НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

**КУРСОВА РОБОТА**

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Аналіз продажу ігор у магазинах для ліцензійної дистрибуції»

Студента 2 курсу групи ІП-01

Спеціальності: 121

«Інженерія програмного забезпечення»

Галько Міли Вячеславівни

«ПРИЙНЯВ» з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис                    Дата

Київ - 2022 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-01 Семестр 4

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Галько Міли Вячеславівни |

|  |  |
| --- | --- |
| 1.Тема роботи | Аналіз продажу ігор у магазинах для ліцензійної дистрибуції |
|  | |
|  | |

|  |  |
| --- | --- |
| 2.Строк здачі студентом закінченої роботи | 17.08.2022 |

|  |  |
| --- | --- |
| 3. Вхідні дані до роботи | методичні вказівки до курсової робота, обрані дані з сайту |
| <https://www.kaggle.com/datasets/andrewmvd/steam-reviews> | |
|  | |
|  | |
|  | |
|  | |

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
| 1.Постановка задачі |
| 2.Аналіз предметної області |
| 3.Інтелектуальний аналіз даних |

5.Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

|  |  |
| --- | --- |
|  | |
|  | |
|  | |
|  | |
| 6.Дата видачі завдання | 16.04.2022 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 16.04.2022 |  |
| 2. | Визначення зовнішніх джерел даних | 20.05.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 25.05.2022 |  |
| 4. | Обґрунтування методів інтелектуального аналізу даних | 15.06.2022 |  |
| 6. | Застосування та порівняння ефективності методів інтелектуального аналізу даних | 23.06.2022 |  |
| 7. | Підготовка пояснювальної записки | 12.07.2022 |  |
| 8. | Здача курсової роботи на перевірку | ## TODO .08.2022 |  |
| 9. | Захист курсової роботи | ## TODO |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Галько Міли Вячеславівни |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | доц. Ліхоузова Т.А |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |
| Керівник |  |  | доц. Олійник Ю.О. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"17" серпня 2022 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: TODO сторінок, TODO рисунків, TODO таблиць, TODO посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз, кластеризацію користувацьких відгуків ігор та на основі цього формує основні гідності активних продуктів. Додатково для створення більш повної картини, виконується додатковий аналіз даних про саму активність аудиторії – формування градації ігор за кількістю певних гравців у виді графічного відображення результатів чи TODO.

Мета роботи: реалізація програмного забезпечення на мові Python для отриманих підготовлених даних для їх подальшого аналізу через різні підходи класифікації.

Дана курсова робота включає в себе: підготовка набору даних, його зменшення; опис аналізу предметної області для подальшого створення програмного забезпечення інтелектуального аналізу даних; кластеризація; графічне відображення результатів аналізу.

TODO МОДЕЛЬ КЛАСИФІКАЦІЇ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, КЛАСТЕРИЗАЦІЯ, TF-IDF, K-MEANS, SPECTRAL CLUSTERING, GUASSIAN MIXTURE, SILHOUETTE SCORE, CALINSKI-HARABASZ INDEX, DAVIES-BOULDIN INDEX, ПОРІВНЯННЯ АЛГОРИТМІВ

**ЗМІСТ**

[ВСТУП 5](#_Toc110813918)

[1. TODO ПОСТАНОВКА ЗАДАЧІ 7](#_Toc110813919)

[2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8](#_Toc110813920)

[3. ВІДБІР ТА ПІДГОТОВКА ДАНИХ 10](#_Toc110813921)

[4. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ 13](#_Toc110813922)

[4.1. Обґрунтування вибору методів інтелектуального аналізу даних 13](#_Toc110813923)

[4.2. Математична модель кластеризації 14](#_Toc110813924)

[4.3. Практичне застосування моделей 17](#_Toc110813925)

[4.4. Аналіз, оцінка та вибір оптимальої моделі 22](#_Toc110813926)

[4.5. Створення рекомендацій 24](#_Toc110813927)

[ВИСНОВКИ 26](#_Toc110813928)

[ПЕРЕЛІК ПОСИЛАНЬ 27](#_Toc110813929)

[ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ 28](#_Toc110813930)

ВСТУП

Нині все більша частина людей або є активними користувачами відеоігор або вперше пробує один з продуктів. І це не секрет, що індустрія відеоігор набирає обертів через стрімкий зріст аудиторії та їх активності. Особливо, великий внесок зробив тривалий карантин під час коронавірусу та популяризація окремих ігрових продуктів через серіали, та світові кібертурніри. Дану статистику зможе помітити кожен, проглянувши дані з магазину Steam.

Steam – сервіс, що надає послуги з цифрової дистрибуції. Вже на момент 2019 року, кількість користувачів досягла 1 мільярду. Клієнти мають змогу як купляти відеоігри, так і спілкуватися, писати коментарі-відгуки на продукти сервісу, демонструвати свої досягнення в іграх. Враховуючи увесь спектр сервісу, можемо на базі цього отримати вищеперераховані дані для їх подальшого аналізу.

Загалом, ці дані можна використати для прогнозу майбутніх продуктів. В залежності від задоволеності гравців та їх витраченого часу на певну гру, можна зробити висновки про успішність теперішнього продукту та його причини. Отже, отримавши результати цього аналізу, розробники відеоігор зможуть проглянути найпопулярніші продукти для коректування створення вже своїх нових продуктів.

Основним предметом аналізу у даній курсовій роботі стануть коментарі користувачів. По перше, наша мета – зрозуміти ставлення гравців до продукту. По друге, важливою особливістю даного програмного забезпечення є розкриття реальних причин успіху продуктів зі слів у відгуках клієнтів.

Звичайно, основна мета це важливо, але додатково буде проаналізовані й інші аспекти даних. Важливо зрозуміти не тільки ставлення користувачів до продуктів, але й спиратись на обсяг аудиторії. Реальність така, що є ігри, що мають позитивний відгук, але невелику кількість користувачів. І навпаки, продукт може бути дуже популярним і водночас мати багато ненависників. Цей аспект призводить до інформаційного парадокса. Ми маємо на меті створити якісний продукт, що зможе отримати позитивні коментарі, однак це не стверджує, що у кінцевому результаті гра буде дуже популярною.

Підсумовуючи усе вищесказане, важливо проаналізувати успіх продуктів та дізнатись їх причини, усвідомлюючи дійсне значення слова «успіх». В залежності від того, чи це значить «популярність», чи «позитивний відгук», чи інше; кінцевий результат аналізу буде мати відповідний вигляд.

1. TODO ПОСТАНОВКА ЗАДАЧІ

Під час виконання курсової роботи необхідно виконати наступні завдання:

Створення сховища даних типу «сніжинка». Структура курсової роботи узгоджена з керівником. Сховище даних повинне містити щонайменше 6 таблиць вимірів та таблицю фактів. Створення ETL процесів для завантаження даних до сховища, їх отримання зі самого сховища.

Реалізувати спроектоване сховище даних з використанням PostgreSQL версії 14.1.

Створення застосунку, що отримує вибірку даних зі створеного сховища, графічно відображає отримані дані, проводить їх інтелектуальний аналіз для отримання рекомендацій за допомогою різних моделей кластеризації.

Для аналізу даних використати такі моделі кластеризації, як K-Means, Bisecting K-Мeans, Spectral Clustering, Guassian Mixture. Для кожної моделі знайти оцінку силуету, індекс Калінського-Харабаша та індекс Девіса-Боулдіна.

Аналіз отриманих результатів, порівняння різних методів кластеризації на даній вибірці, отримання найоптимальнішого методу.

Застосування найоптимальнішої моделі кластеризації для створення рекомендацій.

Використати мову програмування Python 3 для реалізації застосунку.

Курсовий проект здати до дедлайну (початок сесії) та виконати у єдиному стилі написання коду (coding style).

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Сьогодні деякі люди вже не можуть уявити своє життя без комп’ютерних ігор. Також є й ті, що ніколи не грали або спробували свої сили лише декілька разів. Заохочування потреб усіх цих категорій клієнтів є однією з найважливіших проблем спеціалістів ігрової індустрії.

Кожна компанія цієї направленості зобов’язана аналізувати реакцію гравців на випущений продукт. Оскільки саме від потоку клієнтів буде залежати прибуток компанії й тим самим її стабільність, можливість мати найманих співробітників.

Загалом, можна стверджувати те, що для створення нової успішної гри важливо проаналізувати гідності вже активно-користувацьких застосунків. А далі приступати до реалізації нової гри відтворюючи обрані переваги. Отже, для повної оцінки успішності активних комп’ютерних ігор доречно проаналізувати відгуки самих клієнтів. Онлайн магазин Steam забезпечує нас наступними даними:

* Користувацькі коментарі ігор;
* Оцінка гри користувачем;
* Оцінка коментарів іншими гравцями.

Цих даних буде достатньо для аналізу самого ставлення клієнтів. Детальна увага буде приділена саме аналізування текстів відгуків для визначення як ставлення до застосунку, так і зрозуміння, що саме мотивує користувача стабільно грати. Оцінка гри, своєю чергою, характеризує лише ставлення до продукту. А ось оцінку коментарів іншими гравцями можна більш віднести по характеристиках до відгуків. Оскільки задовільність коментаря іншим гравцем само собою означає, що він підтверджує слова іншої людини, як її ставлення, так і згоду щодо гідності гри чи її недоліків.

Загалом, вже на цьому етапі можна стверджувати, що основний аналіз припаде все ж на відгуки користувачів, а оцінки можна відповідно вважати за коефіцієнти, що будуть збільшувати вплив даних на кінцевий результат аналізу. Також оцінки можна використати для поверхневого аналізу даних і побудови допоміжних діаграм.

Підсумовуючи усе вищесказане, необхідно реалізувати програмне забезпечення дотримуючись наступних дій:

* Робота із набором даних (підготовка, зменшення);
* Поверхневий аналіз даних (інтелектуальний аналіз, графічне відображання даних);
* Кластеризація відгуків ігор;
* Відображання у графічному вигляді результатів та їх аналіз.

1. ВІДБІР ТА ПІДГОТОВКА ДАНИХ

**3.1 Загальний аналіз на коректність даних**

В якості набору даних був обраний ## dataset наступного вигляду:

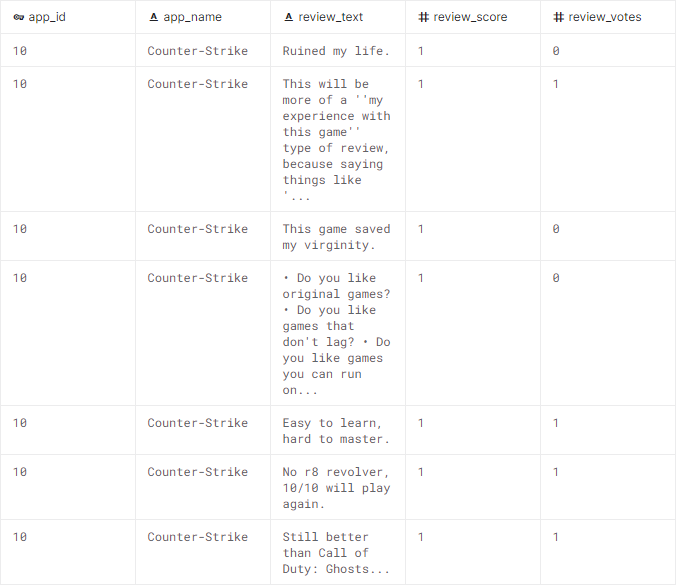


Рисунок 3.1 – Структура вхідних даних

Він налічує у собі такі поля:

* ідентифікатор гри;
* назва гри;
* відгук гравця;
* оцінка гри представлена як 1 – задовольняє, -1 – не задовольняє;
* чи був відгук гравця підтриманий іншими у вигляді 0 – якщо ні, 1 – так.

Отже, необхідно завантажити дані файлу формату .csv у dataframe та перевести поля оцінок та ідентифікатора у тип int. Далі результатом виконання функції info() є перелік загальної кількості полів - 6417106 (рис. 3.2); 5 колонок, як і передбачалося; споживання пам’яті – 171.4+ МВ.



Рисунок 3.2 – Результат виконання функції info()

Щодо «чистоти» значень полів, то маємо наступне: що деякі коментарі не мають сенсу (рис. 3.3). А результат оцінки кількості незаповнених полів демонструє, що деяка частина назв ігор та коментарів відсутня (рис. 3.4). І додатково було б краще зобразити оцінку задовільності гри у булевому значенні; тобто як 0 та 1, а не -1 та 1.



Рисунок 3.3 – Приклад некоректних даних

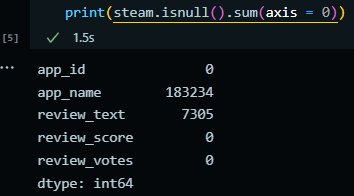


Рисунок 3.4 – Оцінка загальної кількості незаповнених даних у колонках

Отже підготовка даних може буде виконана через виконання наступних дій:

1. видалення полів, що не мають назви гри чи відгуків;
2. заміна значення -1 на 0 у полі «Оцінка гри»;
3. пропуск полів, що не мають сильного впливу (розмір відгуку менше 25 символів та непідтриманий іншими гравцями);
4. видалення полів, що мають некоректні/безглузді відгуки.

### 3.2 Видалення полів із пустими значеннями

З рисунку 3.4 можна зрозуміти, що загалом при очищенні пустот загальна кількість строчок може знизитись максимум на 190539 значень. За допомогою бібліотеки pandas маємо змогу видалити стрічки таблиці через використання функції steam.dropna(inplace = True). Таким чином ми змінюємо оригінальну таблицю зменшуючи загальний обсяг стрічок до 6226728.

При повторному виконанні функції print(steam.isnull().sum(axis = 0)) (рис.3.4) отримуємо для усіх колонок кількість пустот, що дорівнюють нулю. Отже очищення було виконано успішно.

### 3.3 Заміна значень у колонці «Оцінка гри»

За допомогою наступного коду вирішується проблема заміни значення -1 на 0 у колонці «Оцінка гри»:

counter = 0

for i in steam.index:

if steam.loc[i, "review\_score"] == -1:

steam.loc[i, "review\_score"] = 0

counter += 1

print(counter).

Зміна counter дозволяє відмірити кількість замін. На кінець ітерації значення досягло 1156686. За допомогою методу сортування по колонці «Оцінка гри» та її виведенню, стає видно, що мінімальне значення дійсно стало нулем (рис.3.5).

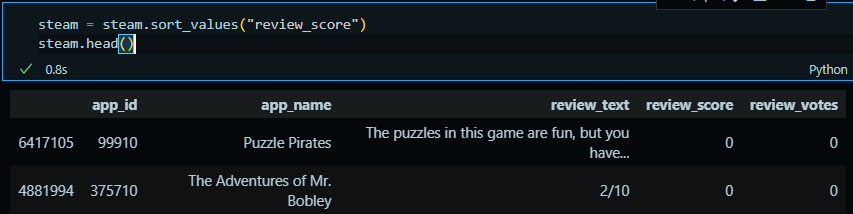


Рисунок 3.5 – Виведення таблиці відсортованою за «review\_score»

### 3.4 Видалення полів з невеликим впливом

Оскільки задана таблиця досягає великого об’єму у 6226728, то для більш оперативної аналізу слід залишити лише впливові поля. Загалом, якщо гравець написав коментар, що є меншим за 10 символів, то він не розкриває повної думки автора. Якщо ж розмір коментаря до 25, і він не оцінений іншими користувачами, можна вважати його не важливим. Отже нехай, виконаємо наступний код:

steam['letters\_count'] = steam['review\_text'].apply(lambda x: len(x)).

Додаємо нову колонку, що містить розмір коментаря користувача у символах. І тепер, можемо використати нові дані для швидкого створення нового dataframe. У ньому будуть стрічки лише ті, що мають довжину коментарів більше за 10; чи ті, що менше за 25, але мають гарну оцінку. Виконавши заданий алгоритм, отримуємо новий фрейм розміром у 5041078 записів.

### 3.5 Видалення некоректних відгуків

Даний пункт є у деякому сенсі непростим, оскільки маємо справу з ## NLP. Важливо визначити, не тільки що саме нас не задовольняє, але й виконати повну нормалізацію текстів для подальшої роботи.

1. СТРУКТУРОВАНА ОБРОБКА ПРИРОДНЬОЇ МОВИ (NLP)
2. очистити відгуки від непотрібних символів
3. переведення знаків тексту у нижній регістр
4. видалення стоп-слів
5. лематизація

ВИСНОВКИ

В результаті виконання курсової роботи було розроблено сховище даних відеоігор типу „сніжинка”, що містить одну таблицю фактів та дев’ять таблиць для представлення вимірів. Реалізовано ETL процеси для завантаження даних, їх валідації та перетворення і завантаження у сховище. Для реалізації поставленої задачі було використано PostgreSQL версії 14.1.

На основі детального опису і проведеного аналізу предметної області було проведено інтелектуальний аналізу даних для кластеризації ігор за допомогою векторизації текстових даних та використання різних моделей. Потім було порівняно ці моделі та обрану найкращу модель. Результати досліджень показали, що найкращою моделлю для цього набору даних стала модель створена за допомогою методу K-Means. Підтвердженням даного висновку є результати оцінок ефективності кластеризаці, а саме оцінки силуету, індексу Калінського-Харабаша та індексу Девіса-Боулдіна. А далі на основі найкращої моделі було створено рекомендаційну систему.

Отже, поставлені задачі були виконані, а також планується розширення функціоналу для взаємодії з сховищем даних відеоігор з використанням більшої кількості факторів, які допоможуть покращити модель та систему. А також в планах для зручності використання функціоналу розробити веб-додаток з графічним інтерфейсом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Wikipedia. K-Means clustering // Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/K-means_clustering>.
2. Wikipedia. Spectral clustering // Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/Spectral_clustering>.
3. Scikit-Learn. Clustering// Scikit-Learn, Machine Learning in Python. –– 2007-2022. –– URL: <https://scikit-learn.org/stable/modules/clustering.html>.
4. Wikipedia. Mixture model// Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/Mixture_model>.
5. Wikipedia. Expectation-maximization algorithm // Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/Expectation-maximization_algorithm>.
6. Wikipedia. Silhouette (clustering) // Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/Silhouette_(clustering)>.
7. Wikipedia. Davies-Bouldin index // Wikipedia, the free encyclopedia. –– 2022. –– URL: <https://en.wikipedia.org/wiki/Davies-Bouldin_index>.
8. Вікіпедія. TF-IDF // Вікіпедія, вільна енциклопедія. –– 2022. –– URL: <https://uk.wikipedia.org/wiki/TF-IDF>.
9. Towards Data Science. Clustering Evaluation strategies –– 2019. –– URL: <https://towardsdatascience.com/clustering-evaluation-strategies-98a4006fcfc>.
10. Towards Data Science. Visualizing Clusters with Python’s Matplotlib –– 2021. –– URL: <https://towardsdatascience.com/visualizing-clusters-with-pythons-matplolib-35ae03d87489>.
11. Towards Data Science. Build Your Own Clustering Based Recommendation Engine –– 2019. –– URL: <https://towardsdatascience.com/build-your-own-clustering-based-recommendation-engine-in-15-minutes-bdddd591d394>.
12. Towards Data Science. How to Easily Cluster Textual Data in Python –– 2021. –– URL: <https://towardsdatascience.com/how-to-easily-cluster-textual-data-in-python-ab27040b07d8>.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

(Найменування програми (документа))

*Тексти програмного коду* забезпечення з створення рекомендацій та регресійному аналізу даних про відеоігри

(Вид носія даних)

*SSD*

*студента групи ІП-01 ІІ курсу*

*Капшук Марії*

(Обсяг програми (документа), арк., Мб)

*17 арк, 16.6 Мб*

-- Dates

WITH TempDates(Day, Month, Year, DayOfWeek, DayOfYear)

AS (

SELECT DISTINCT extract(day from release\_date), extract(month from release\_date),

extract(year from release\_date), extract(dow from release\_date),

extract(doy from release\_date)

FROM stage\_games

UNION

SELECT DISTINCT NULL::numeric, NULL::numeric, year\_of\_release,

NULL::numeric, NULL::numeric

FROM stage\_sales

UNION

SELECT DISTINCT NULL::numeric, NULL::numeric, salesyear,

NULL::numeric, NULL::numeric

FROM stage\_sales

)

INSERT INTO Dates(Day, Month, Year, DayOfWeek, DayOfYear)

SELECT DISTINCT Day, Month, Year, DayOfWeek, DayOfYear

FROM TempDates

WHERE CONCAT(Day, Month, Year) NOT IN

(SELECT CONCAT(Day, Month, Year) FROM Dates)

AND Year is not null;

-- Developers

WITH TempDevelopers(Name, City, AdministrativeDivision, Country, founding\_year)

AS (

SELECT DISTINCT name, city::varchar(20), administrativedivision::varchar(20),

country::varchar(20), founding\_year::integer

FROM stage\_developer

UNION

SELECT DISTINCT developer, null::varchar(20), null::varchar(20), null::varchar(20),

null::integer

FROM stage\_sales

where lower(developer) not in (select lower(name) from stage\_developer)

UNION

SELECT DISTINCT developer, null::varchar(20), null::varchar(20), null::varchar(20),

null::integer

FROM stage\_games

where lower(developer) not in (select lower(name) from stage\_developer)

)

INSERT INTO Developers(Name, City, AdministrativeDivision, Country, founding\_year)

SELECT DISTINCT name, city, administrativedivision, country, founding\_year::integer

FROM TempDevelopers

where lower(name) not in (select lower(name) from developers);

-- Games

WITH TempGames(Name,requiredAge, price)

AS (

SELECT DISTINCT name::varchar(100), required\_age, price

FROM stage\_games

union

SELECT DISTINCT gamename::varchar(100), null::int, null::float

FROM stage\_sales

where lower(gamename) not in (select lower(name) from stage\_games)

)

INSERT INTO Games(Name, requiredage, price)

SELECT DISTINCT Name, requiredage, price

FROM TempGames

where lower(name) not in (select lower(name) from games);

-- Genres

create or replace procedure InsertIntoGenres()

as

$$

declare

row text;

curr\_genre varchar(30);

BEGIN

for row in

select genres from stage\_games

loop

foreach curr\_genre in array string\_to\_array(row, ';') loop

if lower(curr\_genre) not in (select lower(name) from genres) then

insert into genres(name) values(curr\_genre);

end if;

end loop;

end loop;

insert into genres(Name)

select distinct genre

from stage\_sales

where lower(genre) not in (select lower(name) from genres);

end;

$$

language plpgsql;

call InsertIntoGenres();

-- Categories

create or replace procedure InsertIntoCategories()

as

$$

declare

row text;

curr\_category varchar(50);

BEGIN

for row in

select categories from stage\_games

loop

foreach curr\_category in array string\_to\_array(row, ';') loop

if lower(curr\_category) not in (select lower(name) from categories) then

insert into categories(name) values(curr\_category);

end if;

end loop;

end loop;

end;

$$

language plpgsql;

call InsertIntoCategories();

-- Regions

insert into regions(code, name) values ('NA', 'North America');

insert into regions(code, name) values ('EU', 'Europe');

insert into regions(code, name) values ('JP', 'Japan');

insert into regions(code, name) values ('OTH', 'Other');

-- Platforms

create or replace procedure InsertIntoPlatforms()

as

$$

declare

row text;

curr\_platform varchar(20);

BEGIN

for row in

select platforms from stage\_games

loop

foreach curr\_platform in array string\_to\_array(row, ';') loop

if lower(curr\_platform) not in (select lower(name) from platforms) then

insert into platforms(name) values(curr\_platform);

end if;

end loop;

end loop;

insert into platforms(code)

select distinct platform from stage\_sales

where lower(platform) not in (select lower(code) from platforms);

end

$$

language plpgsql;

call InsertIntoPlatforms();

--CategoryList

create or replace procedure InsertIntoCategoryList()

as

$$

declare

row record;

curr\_category varchar(50);

id\_category int;

id\_game int;

BEGIN

for row in

select name, categories from stage\_games

loop

foreach curr\_category in array string\_to\_array(row.categories, ';') loop

select gameid into id\_game from games where lower(name)=lower(row.name);

select categoryid into id\_category from categories

where name=curr\_category;

insert into categorieslist(categoryid, gameid) values(id\_category, id\_game);

end loop;

end loop;

end

$$

language plpgsql;

call InsertIntoCategoryList();

--GenresList

create or replace procedure InsertIntoGenresList()

as

$$

declare

row record;

curr\_genre varchar(30);

id\_genre int;

id\_game int;

BEGIN

for row in

select name, genres from stage\_games

loop

foreach curr\_genre in array string\_to\_array(row.genres, ';') loop

select gameid into id\_game from games where lower(name)=lower(row.name);

select genreid into id\_genre from genres

where name=curr\_genre;

insert into genreslist(genreid, gameid) values(id\_genre, id\_game);

end loop;

end loop;

for row in

select gamename, genre from stage\_sales

where gamename not in (select name from stage\_games)

loop

select gameid into id\_game from games where lower(name)=lower(row.gamename);

select genreid into id\_genre from genres

where name=curr\_genre;

insert into genreslist(genreid, gameid) values(id\_genre, id\_game);

end loop;

end

$$

language plpgsql;

call InsertIntoGenresList();

-- Facts

WITH TempAnalysis(GameID, DeveloperID, PlatformID, ReleaseID, DateID, RegionID,

achievements, usercount, positiveratings, negativeratings, medianplaytime,

average\_playtime, Global\_Sales)

AS (

SELECT DISTINCT GameID, DeveloperID, PlatformID, dates.dateid as ReleaseID,

d.dateid, RegionID, achievements, positive\_ratings,

(string\_to\_array(owners, '-'))[1]::int as usercount,

negative\_ratings, median\_playtime, average\_playtime, global\_sales

FROM games

join stage\_games as sgames

on lower(games.name) = lower(sgames.name)

join stage\_sales as sales

on lower(sales.gamename) = lower(sgames.name)

join developers

on developers.name = sgames.developer

join dates

on make\_date(dates.year, dates.month, dates.day)=sgames.release\_date

join platforms

on platforms.name in (select unnest(string\_to\_array(sgames.platforms, ';')))

join dates as d

on sales.salesyear=d.year and d.day is null

cross join regions

)

INSERT INTO gameanalysis(GameID, DeveloperID, PlatformID, ReleaseID, DateID, RegionID,

achievements, usercount, positiveratings, negativeratings, averageplaytime,

medianplaytime, GlobalSales)

SELECT DISTINCT GameID, DeveloperID, PlatformID, ReleaseID, DateID, RegionID,

achievements, usercount, positiveratings, negativeratings, average\_playtime,

medianplaytime, Global\_Sales

FROM TempAnalysis

where GameID not in (select gameid from gameanalysis);

-- Fill fact of sales of games that in Stage\_sales and Stage\_games

create or replace procedure InsertSales()

as

$$

declare

sales\_row record;

curs cursor for select \* from gameanalysis where regionid is not null

and achievements is not null;

BEGIN

for row in curs loop

select \* into sales\_row from stage\_sales

where lower(gamename) = (select lower(name) from games where gameid = row.gameid)

and lower(developer) = (select lower(name) from developers where developerid = row.developerid)

and year\_of\_release = (select year from dates where dateid = row.releaseid)

and salesyear = (select year from dates where dateid = row.dateid);

if row.regionid = 1 then

update gameanalysis

set regionsales = sales\_row.na\_sales

where current of curs;

elsif row.regionid = 2 then

update gameanalysis

set regionsales = sales\_row.eu\_sales

where current of curs;

elsif row.regionid = 3 then

update gameanalysis

set regionsales = sales\_row.jp\_sales

where current of curs;

elsif row.regionid = 4 then

update gameanalysis

set regionsales = sales\_row.other\_sales

where current of curs;

end if;

end loop;

end

$$

language plpgsql;

call InsertSales();

from pandas import DataFrame

from sklearn.feature\_extraction.text import TfidfVectorizer

class DatasetScaler:

    def \_\_init\_\_(self, dataset: DataFrame, columns : list = None) -> None:

        self.\_\_original\_dataset=dataset

        self.\_\_dataset=dataset.copy(True)

        self.\_\_columns = columns

        if (columns):

            self.\_\_scale()

    @property

    def scaled\_dataset(self) -> DataFrame:

        return self.\_\_dataset

    @property

    def original\_dataset(self) -> DataFrame:

        return self.\_\_original\_dataset

    def \_\_scale(self) -> None:

        vectorized\_data = self.\_\_vectorize\_data().tocoo().data

        divided\_data = self.\_\_divide\_array(vectorized\_data, len(self.\_\_columns))

        for i in range(len(self.\_\_columns)):

            self.\_\_dataset[self.\_\_columns[i]]=divided\_data[i]

    def \_\_divide\_array(self, array : list, n: int):

        divided\_arrays = [[] for \_ in range(n)]

        for i in range(0, len(array), n):

            for j in range(n):

                divided\_arrays[j].append(array[i+j])

        return divided\_arrays

    def \_\_remove\_symbols(self):

        return self.\_\_dataset[self.\_\_columns].replace([r"[^A-Za-z0-9]+"], [""], regex=True)

    def \_\_combine\_data(self, data: DataFrame):

        combined\_data = data[self.\_\_columns].apply(lambda row: \

            ' '.join(row.values.astype(str)), axis=1)

        return combined\_data

    def \_\_vectorize\_data(self):

        data = self.\_\_remove\_symbols()

        combined\_data = self.\_\_combine\_data(data)

        vectorizer = TfidfVectorizer(stop\_words='english')

        return vectorizer.fit\_transform(list(combined\_data))

from pandas import DataFrame

from sklearn.cluster import KMeans, BisectingKMeans, SpectralClustering

from sklearn.mixture import GaussianMixture

from scripts.clusters.helpers import plot\_elbow\_test, get\_clusters, print\_cluster\_evaluation,\

get\_clusters\_count, get\_sum\_of\_square\_errors, cluster\_predict, show\_clusters

from scripts.clusters.kwargs import kmeans\_kwargs, bisecting\_kmeans\_kwargs, \

spectral\_kwargs, gaussian\_mixture\_kwargs

class RecommendationSystem:

def \_\_init\_\_(

self,

dataset: DataFrame) -> None:

self.\_\_dataset = dataset

def build\_system(self, x\_cols: list):

features = self.\_\_dataset[x\_cols]

cluster\_algo = KMeans

kwargs = kmeans\_kwargs

max\_kernels = 30

sse = get\_sum\_of\_square\_errors(features, KMeans, max\_kernels, \*\*kmeans\_kwargs)

n\_clusters= get\_clusters\_count(sse)

self.\_\_model = get\_clusters(features, cluster\_algo, n\_clusters=n\_clusters, \*\*kwargs)

clusters = cluster\_predict(features, self.\_\_model)

self.\_\_dataset['Cluster\_Prediction']=list(clusters)

self.show\_clusters\_info(sse, features, clusters)

def recommend(self, element: DataFrame, dataset: DataFrame, count: int = 10) -> DataFrame:

prediction = self.\_\_dataset.loc[

self.\_\_dataset['Rank'] == int(element['Rank'])]['Cluster\_Prediction']

recommendations = self.\_\_dataset.loc[

self.\_\_dataset['Rank'] != int(element['Rank'])]

recommendations = recommendations.loc[

self.\_\_dataset['Cluster\_Prediction'] == int(prediction)]

recommendations = recommendations.sample(count)

return dataset[dataset['Rank'].isin(list(recommendations['Rank'].values))]

def show\_clusters\_info(self, sse: list, features, clusters: int):

print\_cluster\_evaluation(features, clusters)

plot\_elbow\_test(sse)

show\_clusters(features, cols = ['Platform', 'Genre', 'Publisher'],

y\_kmeans = list(self.\_\_dataset['Cluster\_Prediction']))

from pandas import DataFrame

import pandas as pd

pd.options.mode.chained\_assignment = None

import matplotlib.pyplot as plt

from kneed import KneeLocator

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics import silhouette\_score, calinski\_harabasz\_score, davies\_bouldin\_score

from scripts.config import ASSET\_PATH\_ELBOWPLOT, ASSET\_PATH\_CLUSTERSPLOT, \

ASSET\_PATH\_CLUSTERS\_VALUATION

def draw\_graph(x\_range, y, labels) -> None:

plt.figure(figsize=(10, 8))

plt.plot(x\_range, y)

plt.xticks(x\_range)

plt.xlabel(labels[0])

plt.ylabel(labels[1])

plt.grid(linestyle='--')

plt.show()

def get\_sum\_of\_square\_errors(features: DataFrame, algorithm: object, max\_kernels: int, \*\*kwargs: dict) -> list:

sse = []

for k in range(1, max\_kernels + 1):

clustering = algorithm(n\_clusters=k, \*\*kwargs).fit(features)

sse.append(clustering.inertia\_)

return sse

def get\_clusters\_count(sse: list) -> int:

"""use elbow test"""

kl = KneeLocator(range(1, len(sse) + 1), sse,

curve='convex', direction='decreasing')

return kl.elbow

def get\_clusters(features: DataFrame, algorithm: object, \*\*kwargs: dict):

return algorithm(\*\*kwargs).fit(features)

def cluster\_predict(elements, model):

return model.predict(elements)

def plot\_elbow\_test(sse : list) -> None:

plt.plot(list(range(0, len(sse))), sse)

plt.xlabel("Number of clusters")

plt.ylabel("SSE")

plt.savefig(ASSET\_PATH\_ELBOWPLOT)

def draw\_subplot(fig: plt.figure, pos: int, data: list,

colors: list, view: tuple, labels: list):

ax = fig.add\_subplot(pos, projection='3d')

ax.scatter(data[labels[0]], data[labels[1]],

data[labels[2]], c=colors, s=1)

ax.set\_xlabel(labels[0])

ax.set\_ylabel(labels[1])

ax.set\_zlabel(labels[2])

ax.view\_init(view[0], view[1])

def show\_clusters(features, cols: list, y\_kmeans):

fig = plt.figure(figsize=(16,6))

draw\_subplot(fig, 121, features, y\_kmeans,

view=(60, 30), labels = cols)

draw\_subplot(fig, 122, features, y\_kmeans,

view=(0, 60), labels = cols)

plt.savefig(ASSET\_PATH\_CLUSTERSPLOT)

def print\_cluster\_evaluation(features, clusters: int):

with open(ASSET\_PATH\_CLUSTERS\_VALUATION, 'w') as file:

file.write(f'Silhouette Score: {silhouette\_score(features, clusters)}\n') # max is better

file.write(f'Calinski-Harabasz Index: {calinski\_harabasz\_score(features, clusters)}\n') # max is better

file.write(f'Davies-Bouldin Index: {davies\_bouldin\_score(features, clusters)}\n') # min is better

Решта коду знаходиться за посиланням:

<https://github.com/Marichellka/games-data-analysis>