МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Кафедра комп’ютерних наук та інформаційних технологій

Звіт

з практичної роботи **№7**

з дисципліни

«Інтелектуальний аналіз даних»

Виконав: студент 3 курсу

групи СІНЖ 2022-У1

Павло АКИМОВ

Перевірила:

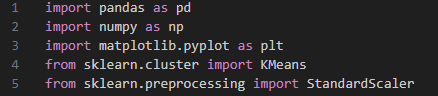
Наталія БРАТЕРСЬКА

Харків

2025

**Хід роботи**

Я розпочав свій аналіз із завантаження даних із файлу "churn-bigml-20.csv", використовуючи бібліотеку pandas. Для цього я використав команду df = pd.read\_csv(file\_path), щоб зчитати дані в датафрейм.



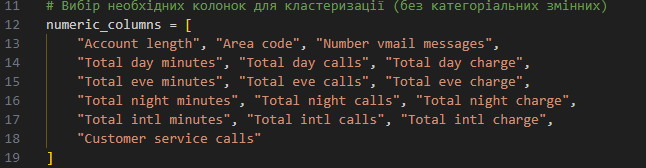
Моя мета полягала в тому, щоб провести кластеризацію даних, тому я обрав лише числові колонки, які можуть бути використані для цього завдання. Я склав список колонок, таких як "Account length", "Area code", "Number vmail messages", "Total day minutes" тощо — загалом 16 числових ознак, які я зберіг у змінній numeric\_columns.



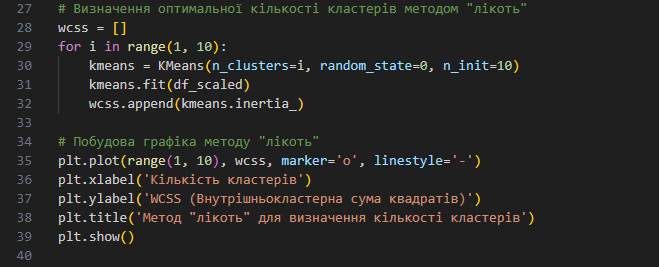
Категоріальні змінні я вирішив не включати, щоб спростити аналіз. Таким чином, я створив новий датафрейм df\_numeric, який містив лише ці числові дані.

Далі я зрозумів, що для коректної роботи алгоритму кластеризації потрібно масштабувати дані, адже різні ознаки мають різні діапазони значень. Для цього я використав StandardScaler із бібліотеки sklearn, застосувавши метод fit\_transform до df\_numeric. Результат я зберіг у змінній df\_scaled.

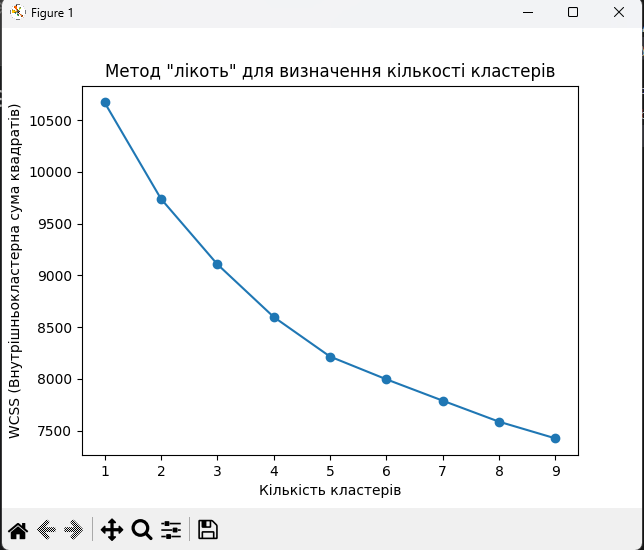
Наступним кроком було визначення оптимальної кількості кластерів для алгоритму KMeans. Я вирішив скористатися методом "лікоть", щоб зробити це обґрунтовано. Для цього я створив цикл, який протестував кількість кластерів від 1 до 9. У кожній ітерації я ініціалізував модель KMeans із відповідною кількістю кластерів, параметром random\_state=0 для відтворюваності та n\_init=10 для стабільності результатів. Після навчання моделі я обчислював внутрішньокластерну суму квадратів (WCSS) і додавав її до списку wcss.

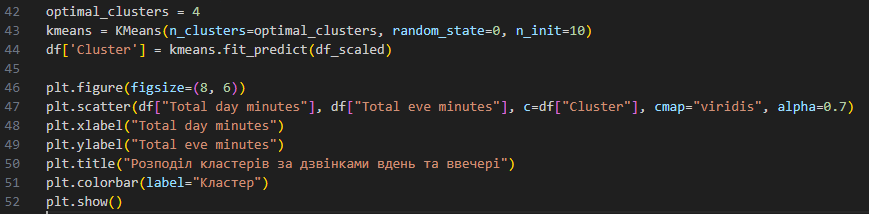


Щоб візуально оцінити результати, я побудував графік методу "лікоть". Я використав бібліотеку matplotlib: намалював залежність WCSS від кількості кластерів за допомогою plt.plot, додав маркери 'o' для точок і підписав осі — "Кількість кластерів" по осі X та "WCSS (Внутрішньокластерна сума квадратів)" по осі Y. Заголовок графіка я зробив "Метод 'лікоть' для визначення кількості кластерів". Нарешті, я викликав plt.show(), щоб відобразити графік.

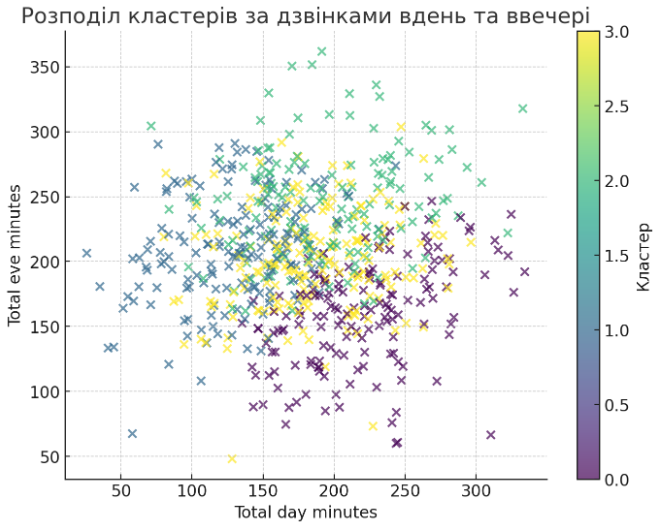


Тепер, дивлячись на графік, я зможу визначити оптимальну кількість кластерів — це буде та точка, де крива WCSS починає "згинатися" і її спад стає менш різким. Цей аналіз допоможе мені краще зрозуміти структуру даних і підготуватися до подальшої кластеризації.



Я вирішив застосувати алгоритм KMeans із 4 кластерами, адже це здалося оптимальним після аналізу методом "лікоть". Для цього я створив модель KMeans із параметрами n\_clusters=4, random\_state=0 для відтворюваності та n\_init=10 для стабільності, а потім навчив її на масштабованих даних df\_scaled за допомогою fit\_predict. Отримані номери кластерів я додав до початкового датафрейму як нову колонку df['Cluster'].

Далі я захотів візуалізувати результати, щоб зрозуміти розподіл кластерів. Я вибрав дві характеристики — "Total day minutes" і "Total eve minutes" — і побудував графік розсіювання за допомогою.



*Посилання на Google Colab*