

МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, КЛАСТЕРИЗАЦІЯ, ТЕКСТОВИЙ АНАЛІЗ, СИСТЕМА

Об’єкт дослідження – моделі кластеризації текстових даних та нейронні мережі.

Предмет дослідження – взаємозв'язок методів кластеризації та процесу прийняття рішень у системах з використанням штучного інтелекту.

Мета магістерської дисертації – дослідити взаємозв’язок між методами класифікації та ефективністю процесу прийняття рішень у системах з використанням штучного інтелекту, оцінюючи рівень впливу кожного методу.

Актуальність роботи полягає у тому, що кількість систем, які використовують методи штучного інтелекту, стрімко зростає. Разом з цим зростає попит на обробку великих обсягів даних, зокрема, текстових, так як природна мова залишається найбільш зручним способом передачі та отримання інформації. Кластеризація даних є ключовим етапом у навчанні моделей такого типу, допомогаючи створювати ознаки для точних прогнозів.

Програмна реалізація виконана за допомогою мови програмування Python 3.10.2 та середовища розробки Visual Studio Code.

Дослідження показало вплив різних алгоритмів кластеризації на точність прогнозів моделей штучного інтелекту. Виявлено оптимальні параметри та моделі кластеризації, які сприяють підвищенню точності моделей штучного інтелекту, та проаналізовано вплив зміни кількості кластерів на якість отриманої інформації для прийняття рішень.

ABSTRACT

Master's thesis: 98p., 22 fig., 21 tables, 1 appendix, 46 sources.

MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, CLUSTERIZATION, TEXT ANALYSIS, SYSTEM

The object of research is text data clustering models and neural networks.

The subject of the study is the relationship between clustering methods and the decision-making process in systems using artificial intelligence.

The purpose of the master's thesis is to investigate the relationship between classification methods and the effectiveness of the decision-making process in systems using artificial intelligence, evaluating the level of influence of each method.

The relevance of the work lies in the fact that the number of systems that use artificial intelligence methods is growing rapidly. At the same time, the demand for processing large volumes of data, particularly text data, is growing, as natural language remains the most convenient way of transmitting and receiving information. Data clustering is a key step in training models of this type, helping to create features for accurate predictions.

The software implementation is made using the Python 3.10.2 programming language and the Visual Studio Code development environment.

The study showed the impact of different clustering algorithms on the accuracy of predictions of artificial intelligence models. The optimal parameters and clustering models, which contribute to increasing the accuracy of artificial intelligence models, have been determined, and the impact of changing the number of clusters on the quality of the received information for decision-making has been analyzed.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

NLP – Natural Language Processing

ІШІ – штучний інтелект

TF-IDF – Term Frequency-Inverse Document Frequency

TruncatedSVD – Truncated Singular Value Decomposition

GMM – Gaussian mixture model

BIC – Bayesian information criterion

AIC – Akaike information criterion

LDA – Latent Dirichlet Allocation

TP – True Positives

TN – True Negatives

FP – False Positives

FN – False Negatives

MAE – Mean Absolute Error

MSE – Mean Squared Error

ВСТУП

У сучасному світі системи з використанням алгоритмів штучного інтелекту стали невід'ємною складовою технологічного прогресу. Вони застосовуються у різних сферах, від автономних автомобілів до медичної діагностики, від особистих асистентів до фінансових структур. Ці системи охоплюють всі сфери життя, перетворюючи їх ізсередини і впливаючи на нашу повсякденність навіть у тих місцях, де це може здатися непростим або неможливим.

Ефективність та здатність до навчання таких систем напряму залежить від вбудованих методів обробки вхідної інформації. У даному контексті виникає необхідність вибору оптимальних методів обробки постійно зростаючих обсягів даних, при цьому кластеризація відіграє важливу роль у цьому процесі. Вона дозволяє структурувати дані та виявити внутрішні закономірності, які сприяють формуванню точних та надійних моделей штучного інтелекту.

У магістерській дисертації наведено дослідження взаємозв'язків між обраними методами кластеризації та якістю інформації, отриманої від моделей з використанням штучного інтелекту. Досліджується як наявність взаємозв'язку в цілому, так і більш детальні аспекти, зокрема оптимальні параметри та моделі кластеризації.

В якості вхідних даних для аналізу обрано набір на основі текстових запитів та категорійних параметрів про користувачів, які допоможуть комплекно проаналізувати поставлену задачу.

Взаємодія з системою відбувається за допомогою чат-боту Telegram, що дозволяє не лише отримувати результати роботи моделей, а й налаштовувати вхідні параметри.

Магістерська дисертація складається з вступу, чотирьох розділів, висновків, переліку джерел посилання та одного додатку. У розділі 1 досліджено актуальність проблеми кластеризації у моделях з використанням

штучного інтелекту, також наведено методи моделювання впливу кластеризації на якість рішень у цих моделях. У цьому розділі також розглянуто деякі сучасні системи, які використовують зазначені методи. У розділі 2 наведено детальний опис використовуваних методів кластеризації, формату текстових даних та архітектури нейронних мереж. У розділі 3 проведено практичний експеримент з побудовою повноцінної системи для дослідження впливу кластеризації на якість рішень в моделях з використанням штучного інтелекту, описано архітектуру побудованої системи та отримані результати. У розділі 4 запропоновано реалізацію стартап проекту з системою інтетектуальної обробки вхідних даних, яка використовуватиме методи кластеризації. Перелік джерел посилання складається з 46 джерел. Додаток А містить лістинг програмного коду.

РОЗДІЛ 1 АНАЛІЗ І ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз актуальності проблеми кластеризації у моделях з використанням штучного інтелекту

Термін «штучний інтелект» вперше було використано на конференції у Дартмутському коледжі (США) 1956 року, де було представлено модель Logic Theorist, авторами якої були Алан Ньюелл і Герберт Саймон. Ця модель могла вирішувати логічні задачі, які раніше вважались виключно людськими та нерозв'язними для машин. Саме ця конференція вважається початком галузі штучного інтелекту.

Протягом наступних років галузь продовжувала активно розвиватись. Так, у 1960-х роках були розроблені нові методи штучного інтелекту, такі як нейронні мережі та машинне навчання [1]. У 1970-х роках штучний інлелект почав впроваджуватись у розробки експертних систем реального світу [2]. У період 1980-2000 років спостерігалось зростання попиту на системи з використанням моделей штучного інтелекту, так як відбулась поява нових технологій на кшталт персонального комп’ютера та паралельного обчислення.

Починаючи з 2000-х років і по сьогодні, галузь машинного навчання активно розвивається, з'являються нові методи штучного інтелекту, такі як глибинне навчання та обробка природної мови. Разом з цим, пропорційно, зростає і відсоток систем, які використовують такі моделі. За результатами досліджень TechJury, у 2023 році 35% компаній використовують можливості моделей штучного інтелекту, а 42% компаній досліджують можливість його впровадження у свої системи [3].

Обробка природної мови для таких систем є однією з ключових напрямків взаємодії, так як це дає можливість отримати необхідну інформацію за допомогою зрозумілих методів. Дослідження Forrester Research у 2022 році

показує, що 25% підприємств розгорнули або активно планують розгортання діалогових рішень штучного інтелекту, які спираються на NL-взаємодію (взаємодію через природню мову) [4]. При цьому за прогнозами Gartner до 2025 року 70% взаємодій з клієнтами будуть пов’язані зі штучним інтелектом, що підкреслює зростаючу важливість таких систем в обслуговуванні клієнтів [5].

Тож, системи з використанням штучного інтелекту та обробкою природньої мови, дійсно важливі для бізнесу, науки та промисловості. Вони відтворюють розумові процеси людини для прийняття рішень через машинне навчання [6]. Однак у таких систем є важлива вимога, яка є блокуючим фактором до повноцінного їх впровадження у повсякдення. Такі системи потребують великих потужностей для обробки вхідних даних та навчання, що є неможливим для організацій малого масштабу.

За для зменшення витрат ресурсів, використовуваних під час навчання моделей такого типу, розробляються інтелектуальні алгоритми обробки та трансформації вхідних даних. Одним з ключових стратегічних підходів на цьому етапі є використання кластеризації. Цей процес дозволяє системі групувати дані на основі схожих характеристик, створюючи при цьому логічні категорії або кластери. Після групування дані стають більш структурованими та впорядкованими, що сприяє більш ефективному їх використанню для навчання моделей штучного інтелекту. Кластеризація дозволяє відокремити важливі закономірності в наборі даних, що полегшує їх аналіз та обробку в подальшому.

У форматі задачі кластеризації є ряд аспектів, які варто враховувати для побудови якісної моделі. По-перше, вона включає в себе вибір оптимального методу кластеризації, що відповідає характеру даних і меті аналізу. Далі, надзвичайно важливим є підбір оптимальних параметрів для конкретного завдання, зокрема, визначення оптимальної кількості кластерів. Правильний вибір кількості кластерів може суттєво вплинути на якість та коректність результатів кластеризації, допомагаючи знайти баланс між розділенням даних

на достатню кількість груп та уникненням перекласифікації або надмірності груп. Тому, визначення оптимальних параметрів в процесі кластеризації є ключовим аспектом для досягнення точних та зрозумілих результатів.

1.2 Огляд методів моделювання впливу кластеризації на якість рішень у моделях штучного інтелекту

Використання кластеризації в системах штучного інтелекту є предметом великого зацікавлення для науковців та фахівців у галузі машинного навчання. Цей підхід має потенціал значно підвищити якість інформації, отриманої для прийняття рішень. Однак, для повноцінної успішної інтеграції моделей такого типу в процес обробки вхідних даних, необхідне глибоке розуміння можливих наслідків. Тому вивчення впливу кластеризації на якість рішень в моделях з використанням штучного інтелекту, стає ключовим аспектом у дослідженні та розвитку таких систем. Для більш глибокого розуміння ефективності та обмежень цього підходу застосовуються різноманітні підходи: порівняння моделей з та без кластеризації, вивчення показників різних моделей, аналіз параметрів моделей тощо.

Перший тип оцінки полягає у застосуванні різних методів кластеризації для навчання однієї й тієї ж моделі штучного інтелекту, з подальшим порівнянням отриманих результатів. Цей підхід дозволяє зрозуміти, як саме вибір конкретного методу впливає на якість та ефективність кінцевої моделі. Окрім цього, навчання однієї моделі різними методами дає змогу виявити їхні сильні та слабкі сторони у контексті конкретного завдання.

Такий аналіз допомагає визначити оптимальний метод для конкретного типу даних та задачі, а також уникнути можливих недоліків, які можуть виникнути при застосуванні певного методу.

Другий метод оцінки полягає у застосуванні одного конкретного методу кластеризації з різною кількістю вихідних кластерів та подальше порівняння отриманих результатів. Це дозволяє проводити глибше дослідження впливу кількості сформованих груп на функціонування моделі штучного інтелекту та її здатність до класифікації та аналізу. Наведений підхід є ключовим у задачі оптиманого групування даних для подальшого використання, так як деталізація даних для різних задач може суттєво відрізнятись.

Такий аналіз дає можливість визначити оптимальну кількість кластерів для конкретного набору даних та завдання, що сприяє підвищенню точності та ефективності моделі штучного інтелекту. Крім того, це дозволяє виявити потенційні обмеження чи переваги використання різної кількості кластерів у випадку конкретних завдань з обробки даних та прийняття рішень.

Третій метод оцінки передбачає порівняння результатів, отриманих від моделей штучного інтелекту, навчених на некластеризованих даних, з моделями, навченими на кластеризованих даних. Такий підхід дозволяє отримати більш глибоке розуміння того, як саме групування даних за допомогою кластеризації впливає на показники кінцевої моделі.

Але варто враховувати, що застовування кластеризації для попередньої обробки даних перед навчанням може суттєво змінити їхню унікальну структуру, спричинити зміни у сприйнятті та обробці інформації, здатності моделі розпізнавати патерни та приймати правильні рішення. Тому такий підхід дає можливіть проаналізувати переваги та недоліки використання цієї стратегії обробки даних у контексті реальних завдань, допомагаючи розробникам штучного інтелекту у виборі найбільш оптимальних методів підготовки даних для досягнення високої ефективності моделей.

1.3 Деякі сучасні системи з використанням кластеризації як методу попередньої обробки інформації

У світі існує багато прикладів структур, які використовують кластеризацію як метод попередньої обробки вхідної інформації перед передачою в моделі з використанням методів штучного інтелекту. Такі системи зустрічаються у кардинально різних сферах. Зокрема, один з найпопулярніших варіантів застосування – це сегментацій клієнтів в електронній комерції.

Яскравим прикладом в контексті дослідження представляється структура, яку створила корпорація Amazon. У цьому випадку, система використовує методи кластеризації для автоматизованого групування клієнтів на підставі аналізу їхньої історії покупок та поведінки в інтернеті. Цей підхід сприяє індивідуалізації рекомендацій та цільової реклами, а також дає змогу системі прогнозувати перелік майбутніх покупок [7]. Такий метод дозволяє покращити взаємодію з користувачами, спрямовуючи їхні потреби та уподобання у відповідний спосіб, що є актуальним у контексті сучасних систем персоналізації та аналізу даних.

Поміж іншого, методи кластеризації виявляють свою корисність у системах автоматичного виявлення аномалій у безпеці мереж. Компанія Cisco Talos використовує ці підходи для виявлення нестандартних моделей мережевого трафіку, які можуть бути ознаками зловмисної активності. Групуючи схожі мережеві події, вони спрямовують свої ресурси на потенційно підозрілі випадки, що сприяє удосконаленню процесу виявлення та запобігання загрозам [8]. Ця стратегія виявлення аномалій у мережах дозволяє ідентифікувати несподівані патерни або відхилення від типової поведінки, що сприяє реакції на потенційні загрози у мережевому середовищі [9].

Також, в сфері медичної діагностики та терапії використання цих підходів набуває значущості, як у випадку клініки Майо. Вони застосовують

методи кластеризації для ретельного аналізу великої кількості медичних даних пацієнтів, таких як медичні записи та результати лабораторних досліджень. Це дозволяє їм виявляти групи пацієнтів з подібними медичними станами та факторами ризику, що сприяє більш глибокій та персоналізованій діагностиці, а також розробці індивідуальних лікувальних схем [10].

1.4 Висновки до розділу 1 та постановка задачі дослідження

Даний розділ дисертації був присвячений висвітленню актуальності застосування методів кластеризації у задачах побудов систем з використанням штучного інтелекту. Такий підхід дійно є значущим, так як дозволяє відокремити важливі закономірності у наборі даних, що на наступних кроках полегшує аналіз та обробку наявних даних.

Також варто зауважити, що хоч методи кластеризації і дають змогу отримати більш уніфіковані (груповані) дані, але вони потребують додаткових затрат ресурсів та часу на попередній аналіз та налаштування вхідних параметрів, так як для кожної задачі вони можуть суттєво відрізнятись.

Постановка задачі дослідження в контексті даної роботи полягає у побудові повноцінної системи, яка дасть змогу проаналізувати вплив кластеризації на моделі з використанням штучного інетелекту у трьох розрізах.

1. Використання різних методів кластеризації в рамках однієї нейронної мережі та оцінка параметрів вихідної моделі, що дасть змогу зрозуміти на скільки важливим є правильний підбір моделі.
2. Використання одного методу кластеризації з різною кількістю вихідних кластерів, що дасть змогу проаналізувати як впливає рівень деталізації на якість різницевої мережі.

3. Порівняння моделей з кластеризацією вхідних даних та без, що дасть змогу проаналізувати загальний вплив наявності кластеризації на якість інформації, отриманої для прийняття рішень.

РОЗДІЛ 2 ОБРАНІ МЕТОДИ І ПІДХОДИ ДО МОДЕЛЮВАННЯ

2.1 Формат текстових даних для нейронних мереж

В сучасному цифровому світі текстовий формат представлення даних є невід’ємною частиною обміну інформацією. Кожен день мільйони користувачів використовують цей спосіб для передачі різноманітних матеріалів: від простих текстових повідомлень до складних наукових текстів. Такий формат представлення є ефективним засобом спілкування, що забезпечує зручність, доступність та швидкість передачі даних. При цьому, кожний текст включає в себе наступне.

1. Зміст, який охоплює всі слова, з яких складаються фрази, речення, абзаци. Він може бути різної складності та характеру: прості текстові повідомлення, новини, художні створи, навчальні матеріали, ділові тексти, блоги та інше.
2. Структура тексту може бути як простою (одна фраза), так і складною з декількома абзацами, розділами та томами. Аналіз структури тексту дає змогу зрозуміти взаємозв’язок між отриманою інформацією.
3. Граматичні особливості та синтаксис також впливають на розуміння тексту, вони надають йому специфічних характеристик, які створюють додаткові взаємозв’язки.
4. Метадані тексту надають додаткові можливості опису тексту, зокрема автор, тема, видання та інше [11–13].

Саме через це прогрес у сфері штучного інтелекту включає в себе адаптацію до цього способу подання даних. Обробка природної мови (Natural Language Processing, NLP) [14] стала однією з ключових галузей в ШІ, спрямованих на аналіз та розуміння людських мовних конструкцій за допомогою комп'ютерів. Практично кожен аспект щоденного життя людини

вже охоплений застосуванням обробки природної мови, представленої через пошукові системи, автовиправлення, фільтрація спаму та багато інших прикладів.

Обробка текстів кличе за собою ряд проблем, з якими стикаються дослідники на етапі аналізу та обробки даних, серед яких визначення взаємозв'язків між словами, різноманітність значень слів, відмінки, розділові знаки та слова, мовні особливості, емоційна складова текстів та інше. Проте, не зважаючи на всі ці складові, існує ряд базових підходів, які допомогають спросити обробку інформації та підготувати дані до передачі в моделі штучного інтелекту.

Одиним із часто-вживаних підходів до обробки тексту є стемінг (англ. stemming) – це концепція, яка базується на скороченні слів до кореня або базової одиниці. Цей підхід дає можливість позбутись слів, які мають одну кореневу структуру, але різні закінчення, суфікси, відмінки, що спрощує дані для машинного аналізу [15]. Приклади поширених алгоритмів стемінгу: ISRIStemmer [16], SnowballStemmers [17], Porter stemmer [18].

Варто зауважити, що у випадку використання цих алгоритмів не враховується контекст речення, тому в деяких випадках може відбутись некоректна заміна, прикладом може бути слово «стільниковий» може отримати скорочення до «стіл», що призведе до некоректності даних. Тому перед застосуванням цих підходів варто провести додатковий аналіз даних, щоб не втратити контекст речення та застосовувати цей метод у комбінації з іншими.

Інша концепція обробки даних – лематизація (англ. Lemmatization). Ця концепція має більш точний процес вибору базової форми слова (леми), використовуючи лексичний аналіз та залучає базу даних WordNet. У відміну від стемінгу, лематизація враховує граматичні та морфологічні правила мови для знаходження коректної форми слова. Це процес є більш точним за стемінг, але потребує більших ресурсів і може бути неефективним у випадку обробки великих текстів та текстових баз даних.

Після проведення попередньої обробки тексту, важливим етапом, до передачі його у моделі штучного інтелекту, виступає векторизація, – процес перетворення даних з тексту у числове представлення. Для цього існує спеціальний інструмент TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency Vectorizer).

1. Частота термінів (TF): вимірює частоту певного слова в одному документі. Популярні слова в документі вважаються більш релевантними.
2. Інверсна частота документа (IDF): ця частина штрафує слова, які часто з'являються в багатьох документах, припускаючи, що вони мають менш конкретне значення. Рідкісні слова набувають більшого значення.

Наведений алгоритм часто використовується у випадку подібності документів, так як він дає можливіть виконувати такі завдання як кластеризація, виявлення плагіату та пошук інформації. Проте, він має і ряд недоліків, які варто брати до уваги при аналізі даних, серед яких: розрідженість вихідних векторів та необхідність коректного нагаштування вхідних параметрів.

Також, зважаючи на те, що вихідні вектори цього алгоритму можуть мати досить велику розмірність, то в комбінації з ним часто використовують метод TruncatedSVD (Truncated Singular Value Decomposition) [19], – метод лінійного алгебраїчного розкладу матриць, який використовується для стиснення матриць. Він працює шляхом обчислення лише n найбільших власних значень і власних векторів матриці, де n – це користувачем задане число компонентів.

Окрім вище зазначених варіантів представлення текстових даних, є ще один – категорійне представлення, за якого стовпець з даними має обмежений розмір (зазвичай одне – два слова у рядку) і налічує багато повторень (категорій). В такому випадку частою практикою є застовування алгоритму Labels Encoder [20], який створює словник відповідностей кожного слова або

словосполучення до його порядкового номеру у словнику, замінюючи таким чином текстові категорійні дані на числові.

2.2 Кластеризація

Інтелектуальний аналіз даних (ІАД) – це ефективний метод обробки інформації. Він спрямований на виявлення корисних, раніше невідомих, та нетривіальних інтерпретацій даних, необхідних для ухвалення рішень у різних галузях. Методи і алгоритми ІАД, зокрема такі як кластеризація, розвивалися на основі досягнень статистики, розпізнавання образів, штучного інтелекту та теорії баз даних [21]. Машинне навчання, взяте в найзагальнішому розумінні, представляє собою автоматичне розпізнавання структури набору даних, який може набувати різноманітних форм відповідно до поставленої мети. Отже, однією з ідей для виявлення структури є пошук груп або кластерів серед даних, цей процес називається кластеризацією.

Під поняттям кластеру в даному випадку розуміється підмножина даних, у якій інфомація більш схожа одна на одну, ніж на ту, що поза кластером. Приклад множини даних, поділених на кластери зображено на рисунку 2.1.

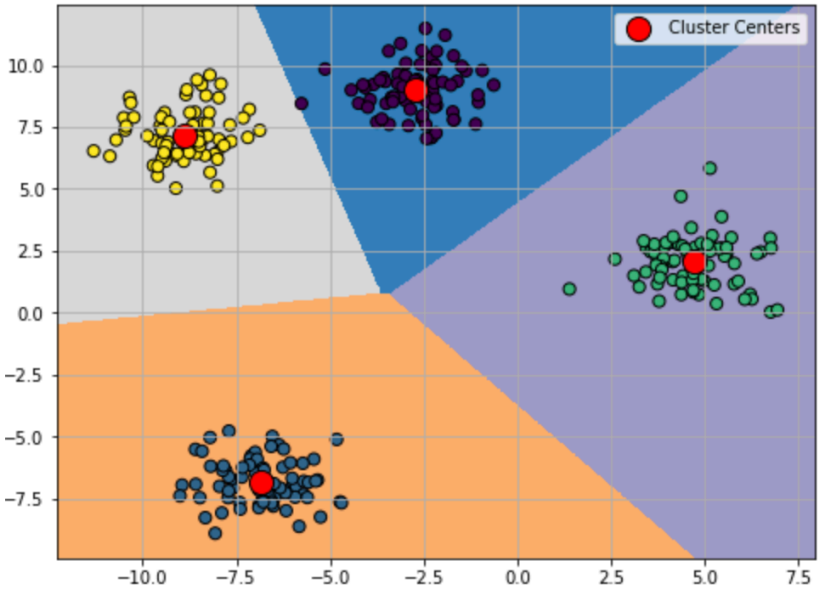

Рисунок 2.1 Приклад кластеризованих даних

Більшість алгоритмів кластеризації спираються на ключове поняття відстані між наборами наших даних. Це визначальний аспект, який допомагає алгоритмам в групуванні та організації даних у відповідності до їхньої схожості або відмінності.

Проте, пошук схожих груп даних є лише частиною з варіантів використання кластеризації. Вона може бути корисною у випадку стиснення даних, коли маємо занадто розріджений набір інформації і для подальшого аналізу необхідне перетворення. Також алгоритми кластеризації корисні у випадку прогнозування, так як на основі створених груп в певних задачах можна отримати більш точні показники та релевантні відповіді [22]. Задача кластеризації в розрізі текстових даних є одним з яскравих прикладів цьому, так як зазвичай набори текстів є досить розрідженими і для прогнозування необхідне виділення груп зі схожими характеристиками.

У даній дисертації розглядаються найбільш поширені алгоритми кластеризації для задачі взаємодії з текстовими даними [23]: KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model.

2.2.1 Модель кластеризації KMeans

Модель кластеризації KMeans є одним з найпопулярніших методів кластерного аналізу у машинному навчанні. Алгоритм базується на ітеративному процесі, де об'єкти даних представлені у вигляді точок у просторі $U \in \mathbb{R}^D$, а кожен кластер характеризується своїм центром, – точкою, яка є рівновіддалена від усіх точок кластеру. Приклад роботи алгоритму KMeans зображено на рисунку 2.2.

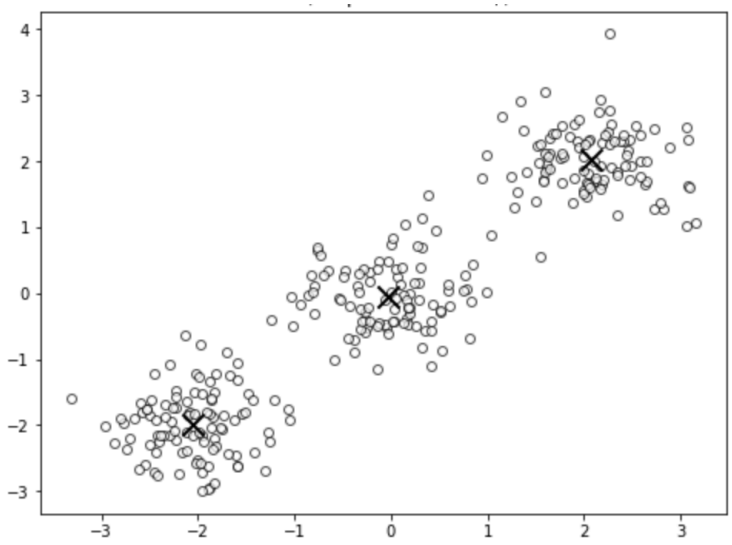

Рисунок 2.2 Приклад роботи KMeans

Основною метою даного алгоритму є мінімізація внутрішньокластерної дисперсії та максимізація відстані між кластерами. Де внутрішньокластерна дисперсія – це сума квадратів відстаней між двома точками та центром їхнього кластеру, яка розраховується за формулою:

$$\mathrm{var(k)} = \sum_{j=1}^{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \delta\left(\mathrm{c_i}, \mathrm{x_j}\right)^2, \ i = \overline{1,k}, \ j = \overline{1,n},$$

де $\delta\left(\mathrm{c_i}, \mathrm{x_j}\right)$ – евклідова відстань між центроїдом $\mathrm{c_i}$ та об’єктом $\mathrm{x_j}$; n – кількість об’єктів даних; k – кількість кластерів.

Це досягається через послідовні ітерації, де кожен об'єкт даних призначається до найближчого за відстанню центру кластеру, після чого обчислюється новий центр кластеру, який є середнім значенням координат всіх призначених йому об'єктів. Цей процес триває до збіжності, коли зміни в центрах кластерів стають незначними або змінюються менше ніж зазначене порогове значення.

Загально, алгоритм моделі KMeans полягає у виконанні наступних етапів.

1. Початковий вибір центроїда: k центроїдів випадковим чином вибираються з даних, або зазначаються конкретним базовим значенням.
2. Визначення кластерів: кожній точці даних ставиться у відповідність найближчий центр кластера на основі евклідової відстані.
3. Розрахунок нових центроїдів як середнього значення координат точок даних, призначених кластеру.
4. Порівняння нового центроїду із попереднім, за умови якщо різниця між ними суттєвіша за порогове значення, п.2–4 повторюються, інакше – розрахунок закінчується, що вказує на стабільність кластеру.

Необхідно зазначити, що вибір початкових центрів кластерів може вплинути на кінцевий результат, іноді призводячи до локальних мінімумів. Тому деякі варіації алгоритму KMeans можуть використовувати різні стратегії вибору початкових центрів для покращення результатів кластеризації.

З нюансів цього методу можна назвати важливість коректного підбору кількості кластерів, так як це впливає на різноманітність кластерів, та вибір метрики відстані. Хоч зазвичай базовою обирається Евклідова відстань, але іноді використовують і Манхеттенську, що може також повпливати на результати роботи моделі. Також важливим залишається початковий вибір центроїдів та наявність викидів у даних, що може спотворювати кінцеві результати.

2.2.2 Модель кластеризації Agglomerative Clustering

Агломеративна кластеризація, потужна техніка аналізу даних, відмінно підходить для виявлення природних угруповань у користувацьких даних шляхом побудови ієрархії подібності. Цей метод на початковому кроці розглядає кожну точку у вхідному наборі даних як окремий кластер, після чого на кожному кроці алгоритму об’єднуються пари найбільш схожих кластерів, зменшуючи загальну кількість кластерів. Цей процес триває доти, доки всі точки не об’єднаються у таку кількість кластерів, яка задана початковою умовою, або буде досягнуто початкову умову. Такий підхід «знизу вгору» контрастує з роздільною кластеризацією, яка починається з усього набору даних і ітеративно розділяє його на частини [24].

Для цієї моделі кластеризації ключовими факторами є визначення методу розрахунку відстані між кластерами, спосіб вибору пари кластерів для об’єднання та критерії зупинки роботи алгоритму.

Існує декілька методів розрахунку відстані між кластерами для цього методу. Розглянемо спрощені формули розрахунку відстані між двома точками.

1. Евклідова відстань – найпоширеніший метод обчислення відстані між двома точками у просторі $U \in \mathbb{R}^D$, який використовує наступну формулу:

$$d = (\sum_{l=1}^{N}(u_l^A - u_l^B)^2)^{\frac{1}{2}}, l = \overline{1,N}, \quad (2.1)$$

де N – кількість вимірів або розмірність простору; u_l^A та u_l^B – l-та координата точкок u^{A} та u^{B} відповідно.

2. Манхеттенська відстань вимірює суму абсолютних різниць між відповідними координатами точок. Розраховується за наступною формулою:

$$d = \sum_{l=1}^{N}\left|u_l^A - u_l^B\right|, l = \overline{1,N}, \quad (2.2)$$

де N – кількість вимірів або розмірність простору; u_l^A та u_l^B – l-та координата точок u^{A} та u^{B} відповідно.

3. Косинусна відстань вимірює кут між векторами точок у просторі, визначається за формулою:

$$d = 1 - \frac{u_1 \cdot u_2}{\|u_1\| \cdot \|u_2\|}, \tag{2.3}$$

де $u_1 \cdot u_2$ – скалярний добуток векторів u_1 та u_2; $\|u_1\| \cdot \|u_2\|$ – добуток норм векторів u_1 та u_2 відповідно.

4. Попередньо обчислена відстань. Метод Agglomerative clustering дозволяє використовувати власне обчислену матрицю відстаней між точками.

Також, важливим фактором є спосіб вибору пари кластерів для об'єднання, наведемо спрощені формули для розрахунку.

1. Ward – мінімізує дисперсію кластерів, які об'єднуються. Він обирає пару кластерів для об'єднання так, щоб додана дисперсія нового кластеру була мінімальною:

$$d_{ward} = \frac{|\hat{C}| \cdot |\check{C}|}{|\hat{C}| + |\check{C}|} \cdot d^2(\hat{C}, \check{C}),$$

де $|\hat{C}|$ та $|\check{C}|$ – кількості точок у скластерах $\hat{C}$ та $\check{C}$ відповідно; $d(\hat{C}, \check{C})$ – відстань між центроїдами кластерів $\hat{C}$ та $\check{C}$.

2. Average – використовує середню відстань між кожною точкою одного кластера та кожною точкою іншого кластеру для визначення відстані між кластерами в цілому.

$$d_{average} = \frac{1}{|\hat{C}| \cdot |\check{C}|} \sum_{q \in \hat{C}} \sum_{p \in \check{C}} d(x_q, x_p),$$

де $|\hat{C}|$ та $|\check{C}|$ – кількості точок у скластерах $\hat{C}$ та $\check{C}$ відповідно; $d(x_q, x_p)$ – відстань між точками x_q та x_p, яка розраховується за формулами (2.1) – (2.3).

3. Complete або maximum – використовує максимальну відстань між усіма парами точок з двох кластерів для визначення відстані між кластерами в цілому. Тобто відстань між кластерами визначається як

найбільша відстань між будь-якими двома точками, одна з яких належить одному кластеру, а інша – іншому [25].

$$d_{complete} = \max\{d(x_q, x_p): q \in \hat{C}, p \in \check{C}\},$$

4. Single або minimum – аналогічний до complete, але використовує мінімальну відстань.

$$d_{single} = \min\{d(x_q, x_p): q \in \hat{C}, p \in \check{C}\},$$

Орім цього, важливо зазначати критерії зупинки алгоритму, вони можуть бути наступними.

1. Фіксована кількість кластерів: алгоритм зупиняється, коли досягнута задана кількість кластерів.
2. Критерій відсічення на дендрограмі: дендрограма відображає послідовне об'єднання кластерів. Можна вибрати поріг на дендрограмі, який відповідає потрібній кількості кластерів, і зупинити об'єднання на цьому рівні.

В рамках цього методу, важливою є візуалізація отриманої дендрограми, деревоподібної структури, що зображує історію злиття кластерів. Аналіз дендрограми допомагає інтерпретувати ієрархічні зв’язки та визначити оптимальну кількість кластерів для вхідних даних.

2.2.3 Модель кластеризації Gaussian Mixture

Змішані моделі Гауса [26] (Gaussian Mixture Model, GMM) пропонують імовірнісний підхід до кластеризації, який моделює сукупність даних як комбінацію кількох гаусівських розподілів з різними параметрами.

Основна ідея GMM полягає в тому, що кожна точка даних генерується прихованим розподілом Гауса з невідомими параметрами. Модель оцінює ці параметри разом із пропорціями змішування, що представляють ймовірність

кожного кластера, використовуючи оцінку максимальної правдоподібності. Це відрізняє його від інших методів кластеризації, таких як KMeans, оскільки кожна точка може належати до кількох кластерів з певною ймовірністю.

Алгоритм GMM використовує метод EM (Expectation-Maximization) для навчання моделі. Процес навчання включає два основних кроки: E-крок (Expectation) та M-крок (Maximization):

– E-крок (Expectation) – на цьому етапі обчислюється ймовірність того, що кожна точка даних належить до кожного кластеру;
– M-крок (Maximization) – на цьому етапі оновлюються параметри кластера на основі отриманих ймовірностей.

Після кількох ітерацій цих двох кроків модель збігається до оптимального набору параметрів гаусівських розподілів, які найкраще описують структуру даних. Точки призначаються до кластерів з урахуванням їхньої ймовірності належності до кожного кластера.

Для запобігання перенавчання моделі застосовуються Байєсівський інформаційний критерій (Bayesian information criterion, BIC) і Інформаційний критерій Акаїке (Akaike information criterion, AIC) [27] – це статистичні критерії, які дозволяють оцінити якість моделі, враховуючи її точність та складність. Вони штрафують за складність моделі та дозволяють обрати найкращу модель серед кількох альтернатив.

AIC оцінює якість статистичних моделей, враховуючи точність та складність моделі. Він враховує не тільки точність моделі, але й кількість параметрів, що використовуються в моделі. Чим менше значення AIC, тим краще модель. Розраховується за формулою:

$$AIC = 2v - 2 \cdot \ln(L),$$

де v – кількість параметрів у моделі; L – максимальна функція правдоподібності моделі даних [28–30].

BIC схожий на AIC, але штрафує за складність сильніше. BIC використовує логарифм вибіркового обсягу даних для штрафування моделей з більшою кількістю параметрів. Розраховується за формулою:

$$BIC = v \cdot \ln(n) - 2 \cdot \ln(L),$$

де v – кількість параметрів у моделі; L – максимальна функція правдоподібності моделі даних, n – розмір вибірки.

Змішані моделі Гауса (GMM) мають свої переваги, зокрема можливість обробки даних з різними рівнями варіації та виявлення кластерів, які можуть накладатися один на одного. Однак, для їх успішної роботи потрібна детальна початкова настройка, і обчислення може бути витратними з обчислювальної точки зору, особливо для великих обсягів даних [31].

2.2.4 Модель кластеризації Latent Dirichlet Allocation

Прихований розподіл Діріхле (Latent Dirichlet Allocation, LDA) — це ймовірнісна модель, яка використовується для тематичного моделювання колекцій документів. Ця модель припускає, що кожен документ походить із суміші прихованих тем, причому кожне слово має ймовірність належності до кожної теми. Ця ймовірнісна структура дозволяє встановлювати різноманітні тематичні зв’язки в документах.

Оцінка параметрів у моделі LDA включає в себе ітераційний процес, який називається вибіркою Гіббса. Цей метод є ітеративним алгоритмом, використовуваним для оновлення тематичних приналежностей слів у документах, що розглядаються, з урахуванням розподілу тем у цих документах та взаємозв'язків між словами та темами.

Процес вибірки Гіббса розглядає кожне слово в документах і по черзі призначає йому нову тематичну приналежність, використовуючи інформацію про тематичний контекст слова та інших слів у документі [32].

Основними кроками моделі LDA є наступні.

1. Ініціалізація моделі – зазначення початкових значень матриць тем слів та документів тем.
2. Ітеративний процес навчання, що полягає у збільшенні ваг слів, які мають більшу відповідність з темою та документом та апроксимації тематичних розподілів для кожного документа та слова шляхом багаторазового оновлення.
3. Завершення навчання відбувається тоді, коли модель дослягає стану рівноваги, всі матриці отримано і вони описують структуру наведених документів.

Цей метод широко використовується для виявлення тематичних зв’язків у великих колекціях текстових даних, а також для сегментації та аналізу документів у залежності від їхнього змісту та тематики. Однак цей метод може бути чутливим до попередньої обробки даних і вибору кількості вихідних тем.

2.3 Складові базової згорткової одновимірної нейронної мережі

Одновимірна згорткова нейронна мережа – це інструмент глибокого навчання, який використовується для обробки послідовних даних у форматі одновимірних векторів, таких як текстові дані, часові ряди або сигнали. Вона складається з шарів, що здійснюють операції згортки над вхідними даними для виявлення в них шаблонів чи ознак. Кожен згортковий шар визначає певний патерн чи ознаку, яка розпізнається у вхідних даних, переміщуючись по всій послідовності [33–35].

Формально, згорткова операція у цій мережі здійснюється за допомогою зсуву фільтра (ядро згортки) по вхідним даним та обчислення скалярного добутку між вхідними даними та цим фільтром.

Формула згорткової операції для одновимірної згорткової нейронної мережі виглядає наступним чином:

$$S(r) = (X * W)[r] = \sum_{g=0}^{h-1} X[r+g] \cdot W[g] + \beta,$$

де $S(r)$ – значення згортки в позиції r; X – вхідні дані; W – ядро згортки (фільтр); h – розмір ядра згортки; β – зміщення.

Операція згортки в Conv1D виконує множення відповідних значень вхідних даних та ядра згортки для кожного кроку зсуву, додає зміщення β та формує вихідну послідовність значень.

Ця операція дозволяє моделі автоматично виявляти певні шаблони, ознаки чи корисну інформацію у вхідних даних за допомогою фільтрів, які ковзають по послідовності.

Основними складовими такої нейронної мережі є наступні.

1. Вхідні дані (Input) – одновимірний вектор або послідовність даних, яка передається на вхід моделі нейронної мережі.
2. Згорткові шари (Convolutional Layers) – шари, які містять фільтри (ядра) та які рухаються вздовж вхідних даних для виконання операції згортки.
3. Функція активації (Activation Function) – після кожного згорткового шару може застосовуватись нелінійна функція активації, така як ReLU (Rectified Linear Activation), щоб надати моделі нелінійність та здатність виражати складні залежності у даних.
4. Пулінгові шари (Pooling Layers) – шари пулінгу використовуються для зменшення розмірності виходу згорткових шарів шляхом об'єднання (пулінгу) інформації з певних областей. Найпоширеніші методи пулінгу – це максимальне пулінг та середнє пулінг.

5. Повністю зв'язані шари (Fully Connected Layers) – після згорткових та пулінгових шарів можуть слідувати повністю зв'язані шари. Ці шари приймають вектори ознак, отримані з попередніх шарів, та виконують класифікацію або регресію шляхом зв'язування кожного вузла з кожним вузлом попереднього шару.
6. Функція втрат (Loss Function) – це функція, яка вимірює різницю між прогнозованими значеннями моделі та правильними відповідями. Мета полягає в тому, щоб мінімізувати цю втрату під час тренування моделі.
7. Оптимізатор (Optimizer) – це алгоритм, який використовується для налаштування параметрів моделі з метою мінімізації втрати під час тренування, наприклад, алгоритми стохастичного градієнтного спуску (SGD), Adam, RMSProp тощо [36,37].

2.4 Основні показники якості моделі нейронної мережі

В оцінці якості моделей штучного інтелекту важливо враховувати декілька ключових показників, що допомагають визначити їхню ефективність та точність. Рекомендовані показники для визначення якості кінцевої моделі включають наступні.

1. Accuracy Score: цей показник визначає відсоток правильно класифікованих прикладів у відношенні до загальної кількості прикладів. Показник приймає значення з діапазону $[0, 1]$ та визначається за наступною формулою:

$$Accuracy\_score = \frac{TN + TP}{TP + FP + TN + FN},$$

де TP (True Positives) – кількість прикладів, які правильно були визнані як позитивні (правильно класифіковані як позитивні); TN (True Negatives) –

кількість прикладів, які правильно були визнані як негативні (правильно класифіковані як негативні); FP (False Positives) – кількість прикладів, які були невірно визнані як позитивні (невірно класифіковані як позитивні); FN (False Negatives) – кількість прикладів, які були невірно визнані як негативні (невірно класифіковані як негативні).

2. Precision: цей показник визначає відсоток правильно класифікованих прикладів у відношенні до загальної кількості прикладів. Показник приймає значення з діапазону $[0, 1]$ та визначається за наступною формулою:

$$Precision = \frac{TP}{TP+FP}. \quad (2.4)$$

3. Recall (чутливість): вимірює відсоток правильно ідентифікованих позитивних прикладів відносно всіх існуючих позитивних прикладів. Показник приймає значення з діапазону $[0, 1]$ та визначається за наступною формулою [38]:

$$Recall = \frac{TP}{TP+FN}. \quad (2.5)$$

4. F1-Score: представляє гармонічне середній між точністю і чутливістю моделі. Показник приймає значення з діапазону $[0, 1]$, у випадку якісної моделі його значення прямує до 1. Визначається за формулою:

$$F1\_Score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall},$$

де $Precision$ визначається за формулою (2.4); $Recall$ – за формулою (2.5).

5. Mean Absolute Error (MAE): вимірює середню абсолютну різницю між прогнозованими та фактичними значеннями. Найкраще значення – близько до 0, найгірше – велике числове значення. Визначається за формулою:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

де n – кількість прикладів у наборі даних; y_i – фактичне спостережене значення для і-го прикладу; $\hat{y}_i$ – прогнозоване значення моделі для i-го прикладу

6. Mean Squared Error (MSE): вимірює середню квадратичну різницю між прогнозованими та фактичними значеннями. Найкраще значення – близько до 0, найгірше – велике числове значення. Визначається за формулою:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \quad (2.6)$$

7. R-squared (R^2): вказує, наскільки змінні прогнозуються моделлю відповідають фактичним даним. Найкраще значення – 1 (ідеальне узгодження). Визначається за формулою:

$$R^2 = 1 - \frac{MSE}{VAR} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2},$$

де $\bar{y}_i = \frac{1}{n}\sum_{i=1}^{n} y_i$— середнє значення y_i; $VAR = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2$ – загальна дисперсія; MSE – визначається за формулою (2.6) [39].

2.5 Висновки до розділу 2

В цьому розділі було досліджено ключові аспекти аналізу текстових даних, описано їх основні застосування та важливість у сучасному контексті завдань. Текстові дані виявляються дійно актуальними в багатьох сферах, особливо у завданнях, пов'язаних з обробкою природного тексту.

Крім того, розглянуто основні методи кластеризації, які використовуються для обробки текстових даних та їх взаємодії. Кластеризація, як метод, полягає в групуванні подібних об'єктів разом у кластери на основі їхньої схожості або взаємозв'язків. Цей процес дозволяє впорядковувати

інформацію та відокремлювати її на групи зі схожими характеристиками, що робить подальший аналіз більш зрозумілим та ефективним.

Було зосереджено увагу на деяких популярних моделях кластеризації, таких як KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model. Кожен з цих методів має свої особливості та застосування в аналізі тексту, дозволяючи розв'язувати різноманітні завдання обробки і класифікації інформації.

Додатково, розглянуто основні аспекти нейронних мереж. Нейронні мережі представляють собою потужний інструмент для вирішення завдань, пов'язаних з аналізом тексту, розпізнаванням образів, передбаченням та іншими областями. Їхній успіх полягає у здатності самостійно вивчати складні залежності у даних, роблячи прогнози або здійснюючи класифікацію з високою точністю.

Нейронні мережі відкривають нові можливості для аналізу текстових даних, допомагаючи у виявленні взаємозв'язків та структур, які можуть залишатися невидимими для класичних методів обробки інформації. Їх важливість полягає в здатності автоматизувати та поліпшувати процеси аналізу великих обсягів текстової інформації, що є критичним у сучасному світі даних.

РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ, АНАЛІЗ І ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ

Мета даного розділу полягає в проведенні емпіричного дослідження з ціллю оцінки впливу кластеризації на якість рішень в моделях штучного інтелекту. Конкретні цілі включають наступні:

- порівняння результатів, отриманих в моделях із використанням кластеризації, з результатами моделей без кластеризації;
- визначення оптимальних параметрів кластеризації для підвищення ефективності моделей;
- формулювання висновків та рекомендацій для використання кластеризації в практичних застосуваннях моделей штучного інтелекту.

Результати цього розділу можуть бути корисними зокрема для розробників та дослідників у галузі штучного інтелекту, а також для вирішення конкретних завдань, пов'язаних з обробкою даних та впровадженням інтелектуальних систем.

3.1 Методика оцінки впливу

У рамках дослідження передбачено розробку повноцінної системи для взаємодії з текстовими (наприклад новини) та числовими вхідними даними з метою надання користувачам рекомендацій щодо тематики новин, що враховують їхні індивідуальні параметри.

Експеримент також передбачає акумуляцію та аналіз показників якості різних моделей з метою подальшого їх порівняння та виявлення рівня впливу

обраного методу кластеризації на якість рішень та визначення впливу кількості кластерів на результати.

Обраний підхід відкриває нові можливості для розуміння процесу кластеризації та його взаємодії з нейронними мережами.

3.1.1 Дизайн експерименту

У рамках дослідження, було розроблено комплексний експеримент, який складається з кількох послідовних етапів. Основні положення систем подібного типу було описано у [40].

У початковій фазі дослідження проводиться аналіз та обробка даних, які є критичним етапом перед розгортанням моделей. Цей етап передбачає повний цикл обробки даних, що включає завантаження певного обсягу вхідних даних з датасету, відбір ключових ознак, проведення процедур очистки від шуму та підготовку текстової інформації до подальшого використання в моделях шляхом перетворення її у числові або векторні формати, які стають зрозумілими та адаптованими для подальшого використання в алгоритмах моделей.

Другий етап полягає у порівнянні двох різних підходів аналізу даних. Перша варіація використовує методи кластеризації, де текстова інформація перетворена у числові вектори, які репрезентують складну структуру тексту. Для цього варіанту експерименту використовуються моделі кластеризації, такі як KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model. Ці моделі відображають різноманіття підходів та рівнів аналізу даних, дозволяючи порівняти їхню ефективність у розпізнаванні та групуванні схожих патернів у текстових даних. Такий порівняльний аналіз спрямований на визначення оптимальних стратегій обробки та категоризації

текстової інформації з метою підвищення точності та рівня інтерпретації отриманих результатів.

В другому варіанті дослідження використовується глибоке навчання, зокрема нейронні мережі, для проведення аналізу та класифікації текстових даних. Нейронна мережа отримує на вході текстові дані у векторному представленні, яке є результатом унікального кодування слів або фраз у числові вектори, що відображають семантичні зв'язки у тексті. У процесі навчання модель отримує текстові вектори та вивчає зв'язки між ними та категоріями, які вони відображають.

Ця варіація експерименту спрямована на використання глибокого навчання для автоматичного призначення порядкових номерів тематик новин, що є важливим етапом у розумінні та категоризації текстових даних. Нейронна мережа вивчає складні неявні залежності між вхідними векторами та категоріями тематик новин шляхом вагової корекції інформації у різних шарах мережі.

У відповідності до методології наукового дослідження, отримані результати експерименту обох варіацій подаються у якості цільових значень для основної нейронної мережі. Ця нейронна мережа є ключовим елементом у процесі обробки та аналізу вхідних даних, приймаючи параметри, що визначаються на етапі ініціалізації програми, та формуючи вектор ознак, які призначені для віднесення новин до відповідних тематик, які максимально відповідають індивідуальним потребам та інтересам конкретного користувача.

Завершальний етап дослідження передбачає конструювання кінцевої моделі на основі отриманих даних від обох варіацій. Це надає можливість для порівняння результатів на тестових вибірках, використовуючи раніше описані метрики якості моделей. Цей етап дослідження спрямований на аналіз та порівняння ефективності обох варіацій моделей у контексті їхньої здатності відповідати поставленій задачі та враховувати індивідуальні потреби користувачів. Результати порівняння дозволять визначити найкращу модель, що відповідає поставленій меті та встановити необхідний рівень впливу,

сприйнятого та застосованого для досягнення задекларованої мети. Загальна схема експерименту представлена на рисунку 3.1.

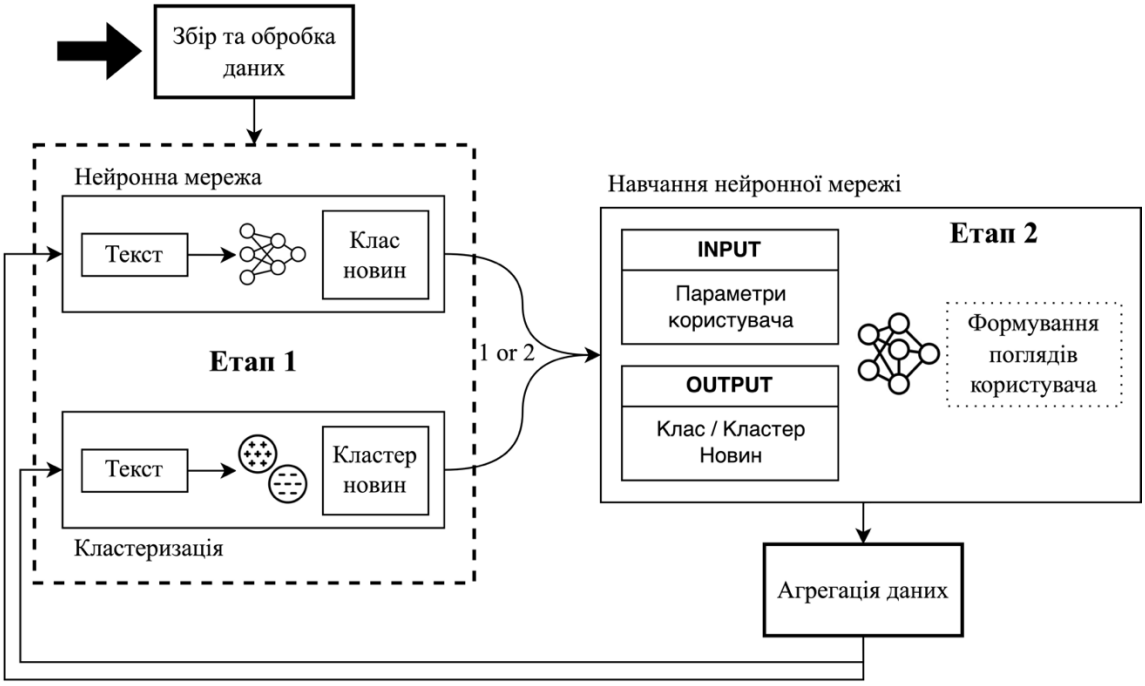

Рисунок 3.1 Загальна схема експерименту

3.2 Збір та підготовка даних

Вхідні дані, використані у цьому науковому дослідженні, складаються зі сполучення даних двох різних джерел: набору даних "20newsgroups" [41] та недавно зібраних даних з користувачів соціальної мережі Twitter (нова назва "X"). Набір даних "20newsgroups" містить інформацію про новини та їх тематику, представлену у вигляді текстових описів різних новинних груп. У той же час дані з соціальної мережі Twitter відображають інтереси користувачів, які переглядали ці новини, доповнені числовими та категоріальними характеристиками.

Сполучення цих двох джерел даних створює велику складність та різноманітність вхідних даних. Вхідні дані представлені як текстові описи, що включають у себе інформацію про новини та їхню тематику, так і числові та категоріальні характеристики, як це показано у таблиці 3.1.

Таблиця 3.1 – Приклад початкового формату даних

Text	**Target name**	**Client ID**	**Age**	**Education**	**Marital status**	**Sex**	**Income**
sure basher pen fan pretti 8 …	rec.sport.hockey	9625	17	Bachelors	Married-spouse-absent	Female	<=50K
brother market highperform, …	comp.sys.ibm.pc.hardware	2968	32	Bachelors	Married-civ-spouse	Male	>50K
>final said dream mediterranean …	talk.politics.mideast	16151	29	HS-grad	Married-civ-spouse	Female	>50K
think scsi card dma transfer disk: …	comp.sys.ibm.pc.hardware	3341	40	HS-grad	Divorced	Female	<=50K
old jasmin drive cannot use new …	comp.sys.mac.hardware	3696	6	9th	Never-married	Female	<=50K

Згідно з таблицею 3.1, до текстових даних належить колонка Text; до категорійних – колноки Target name, Education, Material status, Sex, income; до числових – Client ID та Age.

Перед початком аналізу та моделювання було проведено етап попередньої обробки текстових даних, зокрема в колонці "Text". Цей процес включає в себе видалення спеціальних символів, чисел, повторювальних пропусків тощо. Далі були застосовані процеси стемінгу та лематизації тексту.

Під час обробки текстових даних також були вилучені функціональні слова англійської мови, такі як "at", "a", "the", тощо. Це зроблено з метою виключення слів, які не несуть значущої інформації та можуть додавати шум до аналізу. Остаточна мета цих операцій полягала в покращенні якості текстових даних та підготовці їх для подальшого аналізу та моделювання.

Після процесу очистки текстових даних було застосовано модель **TfidfVectorizer**, яка дозволила перетворити їх на вектори. Ця модель враховує як частоту вживання слова в конкретному документі (TF – term frequency), так і інверсійну частоту вживання слова у всьому корпусі даних (IDF – inverse document frequency). Цей підхід допомагає виділити важливі та рідкісні слова, підвищуючи точність описуваного тексту та знижуючи вплив загальних слів.

Крім того, для подальшого покращення ефективності обробки та зменшення розмірності отриманих векторів ознак була використана модель **TruncatedSVD**. Цей метод зменшує розмірність даних, зберігаючи при цьому найважливіші компоненти. Такий підхід був корисний, так як кількість ознак була великою, але багато з них не несли значущої інформації.

Крім текстових даних, також було проведено обробку категорійних даних (Target Name, Education, Martial Status, Sex, Income). Процес очистки категорійних даних включав в себе кілька важливих кроків.

1. **Кодування категорій**: початкові категорійні дані представлені у формі текстових міток, таких як "чоловік", "жінка", "освіта", "сімейний статус" і т. д. Для того, щоб ці дані можна було використовувати в математичних моделях, їх необхідно перетворити на числові значення. Для цього процесу було застосовано один із поширених методів – **Label Encoding**, де кожній унікальній категорії присвоюється унікальне числове значення (наприклад, 0, 1, 2 і т. д.). Це спрощує подальшу обробку даних.
2. **Обробка відсутніх значень:** у реальних даних часто зустрічаються відсутні значення в категорійних даних. Так як таких даних було відносно небагато (< 0.1%), то було прийнято рішення видалити їх [42].
3. **Очищення та обрізка:** деякі з категорій містили ненормалізовані або непотрібні символи, їх також було вилучено [43].

3.3 Опис побудованих моделей

У контексті практичної реалізації даного наукового дослідження відбулася розробка та втілення широкого спектру моделей різного формату, спрямованих на вирішення завдань кластеризації в текстових даних. Моделі першого типу були створені з використанням повноцінних функціональних класів, що забезпечують розширені можливості налаштування моделей. Ці класи не лише надають можливість адаптувати моделі до специфічних потреб дослідження, а й увімкнули можливості маніпулювання параметрами, зміну кількості кластерів, оптимізацію вхідних даних та інші варіації, що сприяють більш глибокому аналізу та експериментам.

До моделей першого типу належать наступні моделі кластеризації.

1. **KMeans:** цей метод є одним із найпоширеніших алгоритмів кластеризації, який розділяє дані на групи на основі схожості між точками даних. Він базується на мінімізації середньоквадратичного відхилення між точками і центрами кластерів.
2. **Agglomerative Clustering:** цей метод використовує ієрархічний підхід до кластеризації, починаючи з окремих точок і об'єднуючи їх у кластери шляхом послідовного об'єднання найближчих точок.
3. **Gaussian Mixture:** ця модель базується на суміші гауссівських розподілів та припускає, що дані генеруються з декількох гауссівських розподілів, кожен з яких представляє окремий кластер.
4. **Latent Dirichlet Allocation Model (LDA):** ця модель використовується для кластеризації текстових даних та визначення тематичної структури документів, де кожен документ розглядається як суміш тем.

Для об'єктивної оцінки ефективності кластеризаційних моделей та порівняння їх результатів на подальших етапах аналізу було використано модель Sequential. Ця модель представляє собою складну структуру, що

включає послідовні шари нейронів, де вихід одного шару є вхідним сигналом для наступного, створюючи послідовність обчислень.

У цьому дослідженні ця мережа була піддана навчанню з метою виконання завдання передбачення тематики текстових даних. Для цього використовувалися вектори ознак, отримані в результаті попередньої обробки та векторизації тексту. Під час тренування моделі вона вчилася розпізнавати тематичні зв'язки та шаблони у текстах на основі навчального набору даних.

На рисунку 3.2 наведено структуру побудованої моделі, де кожен шар нейронів відображає конкретний рівень абстракції та обробки інформації.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 298, 64)           256

 max_pooling1d (MaxPooling1D  (None, 149, 64)          0
 )

 conv1d_1 (Conv1D)           (None, 147, 128)          24704

 max_pooling1d_1 (MaxPooling  (None, 73, 128)          0
 1D)

 flatten (Flatten)           (None, 9344)              0

 dense (Dense)               (None, 64)                598080

 dense_1 (Dense)             (None, 20)                1300

=================================================================
Total params: 624,340
Trainable params: 624,340
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.2 Структура побудованої нейронної мережі для класифікації векторів текстових даних

На основі представленого на рисунку 3.2, нейронна архітектура включає наступні шари.

1. **Conv1D Layers (Згорткові шари):** ці шари відображають першу стадію аналізу, вони працюють з послідовними даними, використовуючи згорткові нейрони (Conv1D). У моделі реалізовано

два таких шари з метою виявлення та виділення важливих патернів у тексті.

2. **MaxPooling1D Layers (Пулінгові шари):** після кожного згорткового шару використовуються пулінгові шари, які зменшують просторові розміри отриманих виходів згорткових шарів. Ці шари зберігають найбільш важливі ознаки, сприяючи скороченню обчислювальних витрат та покращенню ефективності моделі.
3. **Flatten Layer (Шар розгортання):** після пулінгових шарів дані конвертуються у одновимірний вектор. Це перетворення дозволяє зв'язати зібрані ознаки з подальшими повнозв'язними шарами.
4. **Dense Layers (Повнозв'язні шари):** у моделі використовуються два повнозв'язних шари (Dense). Перший шар, що має 64 нейрони, активується функцією ReLU (Rectified Linear Unit). Цей шар відповідає за виявлення більш складних патернів у векторизованих даних. Другий повнозв'язний шар має кількість нейронів, яка відповідає кількості класів у завданні класифікації. Він використовує функцію активації softmax для отримання ймовірностей належності кожного вхідного зразка до різних класів.

Після визначення архітектури моделі проводиться її компіляція, що включає встановлення параметрів, необхідних для оптимізації та навчання мережі. У якості функції втрат для оцінки різниці між прогнозованими та фактичними значеннями використовується **Categorical Crossentropy** (категоріальна перехресна ентропія). Ця функція особливо підходить для задач класифікації, де вхідні дані можуть належати до різних класів.

Оптимізатор **Adam** використовується для налаштування ваг моделі під час процесу навчання. Вибір цього конкретного оптимізатора базується на його ефективності та популярності у глибокому навчанні. Однією з основних переваг Adam є його адаптивний підхід до кроку навчання. Він комбінує основні ідеї алгоритмів градієнтного спуску з адаптивним коефіцієнтом швидкості навчання.

Adam адаптує крок навчання для кожного параметра, використовуючи історію градієнтів для цього параметра. Це дозволяє алгоритму ефективно пристосовуватися до складних умов оптимізації, уникати перешкод у процесі навчання та прискорювати збіжність моделі до оптимального розв'язку [44].

3.4 Опис користувацького інтерфейсу

Зовнішній інтерфейс створеної системи реалізовано у форматі Telegram-чатботу [45] з можливістю надсилати команди різного типу для налаштувань та взаємодії. Початковий вигляд представлено на рисунку 3.3.

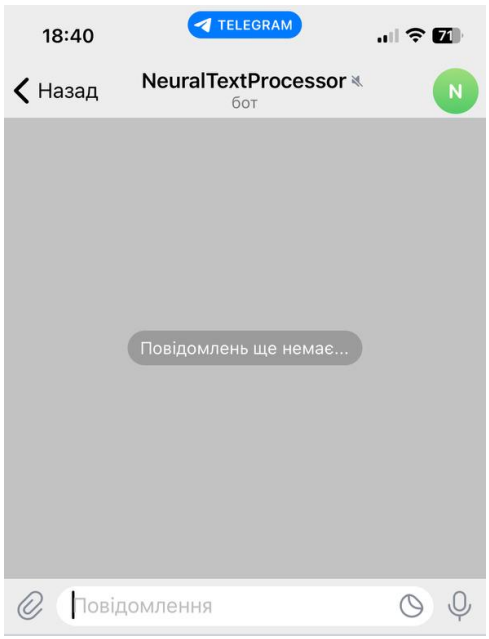

Рисунок 3.3 Початковий стан системи

Відновідно до рисунку 3.3 для того, щоб почати взаємодію з ботом достатньо надіслати повідомлення у форматі **/start**. Одразу після цього система надішле привітальне повідомлення з основними командами для взаємодії. Результат зображено на рисунку 3.4.

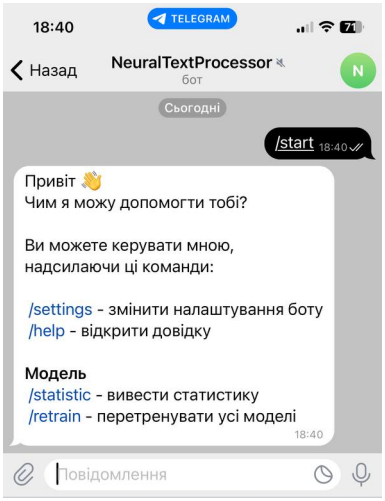

Рисунок 3.4 Початкове повідомлення

Згідно з рисунком 3.4 – перший можливий варіант взаємодії з системою – це зміна налаштувань програми (**/settings**). До параметрів, які можна змінити, відносяться: базова кількість кластерів, яка передається в моделі кластеризації та базова модель кластеризації. Обидва параметри необхідні для того, щоб в подальшому можна було тестувати систему на різних початкових налаштуваннях і порівнювати отримані результати. Зовнішній вигляд названих налаштувань представлений на рисунку 3.5.

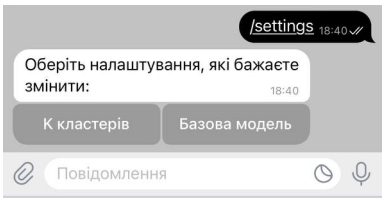

Рисунок 3.5 Варіанти зміни базових налаштувань

Після вибору кнопки «К кластерів» (рисунок 3.5), буде відображено наступне діалогове вікно (рисунок 3.6). Зміна кількості кластерів відбувається за допомогою кнопок «+» та «-», можна змінити кількість від 10 до 50 кластерів. У випадку, якщо після зміни кількості кластерів не натиснути кнопку зберегти, то кількість повернеться до попередньої.

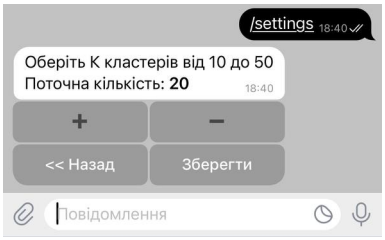

Рисунок 3.6 Зміна кількості кластерів

Також, у системі передбачено декілька видів повідомлень щодо результатів зміни цього налаштування. Повідомлення першого типу – успішне збереження, з'являється якщо після зміни кількості натиснути кнопку «Зберегти» та система успішно збереже нову кількість кластерів. Приклад повідомлення зображено на рисунку 3.7.

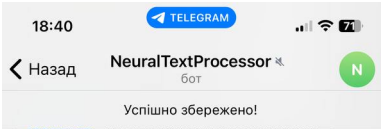

Рисунок 3.7 Успішне збереження нової кількості кластерів

Якщо під час зміни кількості кластерів буде введено занадто мало кластерів (менше 10), то буде відображено повідомлення (попередження) наступного вигляду [46] (рисунок 3.8).

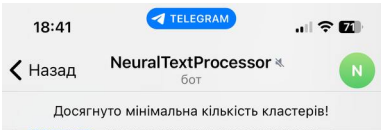

Рисунок 3.8 Помилка: замало кластерів

Зворотнім випадком до рисунку 3.8 є повідомлення на рисунку 3.9 – занадто велика кількість кластерів (більше 50).

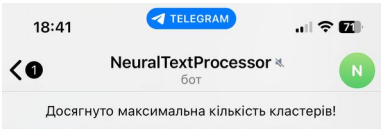

Рисунок 3.9 Помилка: забагато кластерів

Другий тип налаштувань системи – зміна основної моделі кластеризації. Обрана модель використовується для обробки вхідних повідомлень у вигляді звичайного тексту (не команд). Серед запропонованих моделей присутні: KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model.

Згідно з рисунком 3.10, поточною моделлю зазначена Agglomerative, але за необхідності її можна змінити. Для прикладу оберемо модель KMeans, для цього необхідно натиснути відповідну кнопку.

Рисунок 3.10 Вибір базової моделі кластеризації

Рисунок 3.11 демонструє повідомлення, про успішну зміну моделі. Далі ми маємо можливість повернутись на попередній крок.

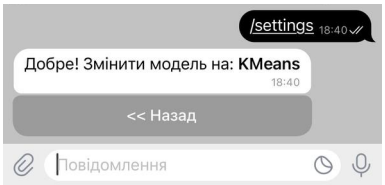

Рисунок 3.11 Результат зміни базової моделі кластеризації

При поверненні на етап вибору моделі кластеризації (рисунок 3.12) маємо оновлену інформацію про поточну модель, надалі вона буде використовуватись для обробки даних.

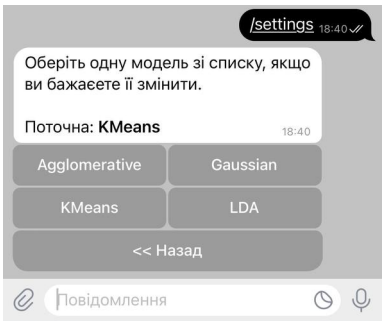

Рисунок 3.12 Нові налаштування базової моделі кластеризації

Відповідно до рисунку 3.4 – наступний етап взаємодії з системою – отримання короткої довідки про програму. Для цього, необхідно або ввести **/help**, або натиснути на відповідний пункт в привітальному повідомленні.

На рисунку 3.13 можна побачити коротку довідку, яка містить автора програми, назву теми та декілька основних можливостей системи.

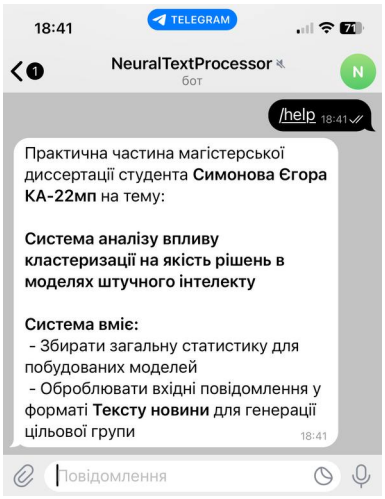

Рисунок 3.13 Довідка про програму

Окрім названих пунктів, модель можна перенавчити вручну, для цього необхідно надіслати команду **/retrain**.

Після того, як була надіслана команда **/retrain**, буде виведено повідомлення про початок перенавчання (рисунок 3.14), для завершення процесу необхідно буде почекати відповідного повідомлення про завершення тренування. Так як моделей багато, то процес може бути довготривалий. За для зменшення часу обробки даних на наступних етапах, у всіх моделей передбачено збереження навчених вагів для подальшого завантаження та використання.

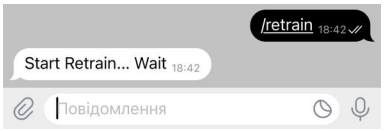

Рисунок 3.14 Перенавчання моделі

Також, система може збирати звіт про якість роботи навчених моделей. Для його формулювання необхідно надіслати відповідну команду **/statistic**.

Після відправки необхідної команди буде відображено повідомлення про стан системи з поточним етапом обробки (рисунок 3.15).

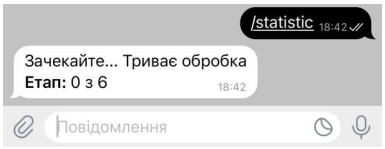

Рисунок 3.15 Формулювання звіту про якість побудованих моделей

Кожен етап формулювання звіту буде оновлювати повідомлення з етапами, формат зображено на рисунку 3.16.

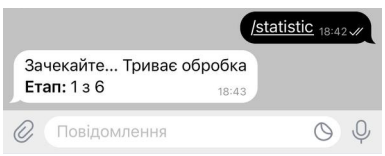

Рисунок 3.16.Зміна етапів формулювання звіту

Разом зі зміною повідомлень у Telegram-чатботі на стороні консолі (під час виконання програми), буде відображатись додаткова інформація про етап обробки та залишок часу.

Відповідно до рисунку 3.17 можна побачити, скільки часу займає одна ітерація, кількість ітерацій, що вже було здійснено та залишок.

```
Етап: 1 з 6
5/5 [==============================] – 55s 11s/step
58/58 [==============================] – 0s 5ms/step
572/572 [==============================] – 2s 4ms/step
Етап: 2 з 6
50/50 [==============================] – 27s 530ms/step
58/58 [==============================] – 0s 973us/step
Етап: 3 з 6
50/50 [==============================] – 29s 557ms/step
58/58 [==============================] – 0s 1ms/step
Етап: 4 з 6
50/50 [==============================] – 29s 563ms/step
58/58 [==============================] – 0s 920us/step
Етап: 5 з 6
50/50 [==============================] – 31s 608ms/step
58/58 [==============================] – 0s 1ms/step
Етап: 6 з 6
50/50 [==============================] – 28s 547ms/step
58/58 [==============================] – 0s 1ms/step
```

Рисунок 3.17 Додаткова інформація про етапи обробки

Після завершення процесу обробки, буде виведено повідомлення у форматі представленому на рисунку 3.18.

Рисунок 3.18 Результат формування звіту

Результат, зображений на рисунку 3.18, представляє собою повідомлення у вигляді таблиці, де стовпцями будуть усі моделі, аналіз яких проводила система, а рядками – метрики за якими відбувалась оцінка. За необхідності результат можна зберегти собі на пристрій, або переслати комусь іншому (вбудований функціонал Telegram).

3.5 Аналіз результатів

Розглядаючи отримані результати, переходимо до аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту. Аналіз проводиться в двох ключових аспектах: по-перше, вивчаємо вплив обраного методу кластеризації, а по-друге – аналізуємо вплив різної кількості кластерів для на якість кінцевих моделей (Етап 2, рисунок 3.1). Цей підхід дозволяє отримати глибше розуміння того, як параметри кластеризації впливають на результати в моделях штучного інтелекту.

3.5.1 Вплив методу кластеризації на якість рішень в моделях штучного інтелекту

Базові налаштування системи: для проведення аналізу, зазначимо цільову кількість кластерів рівною 20. Для оцінки результатів у заданому контексті, звернемось до результатів виводу системи після виконання команди /statistic (рисунок 3.19):

Metrics	Basic	KMeans	Agglomerative	Gaussian	LDA
Accuracy Score	0.7294	0.8497	0.8425	0.8682	0.8316
Recall	0.7294	0.8497	0.8425	0.8682	0.8316
F1-Score	0.7818	0.8409	0.8418	0.8634	0.8294
Mean Absolute Error	0.9956	0.6517	0.6189	0.3811	0.3548
Mean Squared Error	4.3031	3.7551	3.1219	1.9918	0.9169
R-squared	0.4981	0.5596	0.5152	0.632	0.6429

Рисунок 3.19 Результати моделі ШІ з та без моделей кластеризації

На рисунку 3.19 колонками виступають назви моделей, аналіз яких проводився системою. Назви рядків відповідають за метрики, розрахунок яких проводився.

У рамках дослідження було проведено аналіз декількох моделей, призначених для класифікації даних на основі параметрів користувача. Зокрема, порівнювалися модель базового типу (Basic), яка не використовує кластеризацію, з моделями, що включають в себе кластеризаційні підходи, такі як KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model.

Результати дослідження свідчать про високу ефективність поєднання нейронної мережі з моделю кластеризації Gaussian Mixture в контексті поставленої задачі:

1. **Accuracy Score (Точність класифікації):** модель з використанням Gaussian Mixture має Accuracy Score на рівні 0.8682, що означає, що вона правильно класифікувала близько 86.82% прикладів. Це є найвищим результатом порівняно з іншими моделями і свідчить про її здатність точно передбачати класи на основі вхідних параметрів користувача
2. **Recall та F1-Score**: побудована модель також володіє найвищими показниками **Recall** (0.8682) та **F1-Score** (0.8634), що вказує на високу здатність відшукати всі позитивні приклади та досягти балансу між точністю та повнотою класифікації.
3. **Mean Absolute Error (Середня абсолютна похибка):** побудована модель має один з найнижчих показників Mean Absolute Error (0.3811), що свідчить про мінімальну похибку у прогнозуванні.
4. **Mean Squared Error (Середня квадратична похибка):** модель також має низький показник Mean Squared Error (1.9918), що підкреслює точність її прогнозів.
5. **R-squared (R2):** R-squared для даної моделі дорівнює 0.632, що є високим показником. Це свідчить про добру здатність моделі пояснювати зміну в залежній змінній та адекватність її прогнозів.

3.5.2 Вплив кількості кластерів на якість рішень в моделях штучного інтелекту

Для проведення аналізу впливу кількості кластерів на якість рішень в моделях штучного інтелекту, повторно звернемось до результатів роботи команди **/statistic** (рисунок 3.18), але попередньо змінимо базові налаштування системи (**/settings**), а саме – вихідну кількість кластерів.

В рамках даного дослідження було перевірено моделі з різною кількістю вихідних кластерів від 10 до 50, найбільш показовими стали кластери 10, 15, 20 та 50. Цільовою метрикою дослідження було обрано – Accuracy Score.

Результати для 20 кластерів вже були проаналізовані та представлені раніше (рисунок 3.19). Точність для моделей вказана в таблиці 3.2:

Таблиця 3.2. – Точність нейронної мережі з моделями з 20 кластерами

N Clusters	**KMeans**	**Agglomerative**	**Gaussian**	**LDA**
20	0.8497	0. 8425	0. 8682	0. 8316

Після експериментів з варіюванням кількості вихідних кластерів для обробки даних, отримали наступну статистику (таблиця 3.3):

Таблиця 3.3. – Точність нейронної моделі з моделями з різною кількістю вихідних кластерів

N Clusters	**KMeans**	**Agglomerative**	**Gaussian**	**LDA**
10	0,8436	0,8464	0,8507	0,8387
15	0,8409	0,8311	0,8486	0,8316
20	0,8497	0,8425	0,8682	0,8316
30	0,8612	0,8465	0,8721	0,8382
50	0,8518	0,8365	0,8557	0,8309

Як бачимо, в усіх проведених експериментах лідером виявилась нейронна мережа із застосуванням кластеризації за допомогою Gaussian Mixture. Візуалізуємо отримані результати (рисунок 3.20):

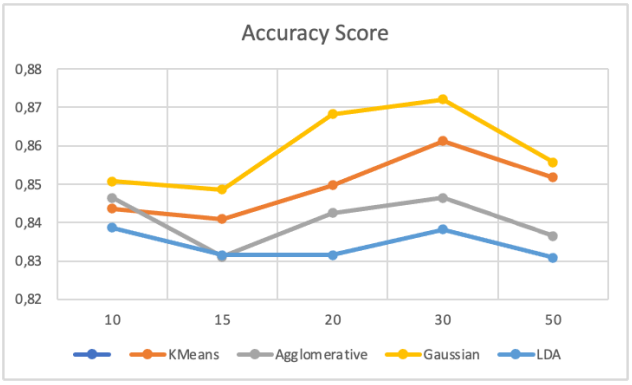

Рисунок 3.20. Візуалізація результатів експерименту

Спостерігаємо схожу динаміку для всіх побудованих моделей, що до 30 вихідних кластерів точність зростає і досягає свого максимуму, після чого спостерігається динаміка на спад.

Також, рисунок 3.20 дає можливість зрозуміти, що присутня шукана залежність між кількістю вихідних кластерів та точністю нейронної мережі.

3.6 Висновки до розділу 3

У цьому розділі була розроблена високофункціональна система, у форматі Telegram-чатботу, для аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту. Ця система визначається своєю гнучкістю та можливістю змінювати базові налаштування зручним користувацьким інтерфейсом, а також формувати звіт, який включає основні метрики для подальшого аналізу.

Застосування цієї системи дало можливість ретельно проаналізувати два аспекти: вплив обраної моделі кластеризації та вплив кількості вхідних кластерів на якість моделей нейронної мережі.

Перше дослідження вказало на те, що моделі, які включають в себе процес кластеризації, проявили кращі результати порівняно з моделями без нього. Найкращим вибором виявилась модель з використанням кластеризації Gaussian Mixture, яка відзначилась найкращою точністю та іншими ключовими показниками якості.

Друге дослідження виявило, що залежність точності нейронної мережі з кластеризацією від кількості кластерів – присутня. Оптимальною кількістю вхідних кластерів для наданих даних є 30. У цьому випадку моделі нейронної мережі показують найкращі результати.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

У сучасному світі спостерігається активний розвиток сервісів, що використовують штучний інтелект для вдосконалення функціональності та надання нових можливостей користувачам. Серед таких сервісів є DALL-E 2, Whisper, GPT-4, Make-A-Video. Важливим аспектом цього розвитку є обробка великих обсягів даних, що використовуються для тренування моделей. Однією з ключових складових цього процесу є кластеризація даних, яка допомагає вирішити завдання агрегації та отримання загальних характеристик даних для подальшої передачі в моделі ШІ. Отже запропонований стартап є актуальним і буде користуватись попитом на ринку.

У даному розділі проводиться маркетинговий аналіз стартап-проекту з метою визначення його ринкової придатності та потенційних напрямків розвитку.

4.1 Формування стратегії розвитку стартапу

Для життєздатності будь-якого стартапу необхідна детально-продумана стратегія його побудови та розвитку. До основних етапів становища стартапу як цілісного продукту та конкурентоспроможної організації можна віднести наступні.

1. **Формування ідеї та дослідження цільового ринку**. Визначення основного напрямку роботи стартапу та детальний аналіз ринку для визначення рівню попиту на запропонований продукт (або послугу), визначення конкурентів

2. **Розробка якісного бізнес-плану.** Має включати в себе стратегію розвитку, маркетингові та фінансові аспекти, наявні та необхідні ресурси. Визначення необхідних інвестицій.
3. **Побудова робочого прототипу та тестування.** Прототип необхідний для демонстрації запланованої ідеї потенційним інвесторам та користувачам проекту. Також за допомогою нього можна отримати зворотній зв’язок щодо покращення продукту від перших користувачів.
4. **Вдосконалення та масштабування проекту.** Впровадження виправлень, отриманих від перших користувачів. Підготовка до масштабування та розширення функціональності.
5. **Залучення інвестицій.** Вивчання інвестиційних можливостей та підготовка пропозиції потенційним інвесторам.
6. **Маркетинг та підготовка до виведення на ринок.** Побудова деталізованої маркетингової стратегії, яка буде включати рекламу, способи просування, бюджет тощо. Побудова партнерських відносин та визначення початкових каналів продажу.
7. **Виведення на ринок та підтримка.** Офіційний реліз продукту (послуги) на ринок та постійна його підтримка, взаємодія з клієнтами та покращення продукту. Корегування стратегії також входить у цей етап, так як з часом основні цілі стартапу можуть змінюватись на основі побажань користувачів.
8. **Оцінка та план подальшого розвитку.** Постійна оцінка результатів та аналіз ринкових конкурентів. Розробка плану подальшого розвитку.

4.2 Формування ідеї стартап проекту

Стартап полягає в побудові гнучкої системи попередньої обробки даних, яка буде базуватись на гібридних моделях штучного інтелекту з використанням кластеризаційних підходів. У таблиці 4.1 представлена основна інформація про стартап.

Таблиця 4.1 – Основна інформація про стартап

Пропонована назва проекту	NeuralClust
Засновник (автор) проекту	Симонов Єгор Денисович
Опис ідеї стартапу	Виведення на ринок гнучкої універсальної системи для попередньої обробки та кластеризації даних з можливістю швидкої інтеграції з сервісами штучного інтелекту.
Термін реалізації стартапу	24 місяці
Необхідні ресурси	Надійне та потужне хмарне сховище великих обсягів даних для зберігання тренувальних вибірок та постійного навчання моделей. Стабільний швидкісний інтернет, доступ до електромережі. Зашифроване програмне забезпечення для розробки та спілкування співробітників. Фінансування на термін реалізації стартапу для оплати рахунків за використання ресурсів та заробітної плати співробітникам. Сертифікація системи.
Опис проблематики, яку вирішує стартап	Вирішення проблеми обробки BigData при навчанні моделей штучного інтелекту.

Продовження таблиці 4.1.

Основні мета та завдання стартапу	Метою даного стартапу є зменшення витрат ресурсів компаній, що займаються розробкою систем з інтегрованим штучним інтелектом за допомогою запропонованого продукту.
Очікувані результати	Повноцінна, функціональна, гнучка система для обробки та кластеризації вхідних даних з можливістю швидкої інтеграції в екосистему проекту.

4.3 Оцінка технічної цінності стартапу та спроможності реалізації

Проведемо детальний розбір описаної ідеї стартапу. У таблиці 4.2 наведений детальний опис.

Таблиця 4.2 – Розбір ідеї стартапу

Ідея стартапу	Можливості системи	Можливості практичного застосування
Виведення на ринок системи для попередньої обробки та кластеризації даних з можливістю швидкої інтеграції з сервісами ШІ.	1. Використання системи для передпроцесингу та візуалізації даних для покращення розуміння даних і виявлення важливих залежностей.	1. Маркетинговий Аналіз: використання для маркетингового аналізу, виявлення споживчих тенденцій, таргетингу та оптимізації кампаній.

Продовження таблиці 4.2.

	2. Можливість використовувати систему для класифікації та сортування великих обсягів даних за певними критеріями.	2. Медичний Аналіз Даних: для обробки та аналізу медичних даних, включаючи діагностику та класифікацію.
		3. Фінансовий Аналіз: використання для аналізу фінансових даних, прогнозування ринкових коливань та оптимізації інвестицій.
		4. Сервіси штучного інтелекту: інтеграція з існуючими сервісами штучного інтелекту для підвищення ефективності рішень.
		5. Наукові дослідження: використання для обробки та аналізу наукових даних в різних галузях, включаючи фізику, біологію, екологію тощо.
		6. Промисловий сектор: застосування в промисловості для контролю, діагностики та оптимізації процесів.
		7. Соціальні дослідження: для аналізу даних, пов'язаних з соціальними явищами та трендами.

Продовження таблиці 4.2.

		8. Електронна торгівля: використовувати для аналізу покупців та покращення рекомендацій.
		9. Енергетика: для аналізу та оптимізації виробництва та споживання енергії.
		10. Освіта та навчання: застосування для персоналізованого навчання та аналізу навчальних даних.

Проаналізуємо технологічну спроможність реалізації зазначеного стартапу. В таблиці 4.3 зазначені основні аспекти.

Таблиця 4.3 – Технічна спроможність реалізації стартапу

Необхідний ресурс	Наявність	Доступність	Вдосконалення
Методи кластеризації даних	Наявні	Загальнодоступні	Потребують
Методи інтеграції з аналітичними платформами	Наявні	Доступні за допомогою API	Потребують
Методи шифрування даних	Наявні	Закриті	Потребують
Методи зберігання та архівації даних	Наявні	Напівзакриті системи, необхідні спеціальні доступи	Не потребують
Методи попередньої обробки даних	Наявні	Унікальні для кожної системи	Потребують розробки

4.4 Аналіз конкуренції

Для початку, проведемо аналіз наявних конкурентів (таблиця 4.4.).

Таблиця 4.4 – Аналіз основних конкурентів

Техніко-економічні хар.-ки ідеї	Стартап	Потенційні конкуренти		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		IBM	AWS			
Зручність	5	3	3	-	-	+
Простота	4	2	3	-	-	+
Повнота	5	5	5	-	+	-
Сумісність	5	3	4	-	-	+
Ціна	5	1	1	-	-	+

Також визначимо тип та рівень конкуренції (таблиця 4.5).

Таблиця 4.5 – Аналіз конкуренції на ринку

Особливості конкурентного середовища		Формат прояву	Вплив на діяльність підприємства
Тип конкуренції	Горизонтальна	Багато підприємств пропонують схожі товари або послуги, конкуруючи між собою.	Вимагає розробки унікальних рішень та стратегій для виокремлення від конкурентів
За рівнем конкурентної боротьби	Висока	Конкуренти активно змагаються за ринкову частку та клієнтів.	Зміцнення маркетингових та стратегічних підходів для збереження і залучення клієнтів.

Продовження таблиці 4.6.

За галузевою ознакою	Сегментація галузі	Різні сегменти мають різні потреби та характеристики.	Необхідність розробки та адаптації продуктів для кожного сегменту ринку.
Конкуренція за видам товарів	Пряма конкуренція за подібними товарами і послугами	Конкуренти пропонують майже ідентичні продукти.	Важливість встановлення конкурентоспроможної ціноутворення та якісних переваг продуктів.
За характером конкурентних переваг	Конкуренція, заснована на ціні та інноваціях	Підприємства змагаються за найкращі цінові пропозиції та новаторські рішення.	Потреба у постійній інновації та вдосконаленні продуктів для збереження конкурентних переваг.
За інтенсивністю	Висока	Конкуренти витрачають значні зусилля на залучення та утримання клієнтів.	Необхідність зосередженості на вдосконаленні маркетингових і стратегічних підходів для виживання на ринку.

Сформуємо перелік факторів конкурентоспроможності на основі наведених даних. Результати представлено у таблиці 4.6.

Таблиця 4.6 – Формування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування
Інновації та технологічний прогрес	Здатність впроваджувати нові технології та інновації, що покращують продукт або послугу.
Якість продукту або послуги	Вища якість в порівнянні з конкурентами сприяє залученню та утриманню клієнтів.
Цінова конкурентоспроможність	Здатність пропонувати конкурентоспроможні ціни на продукт або послугу.
Бренд та репутація	Сильний бренд та добра репутація підприємства створюють довіру серед клієнтів.
Маркетинг та збут	Ефективність маркетингу та збуту впливає на здатність залучати нових клієнтів і зберігати існуючих.
Людський капітал та навчання	Кваліфікований персонал та програми навчання сприяють високій продуктивності та інноваціям.

4.5 Аналіз ринкових можливостей запуску проекту

Для початку проведемо попередній аналіз потенційного ринку збуту. Результат представлений у таблиці 4.7.

Таблиця 4.7 – Аналіз потенційного ринку збуту

Показники стану ринку	Характеристика
Кількість головних гравців, од	5
Загальний обсяг продаж, грн/ум.од	1 млрд. грн / рік
Динаміка ринку (якісна оцінка)	Зростає
Наявність обмежень для входу	Репутація продукту
Специфічні вимоги до стандартизації та сертифікації	Галузь має високі стандарти щодо захисту даних та конфіденційності. Для входу на ринок необхідно відповідати стандартам ISO 27001 та GDPR. Сертифікація є обов'язковою для багатьох клієнтів у галузі медицини, фінансів та громадських послуг.
Середня норма рентабельності в галузі (або по ринку), %	20–25%

Наведемо характеристику потенційних клієнтів (таблиця 4.8).

Таблиця 4.8 – Характеристика потенційних клієнтів

Потреби ринку	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Підприємства шукають інструменти, які допомагали б їм з ефективністю обробляти великі обсяги даних та впроваджувати рішення на основі штучного інтелекту.	Підприємства та організації з різних галузей, такі як фінанси, медицина, маркетинг, та інші. Також, стартап може бути корисним для дослідників та аналітиків, які працюють з даними.	Різні групи клієнтів можуть мати різні потреби та вимоги. Наприклад, фінансові компанії можуть бути більше зацікавлені в аналізі ризиків та прогнозуванні фінансових ринків, тоді як медичні установи можуть шукати рішення для обробки медичних даних та діагностики.	Висока точність, швидкість обробки даних, та забезпечення конфіденційності. Також очікується можливість інтеграції з іншими системами та підтримки різних форматів даних.

Виконаймо оцінку ризиків. Він допоможе виявити можливі перешкоди при введенні продукту в експлуатацію (таблиця 4.9).

Таблиця 4.9 – Оцінка ризиків стартапу

Ризик	Зміст	Можливе вирішення
Конкуренти	Значна конкуренція на ринку	Розвинути унікальність продукту та донести це користувачам
Технологічний	Старіння технологій	Оновлення технічної бази
Безпековий	Ненадійна система безпеки	Впровадження новітніх систем захисту інформації
Фінансовий	Обмежений бюджет	Залучення інвестицій
Репутаційний	Потенційні проблеми з репутацією	Розвиток PR стратегії
Кадровий	Обмеженість ресурсу розробників	Правильне делегування задач та найм нових співробітників

Також, проведемо оцінку можливостей стартапу (таблиця 4.10).

Таблиця 4.10 – Оцінка можливостей стартапу

Можливість	Зміст	Використання
Розширення ринку	Відкриття нових ринків збуту продукту	Вдосконалення маркетингової стратегії
Технологічний	Використання новітніх технологій	Інвестиції у дослідження та розвиток
Партнерський	Укладення партнерських угод	Пошук потенційних партнерів та взаємодія

4.6 Аналіз сильних та слабких сторін системи

Наведемо порівняльний аналіз сильних та слабких сторін запропонованої системи (таблиця 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін системи

Фактор конкурентоспроможності	Бали (1–20)	Рейтинг продуктів-конкурентів
Інновації та технологічний прогрес	16	+2
Якість продукту або послуги	18	+3
Ефективність виробництва	15	+1
Цінова конкурентоспроможність	14	0
Бренд та репутація	17	+2
Маркетинг та збут	19	+3
Розміщення та доступність	13	-2
Людський капітал та навчання	18	+2
Інновації та технологічний прогрес	16	+2

Проведемо SWOT-аналіз стартап-проекту (таблиця 4.12)

Таблиця 4.12 – SWOT-аналіз

Сильні сторони: – інноваційний продукт; – експертна команда; – гнучкість та адаптивність.	Слабкі сторони: – конкурентна боротьба; – фінансова складова.
Можливості: – ріст попиту на аналітику даних; – співпраця з іншими компаніями.	Загрози: – конкуренція; – регулювання та приватність даних; – укономічні коливання.

4.7 Аналіз цільової аудиторії

Для подальшої побудови ринкової стратегії стартап-проекту, необхідно проаналізувати цільову аудиторію проекту. Аналіз наведено в таблиці 4.13.

Таблиця 4.13 – Аналіз цільової групи користувачів

Профіль цільової групи	Готовність прийняти продукт	Орієнтований попит	Інтенсивність конкуренції в сегменті	Простоста входу
Малий бізнес	Висока	Високий	Висока	Середня
Великі компанії	Середня	Високий	Висока	Складна
Дослідницькі лабораторії	Висока	Високий	Середня	Легка
Стартапи	Висока	Середній	Середня	Середня

4.8 Розробка ринкової стратегії стартап-проекту

Сформулюємо базову стратегію розвитку продукту (таблиця 4.14).

Таблиця 4.14 – Базова стратегія розвитку продукту

Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
Маркетинкова сегментація та концентрація на ринку	Інноваційність та надійність	Диференціація продукту

Сформулюємо базову стратегію конкурентної поведінки (таблиця 4.15).

Таблиця 4.15 – Базова стратегія конкурентної поведінки

Чи є проект «першопрохідцем»?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів	Чи буде компанія копіювати основні характеристики товару конкурента?	Стратегія конкурентної поведінки
Ні	Так	Ні	Виклику лідера

Визначимо стратегію позиціонування для стартапу (таблиця 4.16).

Таблиця 4.16 – Позиціонування стартапу

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
Легка інтегрованість, якісність, захищеність, надійність	Диференціація продукту	Унікальні алгоритми кластеризації та обробки даних. Швидка інтеграція з існуючими системами штучного інтелекту. Скорочення часу та зусиль для обробки великих обсягів даних.	Інноваційні технології в обробці даних. Ефективна інтеграція з існуючими сервісами. Зменшення трудомісткості та витрат часу на обробку даних.

4.9 Розробка маркетингової програми стартап-проекту

Сформулюємо ключові переваги побудованої концепції потенційного продукту. Результати аналізу представлені в таблиці 4.17.

Таблиця 4.17 – Ключові переваги побудованої концепції продукту

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
Ефективна обробка та аналіз даних	Зниження трудомісткості та витрат часу на обробку та аналіз великих обсягів даних.	Унікальні алгоритми кластеризації та обробки даних
Підтримка великих обсягів даних	Збільшення продуктивності та зменшення витрат в компаніях, що працюють з великими обсягами даних.	Інноваційні технології в обробці даних.
Забезпечення безпеки даних та конфіденційності	Гарантує захист важливої інформації та особистих даних від несанкціонованого доступу.	Розроблені алгоритми шифрування та забезпечення конфіденційності.
Автоматизація процесів аналізу даних	Зменшення людської помилки та підвищення продуктивності завдяки автоматизованому аналізу даних.	Висока швидкість та точність аналізу. Можливість програмного регулювання обробки даних.
Інтеграція зі сторонніми системами	Покращення співпраці та інтеграції з існуючими системами та сервісами.	Підтримка різних форматів та протоколів інтеграції.

Сформулюємо концепцію маркетингових комунікацій (таблиця 4.18).

Таблиця 4.18 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламної повідомлення	Концепція рекламного звернення
Цільова аудиторія активно використовує соціальні мережі та онлайн-платформи. Вони реагують на рекламу з відгуками та відгуками в мережі.	Соціальні мережі (Facebook, Instagram, LinkedIn), Google Ads, електронна пошта, веб-сайт.	Продукт високою надійністю, рівнем безпеки та зручністю використання.	Забезпечити інформування про високий рівень безпеки, інноваційність та зручність використання.	Використовувати сучасний та ефективний підхід до реклами з акцентом на надійності та інноваціях.
Цільова аудиторія активно читає та дописується до професійних журналів, вони відвідують конференції та виставки у відповідній галузі.	Спеціалізовані журнали, конференції, виставки, експертні публікації.	Високий рівень експертизи та унікальність послуги в порівнянні з конкурентами.	Представити стартап як лідера галузі, що пропонує найбільш передові рішення.	Використовувати фахову лексику та звертатися до цільової аудиторії як до фахівців у галузі.

Продовження таблиці 4.19.

Цільова аудиторія шукає рішення, яке спрощує їхні процеси та збільшує продуктивність. Вони шукають відгуки та рекомендації від колег та експертів.	Портал професійних рецензій, форуми та блоги відомих експертів.	Зменшення витрат та підвищення продуктивності завдяки розробці.	Переконати цільових клієнтів у великому покращенні їхніх процесів та розкрити можливості продукту.	Підкреслити користь продукту для продуктивності та вигідності процесів.

4.10 Висновки до розділу 4

В даному розділі розглядалась побудова повноцінного стартап проекту. В рамках побудови проекту було проведено аналіз на рівні його стратегічного розвитку і ринкового позиціонування. Було проведено наступні етапи:

- аналіз потреб ринку: проведено дослідження ринку, визначено потреби цільової аудиторії та її поведінку;
- конкурентний аналіз: здійснено аналіз конкурентів, їхніх слабких та сильних сторін, що дало можливість визначити можливі конкурентні переваги власного стартапу;
- аналіз конкурентоспроможності: розроблено таблицю факторів конкурентоспроможності, що допомогло виокремити ключові конкурентні переваги;
- SWOT-аналіз: визначено сильні та слабкі сторони стартапу, а також можливості та загрози, що впливають на його розвиток;

– маркетингові комунікації: розроблено концепцію маркетингових комунікацій для досягнення цілей в обраному сегменті цільової аудиторії.

ВИСНОВКИ

Проведене наукове дослідження націлене на аналіз впливу наявності процесу кластеризації на ефективність моделей штучного інтелекту. У рамках даного експерименту було досліджено актуальність проблеми обробки текстових даних в рамках взаємодії з ними в системах з використанням алгоритмів штучного інтелекту. Також було висвітлено основні аспекти кластеризації та актуальних методів групування даних, відмінності наведених методів та можливі варіанти використання для різних задач попередньої обробки інформації.

В рамках цієї магістерської дисертації було розроблено інтелектуальну систему, в форматі Telegram-чатботу, для зручного взаємозв'язку та аналізу отриманих результатів. Дослідження ставило перед собою дві ключові проблеми: по-перше, вивчення впливу вибору методу кластеризації на якість штучного інтелекту шляхом порівняння результатів моделей нейронної мережі з та без використання кластеризації. По-друге, визначення впливу кількості сформованих кластерів на якість штучного інтелекту, включаючи динаміку показників у залежності від кількості кластерів для різних моделей.

Серед запропонованих методів кластеризації в рамках даного дослідження були розглянуті такі як KMeans, Agglomerative Clustering, Gaussian Mixture та Latent Dirichlet Allocation Model. В аналізі отриманих результатів були розкриті ключові висновки, зокрема підтвердження гіпотези щодо наявності впливу процесу кластеризації на ефективність моделей та підтвердження важливості кількості створених кластерів для досягнення високої якості моделей штучного інтелекту.

Також було розглянуто можливі шляхи використання отриманих результатів та вдосконалення проекту для побудови стартапу, де було висвітлено ключові аспекти, такі як: аналіз потреб ринку, конкурентний аналіз

та аналіз конкурентоспроможності, SWOT-аналіз та аналіз маркетингових комунікацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Russell S., Norvig P. Artificial Intelligence: A Modern Approach 4th Edition, Pearson: 2020. P. 3.
2. Nils J. Nilsson The History of Artificial Intelligence, Springer: 2009. P. 20.
3. 101 Artificial Intelligence Statistics [Updated for 2023] URL: https://techjury.net/blog/ai-statistics/
4. Forrester’s US 2022 Customer Experience Index: Nearly 20% Of Brands See Drop In Customer Experience Quality, URL: https://www.forrester.com/press-newsroom/forrester-us-2022-customer-experience-index/
5. Enterprise Conversational AI Platforms Reviews and Ratings, URL: https://www.gartner.com/reviews/market/enterprise-conversational-ai-platforms
6. Симонов Є.Д., Макаренко О.С., Бідюк П.І. Система аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту, ІІ науково-практична конференція «Системні науки та інформатика», КПІ ім. Ігоря Сікорського, Київ, 4-8 грудня, 2023. С. 207-212.
7. Amazon Personalize, URL: https://aws.amazon.com/personalize/
8. Threat Research, URL: https://blogs.cisco.com/security/talos
9. Згуровський М. З., Панкратова Н. Д., Основи системного аналізу: підруч. [для студ. вищ. навч. закл.]: Вид. група ВНУ, 2007. 544 с.
10. Mayo clinic research core facilities, URL: https://www.mayo.edu/research/core-facilities/services/data-analytics
11. Russell M. Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites, Sebastopol, CA : O'Reilly Media, 2011.

12. Bird S., Klein E., Lope E., Natural Language Processing with Python., Sebastopol, CA : O'Reilly Media, 2009.
13. Pinker S. The Language Instinct: How The Mind Creates Language Reprint edition., HarperCollins, 2010
14. Нейролінгвістичне програмування, URL: https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BB%D1%96%D0%BD%D0%B3%D0%B2%D1%96%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
15. Oswald C. Natural language processing and machine learning for developers, Mercury Learning and Information, 2021
16. Nltk.stem.isri, URL: https://www.nltk.org/_modules/nltk/stem/isri.html
17. SnowballStemmer, URL: https://www.nltk.org/api/nltk.stem.SnowballStemmer.html?highlight=stopwords
18. Стеммер Портера, URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BC%D0%BC%D0%B5%D1%80_%D0%9F%D0%BE%D1%80%D1%82%D0%B5%D1%80%D0%B0
19. TruncatedSVD, URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
20. LabelEncoder, URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
21. Дубініна С.В., Бідюк П.І. Застосування методів інтелектуального аналізу даних до розв’язання задач актуарного моделювання та оцінювання фінансових ризиків. Системні дослідження та інформаційні технології, 2017, № 1, С. 49-64.

22. Adams R.P. K-Means Clustering and Related Algorithms, COS 324 Elements of Machine Learning Princeton University, 2018
23. Müller A.C. Introduction to Machine Learning with Python, Sebastopol, CA : O'Reilly Media, 2016.
24. Bishop C. Pattern Recognition and Machine Learning, New York: Springer New York, 2016**.**
25. AgglomerativeClustering, URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
26. Murphy K. Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
27. Бідюк П.І., Кожухівська О.А. Моделювання і короткострокове прогнозування гетероскедастичних процесів, Індуктивне моделювання складних систем: Зб. наук. пр.: МННЦ ІТС НАН та МОН України, 2012, Вип. 4, С. 48-63.
28. Gerald J. Glass, Kenneth F. Bonham-Carter. Statistical Methods in the Biological Sciences 1st Edition, Chapman and Hall/CRC, 2014
29. Кузнєцова Н.В., Бідюк П.І. Динамічне моделювання фінансових ризиків, Індуктивне моделювання складних систем: Зб. наук.: МННЦ ІТС НАН та МОН України, 2017, Вип. 9, С. 122-137.
30. Данилов В. Я., Жиров О.Л., Бідюк П.І. Оцінювання кредитних ризиків методами інтелектуального анализу даних, Системні дослідження та інформаційні технології, 2017, № 1, С. 33-48.
31. Фельдман Л.П., Петренко А.І.,Дмитрієва О.А. Чисельні методи в інформатиці (підручник), Київ, ВНУ, 2006, 480 с.
32. Unlocking the Hidden Themes of Text with Topic Modeling, URL: https://medium.com/mlearning-ai/unlocking-the-hidden-themes-of-text-with-topic-modeling-c568aabcd1c3
33. Aurélien Géron Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, Sebastopol, CA : O'Reilly Media, 2016.

34. Ian Goodfellow, Yoshua Bengio, Aaron Courville Deep Learning, MIT Press, 2016.
35. François Chollet Deep Learning with Python, Manning Publications Co., 2018.
36. Makarenko A. Multiple-Valued Neural Networks and Branching Neural Networks. In: Shi, P., Stefanovski, J., Kacprzyk, J. (eds) Complex Systems: Spanning Control and Computational Cybernetics: Foundations. Studies in Systems, Decision and Control, vol. 414. Springer, Cham, 2022, P. 457-473.
37. Makarenko A. Multiple-Valued Artificial Neural Networks. In: International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14-19 July 2019, 1-6.
38. Accuracy vs. precision vs. recall in machine learning: what's the difference? URL: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall
39. Know The Best Evaluation Metrics for Your Regression Model URL : !https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/#:~:text=Commonly%20used%20metrics%20include%20mean,characteristics%20of%20the%20regression%20problem
40. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень: навч. посіб.– К.: ННК "ІПСА" НТУУ "КПІ", 2010, 340 с.
41. The 20 newsgroups text dataset URL: https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html
42. Кузнєцова Н.В. Виявлення та оброблення невизначеностей у формі неповних даних методами інтелектуального анализу Системні дослідження та інформаційні технології, 2016, № 2, С. 104-115.

43. Згуровський М.З, Петренко А.І. Оброблення наукових даних в умовах інформаційного «буму», Систем. дослідж. та інформ. технології, 2012, № 2, С. 7-25.
44. Зайченко Ю.П., Севаее Ф., Келестин Ю.В. Сравнительный анализ эффективности нечетких нейронных сетей в задачах прогнозирования в экономике и финансовой сфере, Систем. дослідж. та інформ. технології, 2006, № 1, С. 56-70.
45. pyTelegramBotAPI URL: https://pypi.org/project/pyTelegramBotAPI/
46. Лопатін О.К., «Alert»-технології, що грунтуються на теорії динамічних систем в економічних задачах, Систем. дослідж. та інформ. технології, 2011, № 1, С. 50-56.

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО КОДУ

configs.yml

```
chat_configs:
  owner: Symonov Yehor KA-22mp
  token: 6460260796:AAFUT6dRN5Toxu3r9Z1ivlg5QgUaOKRoPAs
speech:
  alerts:
    max_clusters: Досягнуто максимальна кількість кластерів!
    min_clusters: Досягнуто мінімальна кількість кластерів!
    save_success: Успішно збережено!
  buttons:
    add_clusters: "\u2795"
    back: "<< Назад"
    save: "Зберегти"
    settings_option1: К кластерів
    settings_option2: Базова модель
    sub_clusters: "\u2796"
  messages:
    greeting_message: "Привіт \U0001F44B \nЧим я можу допомогти тобі?\n\nВи можете керувати мною, надсилаючи ці команди:\n\n\
      \ /settings - змінити налаштування боту\n\
      \ /help - відкрити довідку\n\n*Модель*\n\
      \ /statistic - вивести статистику\n\
      \ /retrain - перетренувати усі моделі"
    help_message: "Практична частина магістерської диссертації студента *Симонова Єгора КА-22мп* на тему:\n\n*Система аналізу впливу кластеризації на якість рішень в моделях штучного інтелекту*\n\n*Система вміє:*\n - Збирати загальну статистику для побудованих моделей\n - Оброблювати вхідні повідомлення у форматі *Тексту новини* для генерації цільової групи"
    settings_clusters: "Оберіть К кластерів від 10 до 50\nПоточна кількість: "
    settings_first_step: "Оберіть налаштування, які бажаєте змінити:"
    settings_model_2step: "Оберіть одну модель зі списку, якщо ви бажаєете її змінити.\n\nПоточна: "
    settings_model_3step: 'Добре! Змінити модель на: '
    wait: "Зачекайте... Триває обробка\n*Етап:* "
  models:
    model1: Agglomerative
    model2: Gaussian
    model3: KMeans
    model4: LDA
```

current_state.yml

```
chat_configs:
  basic_count_clusters: 20
  basic_model: KMeans
```

calbackQueryHandler.py

```
import telebot
import yaml
from telebot import types
from handlers.messageHandler import setup_message_handlers


def setup_callback_query_handlers(bot, configs):
    global current_state
    global n_clusters
    global cur_model
    with open('current_state.yml', 'r') as file:
        current_state = yaml.safe_load(file)
    n_clusters = current_state['chat_configs']['basic_count_clusters']
    cur_model = current_state['chat_configs']['basic_model']

    @bot.callback_query_handler(func=lambda call: call.data in ["settings_option1",
'add_clusters', 'sub_clusters'])
    def settings_option1_callback(call):
        global n_clusters
        chat_id = call.message.chat.id
        message_id=call.message.message_id

        if call.data == 'add_clusters':
            if n_clusters == 50:
                bot.answer_callback_query(call.id,
text=configs['speech']['alerts']['max_clusters'])
            else: n_clusters += 1
        elif call.data == 'sub_clusters':
            if n_clusters == 10:
                bot.answer_callback_query(call.id,
text=configs['speech']['alerts']['min_clusters'])
            else: n_clusters -= 1

        keyboard = types.InlineKeyboardMarkup()
        add_button =
types.InlineKeyboardButton(configs['speech']['buttons']['add_clusters'],
callback_data="add_clusters")
        sub_button = types.InlineKeyboardButton(configs['speech']['buttons']['sub_clusters'],
callback_data="sub_clusters")
        back_button = types.InlineKeyboardButton(configs['speech']['buttons']['back'],
callback_data="back_main_settings")
```

```
        save_button = types.InlineKeyboardButton(configs['speech']['buttons']['save'],
callback_data="save_new_n_clusters")
        keyboard.row(add_button, sub_button)
        keyboard.add(back_button, save_button)

        answer_message = configs['speech']['messages']['settings_clusters'] + "*"+
str(n_clusters) + "*"
        try:
            bot.edit_message_text(chat_id=chat_id, message_id=message_id,
text=answer_message, reply_markup=keyboard, parse_mode="Markdown")
        except: pass

    @bot.callback_query_handler(func=lambda call: call.data == "back_main_settings")
    def settings_back_to_main(call):
        global n_clusters
        chat_id = call.message.chat.id
        message_id=call.message.message_id

        keyboard = types.InlineKeyboardMarkup()
        option1 =
types.InlineKeyboardButton(configs['speech']['buttons']['settings_option1'],
callback_data="settings_option1")
        option2 =
types.InlineKeyboardButton(configs['speech']['buttons']['settings_option2'],
callback_data="settings_option2")
        keyboard.row(option1, option2)

        n_clusters = current_state['chat_configs']['basic_count_clusters']
        bot.edit_message_text(chat_id=chat_id, message_id=message_id,
text=configs['speech']['messages']['settings_first_step'], reply_markup=keyboard)

    @bot.callback_query_handler(func=lambda call: call.data == "save_new_n_clusters")
    def update_n_clusters(call):
        global current_state
        global n_clusters

        with open("current_state.yml", "w") as file:
            current_state['chat_configs']['basic_count_clusters'] = n_clusters
            yaml.dump(current_state, file, default_flow_style=False)

        bot.answer_callback_query(call.id, text=configs['speech']['alerts']['save_success'])

    @bot.callback_query_handler(func=lambda call: call.data in ["settings_option2",
"back_set_model_st2"])
    def settings_option2_callback(call):
        chat_id = call.message.chat.id
        message_id=call.message.message_id

        keyboard = types.InlineKeyboardMarkup()
        model1 = types.InlineKeyboardButton(configs['speech']['models']['model1'],
callback_data="model1")
```

```
            model2 = types.InlineKeyboardButton(configs['speech']['models']['model2'],
callback_data="model2")
            model3 = types.InlineKeyboardButton(configs['speech']['models']['model3'],
callback_data="model3")
            model4 = types.InlineKeyboardButton(configs['speech']['models']['model4'],
callback_data="model4")
            back_button = types.InlineKeyboardButton(configs['speech']['buttons']['back'],
callback_data="back_main_settings")
            keyboard.row(model1, model2)
            keyboard.row(model3, model4)
            keyboard.add(back_button)

            answer_message = configs['speech']['messages']['settings_model_2step'] + "*"+
cur_model + "*"
            bot.edit_message_text(chat_id=chat_id, message_id=message_id,
text=answer_message, reply_markup=keyboard, parse_mode="Markdown")

        @bot.callback_query_handler(func=lambda call: call.data in ['model1', 'model2',
'model3', 'model4'])
        def generic_basic_model_callback(call):
            global current_state
            global cur_model
            chat_id = call.message.chat.id
            message_id=call.message.message_id

            with open('current_state.yml', 'w') as file:
                cur_model = configs['speech']['models'][call.data]
                current_state['chat_configs']['basic_model'] = cur_model
                yaml.dump(current_state, file, default_flow_style=False)

            keyboard = types.InlineKeyboardMarkup()
            back_button = types.InlineKeyboardButton(configs['speech']['buttons']['back'],
callback_data="back_set_model_st2")
            keyboard.add(back_button)

            answer_text = configs['speech']['messages']['settings_model_3step'] + "*"+
cur_model + "*"
            bot.edit_message_text(chat_id=chat_id, message_id=message_id, text=answer_text,
reply_markup=keyboard, parse_mode="Markdown")
```

messageHandler.py

```
import telebot
from telebot import types
import urllib.parse
import pandas as pd
import numpy as np
from models.KMeans import KMeansModel
from models.AgglomerativeClustering import AgglomerativeClusteringModel
```

```
from models.GaussianMixture import GaussianMixtureModel
from models.LatentDirichletAllocation import LatentDirichletAllocationModel
from models.TextToTopic import TextToTopicModel
from models.TopicRecommend import TopicRecommendModel
import matplotlib.pyplot as plt
from io import BytesIO
from PIL import Image
import yaml


def setup_message_handlers(bot, configs, x, y, newsgroups_all, N_CLUSTERS):
    # Model 1: KMeans
    model1 = KMeansModel(x, y, n_clusters=N_CLUSTERS)
    newsgroups_all['kmeans_cluster'] = model1.predicted_labels

    # Model 2: AgglomerativeClusteringModel
    model2 = AgglomerativeClusteringModel(x, y, n_clusters=N_CLUSTERS)
    newsgroups_all['agglomeratice_cluster'] = model2.predicted_labels

    # Model 3: GaussianMixtureModel
    model3 = GaussianMixtureModel(x, y, n_clusters=N_CLUSTERS)
    newsgroups_all['gaussian_cluster'] = model3.predicted_labels

    # Model 4: LatentDirichletAllocationModel
    model4 = LatentDirichletAllocationModel(x, y, n_clusters=N_CLUSTERS)
    newsgroups_all['lda_cluster'] = model4.predicted_labels

    @bot.message_handler(commands=['start'])
    def handle_start(message):
        user_id = message.chat.id
        # Greeting message
        bot.send_message(user_id, configs['speech']['messages']['greeting_message'],
parse_mode="Markdown")


    @bot.message_handler(commands=['settings'])
    def handle_settings(message):
        user_id = message.chat.id

        keyboard = types.InlineKeyboardMarkup()
        option1 =
types.InlineKeyboardButton(configs['speech']['buttons']['settings_option1'],
callback_data="settings_option1")
        option2 =
types.InlineKeyboardButton(configs['speech']['buttons']['settings_option2'],
callback_data="settings_option2")
        keyboard.row(option1, option2)

        bot.send_message(user_id, configs['speech']['messages']['settings_first_step'],
reply_markup=keyboard)

    @bot.message_handler(commands=['help'])
```

```
        def handle_help(message):
            user_id = message.chat.id
            bot.send_message(user_id, configs['speech']['messages']['help_message'],
parse_mode="Markdown")

        @bot.message_handler(commands=['statistic'])
        def handle_statistic(message):
            user_id = message.chat.id
            print('Етап: 1 з 6')
            message1 = bot.send_message(user_id, configs['speech']['messages']['wait']+'0 з 6',
parse_mode="Markdown")
            urm = TextToTopicModel(x, newsgroups_all['target'].values)
            urm.fit_predict()
            newsgroups_all['predicted_topic'] = np.argmax(urm.predict(urm.x), axis=1)
            bot.delete_message(user_id, message1.message_id)
            message2 = bot.send_message(user_id, configs['speech']['messages']['wait']+'1 з 6',
parse_mode="Markdown")

            # TRM with NN Predicted Topic
            print('Етап: 2 з 6')
            trm_model_base = TopicRecommendModel(newsgroups_all[['age', 'education',
'marital_status', 'sex', 'income', 'predicted_topic']], 'predicted_topic')
            trm_model_base.fit_predict()
            trm_model_base_stat = pd.DataFrame(trm_model_base.get_stat(), index=['Basic']).T
            bot.delete_message(user_id, message2.message_id)
            message3 = bot.send_message(user_id, configs['speech']['messages']['wait']+'2 з 6',
parse_mode="Markdown")

            # TRM with Model 1 (KMeans)
            print('Етап: 3 з 6')
            trm_model_kmeans = TopicRecommendModel(newsgroups_all[['age', 'education',
'marital_status', 'sex', 'income', 'kmeans_cluster']], 'kmeans_cluster')
            trm_model_kmeans.fit_predict()
            trm_model_kmeans_stat = pd.DataFrame(trm_model_kmeans.get_stat(),
index=['KMeans']).T
            bot.delete_message(user_id, message3.message_id)
            message4 = bot.send_message(user_id, configs['speech']['messages']['wait']+'3 з 6',
parse_mode="Markdown")

            # TRM with Model 2 (Agglomerative)
            print('Етап: 4 з 6')
            trm_model_agglomerative = TopicRecommendModel(newsgroups_all[['age',
'education', 'marital_status', 'sex', 'income', 'agglomeratice_cluster']], 'agglomeratice_cluster')
            trm_model_agglomerative.fit_predict()
            trm_model_agglomerative_stat =
pd.DataFrame(trm_model_agglomerative.get_stat(), index=['Agglomerative']).T
            bot.delete_message(user_id, message4.message_id)
            message5 = bot.send_message(user_id, configs['speech']['messages']['wait']+'4 з 6',
parse_mode="Markdown")

            # TRM with Model 3 (Gaussian)
            print('Етап: 5 з 6')
```

```
            trm_model_gaussian = TopicRecommendModel(newsgroups_all[['age', 'education',
'marital_status', 'sex', 'income', 'gaussian_cluster']], 'gaussian_cluster')
            trm_model_gaussian.fit_predict()
            trm_model_gaussian_stat = pd.DataFrame(trm_model_gaussian.get_stat(),
index=['Gaussian']).T
            bot.delete_message(user_id, message5.message_id)
            message6 = bot.send_message(user_id, configs['speech']['messages']['wait']+'5 з 6',
parse_mode="Markdown")

            # TRM with Model 4 (LDA)
            print('Етап: 6 з 6')
            trm_model_lda = TopicRecommendModel(newsgroups_all[['age', 'education',
'marital_status', 'sex', 'income', 'lda_cluster']], 'lda_cluster')
            trm_model_lda.fit_predict()
            trm_model_lda_stat = pd.DataFrame(trm_model_lda.get_stat(), index=['LDA']).T
            bot.delete_message(user_id, message6.message_id)
            message7 = bot.send_message(user_id, configs['speech']['messages']['wait']+'6 з 6',
parse_mode="Markdown")

            trm_model_stat = pd.concat([trm_model_base_stat, trm_model_kmeans_stat,
trm_model_agglomerative_stat, trm_model_gaussian_stat, trm_model_lda_stat], axis=1)
            trm_model_stat = trm_model_stat.applymap(lambda x: f'{x:.4f}')
            trm_model_stat.reset_index(inplace=True)
            trm_model_stat = trm_model_stat.rename(columns={'index': 'Metrics'})

            fig, ax = plt.subplots()
            ax.axis('off')  # Hide the axes
            table = ax.table(cellText=trm_model_stat.values,
colLabels=trm_model_stat.columns, cellLoc='center', loc='center')
            table.auto_set_font_size(False)
            table.set_fontsize(14)  # Adjust the font size for the table headers
            table.scale(2.5, 1.5)

            # Adjust the font size for cell text
            for (i, j), cell in table._cells.items():
                if i == 0:  # i=0 represents the header row
                    cell.set_text_props(fontsize=12)  # Adjust the font size for header cells
                    cell.set_text_props(weight='bold')  # Make header cells bold
                else:
                    cell.set_text_props(fontsize=10)  # Adjust the font size for data cells
                    if j == 0:  # j=0 represents the first column (indices)
                        cell.set_text_props(weight='bold')

            image_buffer = BytesIO()
            plt.savefig(image_buffer, format='png', bbox_inches='tight', pad_inches=0)
            image_buffer.seek(0)
            image = Image.open(image_buffer)

            bot.delete_message(user_id, message7.message_id)
            bot.send_message(user_id, 'Result: ', parse_mode="Markdown")
            bot.send_photo(user_id, image)
            image_buffer.close()
```

```
            plt.close()

        @bot.message_handler(commands=['retrain'])
        def handle_retrain(message):
            user_id = message.chat.id
            bot.send_message(user_id, 'Start Retrain... Wait ', parse_mode="Markdown")

        @bot.message_handler(func=lambda message: True)
        def handle_message(message):
            user_id = message.chat.id
            with open('current_state.yml', 'r') as file:
                temp = yaml.safe_load(file)
            N_CLUSTERS = temp['chat_configs']['basic_count_clusters']
            trm_model_stat = pd.read_csv('test'+str(N_CLUSTERS)+'.csv')
            trm_model_stat = trm_model_stat.rename(columns={"Unnamed: 0": 'Metrics'})
            #trm_model_stat = trm_model_stat.drop('Basic', axis=1)
            fig, ax = plt.subplots()
            ax.axis('off')  # Hide the axes
            table = ax.table(cellText=trm_model_stat.values,
colLabels=trm_model_stat.columns, cellLoc='center', loc='center')
            table.auto_set_font_size(False)
            table.set_fontsize(14)  # Adjust the font size for the table headers
            table.scale(2.5, 1.5)

            # Adjust the font size for cell text
            for (i, j), cell in table._cells.items():
                if i == 0:  # i=0 represents the header row
                    cell.set_text_props(fontsize=12)  # Adjust the font size for header cells
                    cell.set_text_props(weight='bold')  # Make header cells bold
                else:
                    cell.set_text_props(fontsize=10)  # Adjust the font size for data cells
                    if j == 0:  # j=0 represents the first column (indices)
                        cell.set_text_props(weight='bold')

            image_buffer = BytesIO()
            plt.savefig(image_buffer, format='png', bbox_inches='tight', pad_inches=0)
            image_buffer.seek(0)
            image = Image.open(image_buffer)

            bot.send_message(user_id, 'Result: ', parse_mode="Markdown")
            bot.send_photo(user_id, image)
            image_buffer.close()
            plt.close()
            #predicted_category = model.predict([message.text])
            # Get the category name from the target index
            #category_name = newsgroups.target_names[predicted_category[0]]
            #bot.send_message(user_id, category_name)
```

AgglomerativeClustering.py

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import adjusted_rand_score, adjusted_mutual_info_score, silhouette_score, rand_score, davies_bouldin_score, homogeneity_score, completeness_score, v_measure_score
import joblib
import os
import numpy as np


class AgglomerativeClusteringModel:
    def __init__(self, x, y, n_clusters=20, reduce=False, linkage="ward", retrain=False):
        if reduce:
            svd = TruncatedSVD(n_components=300)
            self.x = svd.fit_transform(x)
        else: self.x = x.toarray()
        self.y = y
        self.n_clusters = n_clusters
        self.linkage = linkage
        self.retrain = retrain
        self.build_model()

    def build_model(self):
        self.model_weights_path = f'models/saved_models/agglomerative_{self.n_clusters}clusters.pkl'
        if os.path.exists(self.model_weights_path) and not self.retrain:
            self.model = joblib.load(self.model_weights_path)
        else:
            self.model = AgglomerativeClustering(n_clusters=self.n_clusters, linkage=self.linkage)
            self.model.fit(self.x)
        self.predicted_labels = self.model.labels_

        if not os.path.exists(self.model_weights_path) or self.retrain: joblib.dump(self.model, self.model_weights_path)

    def get_stat(self):
        return {
            'Adjusted Rand Index (ARI)': adjusted_rand_score(self.y, self.predicted_labels),
            'Adjusted Mutual Information (AMI)': adjusted_mutual_info_score(self.y, self.predicted_labels),
            'Silhouette Score': silhouette_score(self.x, self.predicted_labels),
            'Rand Score': rand_score(self.y, self.predicted_labels),
            # 'Davies Bouldin Score': davies_bouldin_score(self.x, predicted_labels),
            'Homogeneity': homogeneity_score(self.y, self.predicted_labels),
            'Completeness': completeness_score(self.y, self.predicted_labels),
            'V-measure': v_measure_score(self.y, self.predicted_labels)
        }
```

GaussianMixture.py

```
from sklearn.mixture import GaussianMixture
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import adjusted_rand_score, adjusted_mutual_info_score, silhouette_score, rand_score, davies_bouldin_score, homogeneity_score, completeness_score, v_measure_score
import joblib
import os
import numpy as np


class GaussianMixtureModel:
    def __init__(self, x, y, n_clusters=20, reduce=True, retrain=False):
        if reduce:
            svd = TruncatedSVD(n_components=300)
            self.x = svd.fit_transform(x)
        else: self.x = x.toarray()
        self.y = y
        self.n_clusters = n_clusters
        self.retrain = retrain
        self.build_model()

    def build_model(self):
        self.model_weights_path = f'models/saved_models/gaussian_{self.n_clusters}clusters.pkl'
        if os.path.exists(self.model_weights_path) and not self.retrain:
            self.model = joblib.load(self.model_weights_path)
        else:
            self.model = GaussianMixture(n_components=self.n_clusters)
            self.model.fit(self.x)
        self.predicted_labels = self.model.predict(self.x)

        if not os.path.exists(self.model_weights_path) or self.retrain: joblib.dump(self.model, self.model_weights_path)

    def get_stat(self):
        return {
            'Adjusted Rand Index (ARI)': adjusted_rand_score(self.y, self.predicted_labels),
            'Adjusted Mutual Information (AMI)': adjusted_mutual_info_score(self.y, self.predicted_labels),
            'Silhouette Score': silhouette_score(self.x, self.predicted_labels),
            'Rand Score': rand_score(self.y, self.predicted_labels),
            # 'Davies Bouldin Score': davies_bouldin_score(self.x, predicted_labels),
            'Homogeneity': homogeneity_score(self.y, self.predicted_labels),
            'Completeness': completeness_score(self.y, self.predicted_labels),
            'V-measure': v_measure_score(self.y, self.predicted_labels)
        }
```

KMeans.py

```
from sklearn.cluster import KMeans
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import adjusted_rand_score, adjusted_mutual_info_score, silhouette_score, rand_score, davies_bouldin_score, homogeneity_score, completeness_score, v_measure_score
import joblib
import os
import numpy as np


class KMeansModel:
    def __init__(self, x, y, n_clusters=20, reduce=True, init='k-means++', n_init=100, retrain=False):
        if reduce:
            svd = TruncatedSVD(n_components=300)
            self.x = svd.fit_transform(x)
        else: self.x = x.toarray()
        self.y = y
        self.n_clusters = n_clusters
        self.init = init
        self.n_init = n_init
        self.retrain = retrain
        self.build_model()

    def build_model(self):
        self.model_weights_path = f'models/saved_models/kmeans_{self.n_clusters}clusters.pkl'
        if os.path.exists(self.model_weights_path) and not self.retrain:
            self.model = joblib.load(self.model_weights_path)
        else:
            self.model = KMeans(n_clusters=self.n_clusters, random_state=0, init=self.init, n_init=self.n_init)
            self.model.fit(self.x)
        self.predicted_labels = self.model.predict(self.x)

        if not os.path.exists(self.model_weights_path) or self.retrain: joblib.dump(self.model, self.model_weights_path)

    def get_stat(self):
        return {
            'Adjusted Rand Index (ARI)': [adjusted_rand_score(self.y, self.predicted_labels)],
            'Adjusted Mutual Information (AMI)': [adjusted_mutual_info_score(self.y, self.predicted_labels)],
            'Silhouette Score': [silhouette_score(self.x, self.predicted_labels)],
            'Rand Score': [rand_score(self.y, self.predicted_labels)],
            # 'Davies Bouldin Score': [davies_bouldin_score(self.x, predicted_labels)],
            'Homogeneity': [homogeneity_score(self.y, self.predicted_labels)],
            'Completeness': [completeness_score(self.y, self.predicted_labels)],
```

```
            'V-measure': [v_measure_score(self.y, self.predicted_labels)]
        }
```

LatentDirichletAllocation.py

```
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import adjusted_rand_score, adjusted_mutual_info_score, silhouette_score, rand_score, davies_bouldin_score, homogeneity_score, completeness_score, v_measure_score
import joblib
import os
import numpy as np


class LatentDirichletAllocationModel:
    def __init__(self, x, y, n_clusters=20, reduce=False, retrain=False):
        if reduce:
            svd = TruncatedSVD(n_components=300)
            self.x = svd.fit_transform(x)
        else: self.x = x.toarray()
        self.y = y
        self.n_topics = n_clusters
        self.retrain = retrain
        self.build_model()

    def build_model(self):
        self.model_weights_path = f'models/saved_models/lda_{self.n_topics}clusters.pkl'
        if os.path.exists(self.model_weights_path) and not self.retrain:
            self.model = joblib.load(self.model_weights_path)
        else:
            self.model = LatentDirichletAllocation(n_components=self.n_topics, random_state=0)
            self.model.fit(self.x)
        document_topics = self.model.transform(self.x)
        self.predicted_labels = document_topics.argmax(axis=1)


        if not os.path.exists(self.model_weights_path) or self.retrain: joblib.dump(self.model, self.model_weights_path)

    def get_stat(self):
        return {
            'Adjusted Rand Index (ARI)': adjusted_rand_score(self.y, self.predicted_labels),
            'Adjusted Mutual Information (AMI)': adjusted_mutual_info_score(self.y, self.predicted_labels),
            'Silhouette Score': silhouette_score(self.x, self.predicted_labels),
            'Rand Score': rand_score(self.y, self.predicted_labels),
            # 'Davies Bouldin Score': davies_bouldin_score(self.x, predicted_labels),
```

```
            'Homogeneity': homogeneity_score(self.y, self.predicted_labels),
            'Completeness': completeness_score(self.y, self.predicted_labels),
            'V-measure': v_measure_score(self.y, self.predicted_labels)
        }
```

TextToTopic.py

```
from keras.callbacks import Progbar
from keras.models import Sequential
from keras.layers import Dense, MaxPooling1D, Flatten, Conv1D
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import accuracy_score, recall_score, f1_score, mean_absolute_error,
mean_squared_error, r2_score
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import tensorflow as tf
import numpy as np
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'


class TextToTopicModel:
    def __init__(self, x, y, n_clusters=20, epochs = 5, reduce=True):
        if reduce:
            svd = TruncatedSVD(n_components=300)
            self.x = svd.fit_transform(x)
        else: self.x = x.toarray()
        self.y = y
        self.n_classes = n_clusters
        self.epochs = epochs
        self.build_model()

    def build_model(self):
        self.model = Sequential()
        self.model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(self.x.shape[1], 1)))
        self.model.add(MaxPooling1D(pool_size=2))
        self.model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
        self.model.add(MaxPooling1D(pool_size=2))
        self.model.add(Flatten())
        self.model.add(Dense(64, activation='relu'))
        self.model.add(Dense(self.n_classes, activation='softmax'))
        self.model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

    def fit_predict(self):
        self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(self.x, self.y,
test_size=0.1, random_state=42)
```

```
            y_train_onehot = to_categorical(self.y_train, num_classes=self.n_classes)
            y_test_onehot = to_categorical(self.y_test, num_classes=self.n_classes)
            # Create a Progress Bar callback for Train
            progbar = Progbar(target=self.epochs)
            for epoch in range(self.epochs):  # Adjust the number of epochs as needed
                self.model.fit(self.x_train, y_train_onehot, batch_size=64, verbose=0)
                progbar.update(epoch + 1)  # Update the progress bar

            # Train the model (old variant)
            self.model.fit(self.x_train, y_train_onehot, epochs=50, batch_size=64,
validation_data=(self.x_test, y_test_onehot), callbacks=None, verbose=0)

            # Predict on the test data
            self.y_scores = self.model.predict(self.x_test)
            self.y_pred = np.argmax(self.y_scores, axis=1)  # Convert one-hot encoded
predictions to class labels


        def predict(self, x):
            return self.model.predict(x)

        def get_stat(self):
            return {
                'Accuracy Score': [accuracy_score(self.y_test, self.y_pred)],
                'Recall': [recall_score(self.y_test, self.y_pred, average='weighted')],
                'F1-Score': [f1_score(self.y_test, self.y_pred, average='weighted')],
                'Mean Absolute Error (MAE)': [mean_absolute_error(self.y_test, self.y_pred)],
                'Mean Squared Error (MSE)': [mean_squared_error(self.y_test, self.y_pred)],
                'R-squared (R2': [r2_score(self.y_test, self.y_pred)]
            }
```

TopicRecommend.py

```
from keras.callbacks import Progbar
from sklearn.metrics import accuracy_score, recall_score, f1_score, mean_absolute_error,
mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from tensorflow import keras
import numpy as np


class TopicRecommendModel:
    def __init__(self, input_data, target_column, n_classes=20, epochs=50):
        self.X = input_data.drop(columns=target_column).values
        self.y = input_data[target_column].values
        self.epochs = epochs
        self.n_classes = n_classes
        self.build_model()
```

```
        def build_model(self):
            self.model = keras.Sequential([
                keras.layers.Input(shape=(5,)),
                keras.layers.Reshape((5, 1)),  # Adding an extra dimension for Conv1D
                keras.layers.Conv1D(64, kernel_size=5, activation='relu', padding='same'),
                keras.layers.MaxPooling1D(pool_size=2),
                keras.layers.Conv1D(128, kernel_size=5, activation='relu', padding='same'),
                keras.layers.MaxPooling1D(pool_size=2),
                keras.layers.Flatten(),
                keras.layers.Dense(256, activation='relu'),
                keras.layers.Dropout(0.3),
                keras.layers.Dense(self.n_classes, activation='softmax')
            ])
            optimizer = keras.optimizers.Adam(learning_rate=0.001)
            self.model.compile(optimizer=optimizer,       loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

        def fit_predict(self):
            self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(self.X, self.y,
test_size=0.1, random_state=42)
            # Create a Progress Bar callback for Train
            progbar = Progbar(target=self.epochs)
            for epoch in range(self.epochs):  # Adjust the number of epochs as needed
                self.model.fit(self.x_train, self.y_train, batch_size=64, verbose=0)
                progbar.update(epoch + 1)  # Update the progress bar

            self.y_scores = self.model.predict(self.x_test)
            self.y_pred = np.argmax(self.y_scores, axis=1)

        def get_stat(self):
            return {
                'Accuracy Score': [accuracy_score(self.y_test, self.y_pred)],
                'Recall': [recall_score(self.y_test, self.y_pred, average='weighted')],
                'F1-Score': [f1_score(self.y_test, self.y_pred, average='weighted')],
                'Mean Absolute Error': [mean_absolute_error(self.y_test, self.y_pred)],
                'Mean Squared Error': [mean_squared_error(self.y_test, self.y_pred)],
                'R-squared': [r2_score(self.y_test, self.y_pred)]
            }
```