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 факторів формування оцінки відеоігор на основі даних SteamSpy, Metacritic, IGDB та GameFAQ»

Студента 2 курсу ІП-21 групи
Спеціальності: 121
«Інженерія програмного забезпечення»
Бойка Бориса Дмитровича

«ПРИЙНЯВ» з оцінкою

_____

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

_____
Підпис Дата

Київ - 2024 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-21 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Бойка Борис Дмитровича

1.Тема роботи Аналіз факторів формування оцінки відеоігор на основі даних SteamSpy, Metacritic, IGDB та GameFAQ

2.Строк здачі студентом закінченої роботи 19.06.2024

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту
https://www.kaggle.com/datasets/arnabchaki/popular-video-games-1980-2023
https://www.kaggle.com/datasets/juttugarakesh/video-game-data/data
https://github.com/leinstay/steamdb?tab=readme-ov-file
https://www.kaggle.com/datasets/destring/metacritic-reviewed-games-since-2000
https://www.kaggle.com/datasets/whigmalwhim/steam-releases
https://www.kaggle.com/datasets/nikdavis/steam-store-games?select=steam.csv
https://www.kaggle.com/datasets/tamber/steam-video-games

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

1.Постановка задачі
2.Аналіз предметної області
3.Розробка сховища даних
4.Інтелектуальний аналіз даних

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

6.Дата видачі завдання 16.04.2024

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.04.2024	
2.	Визначення зовнішніх джерел даних	20.04.2024	
3.	Пошук та вивчення літератури з питань курсової роботи	22.04.2024	
4.	Розробка моделі сховища даних	25.04.2024	
4.	Розробка ETL процесів	28.04.2024	
5.	Обґрунтування методів інтелектуального аналізу даних	5.05.2024	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	10.05.2024	
7.	Підготовка пояснювальної записки	25.05.2024	
8.	Здача курсової роботи на перевірку	29.05.2024	
9.	Захист курсової роботи	6.06.2024	

Студент Бойко Борис Дмитрович (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)

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 40 сторінок, 41 рисунок, 14 таблиць, 9 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: проектування та реалізація сховища даних та ETL процесів, а також реалізація програмного забезпечення для отримання даних зі сховища та їх подальшого аналізу та прогнозування.

Дана курсова робота включає в себе: опис проектування, створення та заповнення сховища даних за даною задачею за допомогою фізичної моделі бази даних, опис створення програмного забезпечення для інтелектуального аналізу даних, їх графічного відображення та прогнозування за допомогою різних моделей.

СХОВИЩЕ ДАННИХ, МОДЕЛЬ ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ФІЗИЧНА МОДЕЛЬ БАЗИ ДАНИХ, ELT ПРОЦЕСИ, МОДЕЛЬ DECISION TREE, XGBOOST, LGBM, ДИСПЕРСІЙНИЙ АНАЛІЗ, ГРАФІЧНЕ ПРЕДСТАВЛЕННЯ.

ВСТУП

Ігрова індустрія є надзвичайно конкурентним бізнесом. У якому, відгуки критиків часто визначають характер сприйняття продукту гравцями, чим безпосередньо впливають на об’єми передзамовлень та швидких продажів. Відомі випадки, коли ігри виходили в світ у недоробленому стані, однак завдяки теплим оцінкам збирали велику касу. Подальша доля таких ігор суттєво різниться. Частині з них навіть поталанило утвердитися в статусі культових.

Отже, дана робота має на меті визначити властивості притаманні визнаним іграм, важливість фігури видавця та розробника та різницю в підходах до оцінювань у критиків на різних ресурсах.

Для проведення аналізу задіяно різноманітні техніки, такі як кореляційний, дисперсійний аналіз поряд з методами машинного навчання на основі лінійних, деревоподібних алгоритмів та їх варіацій.

Математичні обрахунки та навчання моделей виконано на потужностях мови програмування Python3 [1] із застосуванням засобів зовнішнього доступу до сховища даних sqlalchemy, статистичних пакетів scipy і scikit-learn та інструментів візуалізації matplotlib і seaborn.

1.ПОСТАНОВКА ЗАДАЧІ

Під час виконання курсової роботи необхідно виконати наступні завдання: Спроектувати та побудувати сховище даних типу «сніжинка» з використанням СУБД PostgreSQL версії 16. Сховище даних повинне містити щонайменше 6 таблиць вимірів та таблицю фактів. Для автоматизації обробки даних реалізувати ETL процес, що має включати завантаження даних до області обробки, проходження етапів очистки та збагачення і заповнення

сховища, а також оновлення та додавання даних до таблиць вимірів за допомогою SCD першого та третього типів.

Створений код відповідає наступним критеріям: отримує вибірку даних зі створеного сховища, графічно відображає отримані дані, проводить їх інтелектуальний аналіз для отримання передбачення за допомогою різних моделей прогнозування.

У рамках інтелектуального аналізу необхідно провести дослідження прямих кореляційних зв'язків між числовими ознаками, перевірити тип розподілу категоріальних ознак за допомогою критерія Шапіро-Уілка з метою підбору оптимального методу дисперсійного аналізу. Результати кількох попередніх обрахунків для одиночних ознак слід підкріпити комплексним аналізом важливості.

Найкращі ознаки мають бути протестовані у якості предикторів для мультифакторної та поліноміальної регресії і регресійних адаптацій моделей Random Forest, XGBoost і LightGBM. Спираючись на значення середньоквадратичного відхилення й коефіцієнту детермінації виділити 2-3 найкращі моделі. Спробувати покращити обрані моделі шляхом пошуку за сіткою. Провести візуальне порівняння прогнозу обраних моделей.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розвиток ігровою індустрії породжує активні зміни у масовій культурі. До прикладу, популярні стримінгові сервіси Netflix, HBO, Amazon Prime охоче знімають телевізійні шоу на ігрову тематику; виробники речей стилізують свої вироби; письменники пишуть твори. Продовжуючи, можна охопити ще безліч зв'язків, які роблять ігри помітними для сучасної людини. Варто лиш зауважити, що річний обсяг випуску ігор різного масштабу вимірюється у сотнях одиниць. І лише мала частка стає відомою на широку аудиторію. Першочергово ігри потрапляють до рук критиків. Уже послуговуючись їх думкою видання публікують новини. Не секрет, що гучні новини отримують більше уваги. Відповідно так звані прохідні ігри, здебільшого не висвітлюються. Таким чином гравці не отримують захопливого досвіду, а

творців спіткає фінансові невдача. До занижених оцінок може призвести недбала маркетингова кампанія видавця, відсутність злагоди між студіями-розробниками або врешті решт неправильно сформований ігролад, що дійсно псує враження від гри.

Це дослідження може бути корисним як для людей, що бажають зрозуміти чого можна очікувати від тої чи іншої гри, так і для фахівців бізнесу, що починають свій шлях або шукають гідних партнерів.

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

− проектування сховища даних;

− створення ETL процесів для завантаження і оновлення даних; − створення вибірки даних з сховища;

− інтелектуальний аналіз даних;

− використання декількох моделей прогнозування даних; − прогнозування оцінок ігор;

− графічне відображення отриманих результатів та аналіз.

3.РОЗРОБКА СХОВИЩА ДАНИХ

3.1 Розробка моделі сховища даних

Для виконання курсової роботи було обрано 7 джерел відкритих даних на сайті https://www.kaggle.com//, що включають в себе:

− Інформація про жанри ігор:

https://www.kaggle.com/datasets/arnabchaki/popular-video-games-1980-2023

− Популярність ігор у різних регіонах:

https://www.kaggle.com/datasets/juttugarakesh/video-game-data/data −

Особливості ігроладу:

https://github.com/leinstay/steamdb?tab=readme-ov-file

− Дані агрегатора оцінок Metacritic:

https://www.kaggle.com/datasets/destring/metacritic-reviewed-games-since-2000 −

Дані ігрового майданчика Steam:

https://www.kaggle.com/datasets/whigmalwhim/steam-releases

https://www.kaggle.com/datasets/nikdavis/steam-store-games?select=steam.csv

https://www.kaggle.com/datasets/tamber/steam-video-games

На основі детального опису та проведеного аналізу предметної області було розроблено наступну модель сховища даних відеоігор за типом „сніжинка” (рисунок 3.1).

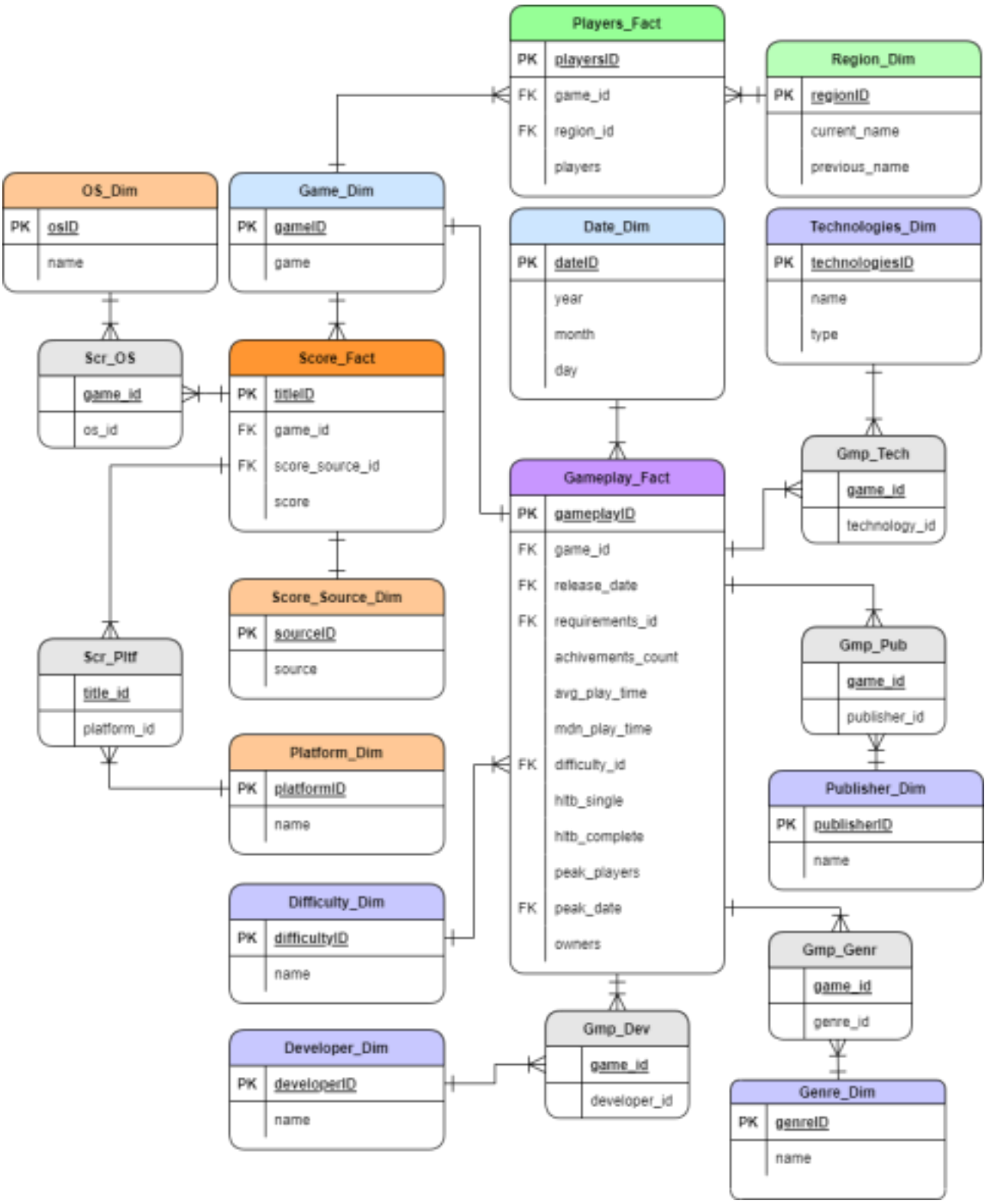

Рисунок 3.1 - Проектування моделі сховища за типом сніжинка

У моделі сховища за типом сніжинка спроектовано три таблиці фактів та одинадцять таблиць вимірів. На даній моделі видно, що головною таблицею вимірів є таблиця dim_Game, яка відповідає за самі відеоігри. Вона слугує єднальним звеном для більшості маніпуляцій над вмістом сховища, зокрема

над головною фактовою таблицею Gameplay_Fact Нижче наведено опис полів кожної таблиці сховища даних відеоігор.

Назва поля	Опис
OsID	Унікальний ідентифікатор
Name	Назва операційної системи

Таблиця 3.1.1 OS_Dim

Назва поля	Опис
SourceID	Унікальний ідентифікатор
Name	Назва ресурсу, який виставляє грі оцінку

Таблиця 3.1.2 Score_Source_Dim

Назва поля	Опис
PlatformID	Унікальний ідентифікатор
Name	Назва ігрової платформи (пк, консолі, портативного пристрою)

Таблиця 3.1.3 Platform_Dim

Назва поля	Опис
TitleID	Унікальний ідентифікатор
Game_id	Унікальний ідентифікатор гри, якій виставлено оцінку
Score_source_id	Унікальний ідентифікатор ресурсу, який виставив оцінку
Score	Оцінка гри від 0 до 100

Таблиця 3.1.4 Score_Fact

Назва поля	Опис
GameID	Унікальний ідентифікатор
Game	Назва гри

Таблиця 3.1.5 Game_Dim

Назва поля	Опис
DateID	Унікальний ідентифікатор
Year	Рік в числовому форматі
Month	Місяць в числовому форматі
Day	День в числовому форматі

Таблиця 3.1.6 Date_Dim

Назва поля	Опис
RegionID	Унікальний ідентифікатор
Current_name	Поточна назва регіону продажів
Previous_name	Попередня назва регіону продажів

Таблиця 3.1.7 Region_Dim

Назва поля	Опис
PlayersID	Унікальний ідентифікатор
Game_id	Унікальний ідентифікатор гри, яка продається в регіоні
Region_id	Унікальний ідентифікатор регіону, у якому продаються ігри
Players	Кількість гравців/продажів гри в регіоні

Таблиця 3.1.8 Players_Fact

Назва поля	Опис
TechnologyID	Унікальний ідентифікатор
Name	Назва технології
Type	Тип технології (Двигун, Набір для розробки тд)

Таблиця 3.1.9 Technology_Dim

Назва поля	Опис
DifficultyID	Унікальний ідентифікатор
Name	Назва складності гри

Таблиця 3.1.10 Difficulty_Dim

Назва поля	Опис
DeveloperID	Унікальний ідентифікатор
Name	Назва студії розробників

Таблиця 3.1.11 Developer_Dim

Назва поля	Опис
PublisherID	Унікальний ідентифікатор
Name	Юридична назва видавця гри

Таблиця 3.1.12 Publisher_Dim

Назва поля	Опис
GenreID	Унікальний ідентифікатор
Name	Назва ігрового жанру

Таблиця 3.1.13 Genre_Dim

Назва поля	Опис
GameplayID	Унікальний ідентифікатор
Release_date	Унікальний ідентифікатор дати першого видання гри
Achievements_count	Кількість досягнень, які гравець може отримати

Avg_play_time	Середній час гри
Mdn_play_time	Медіана часу гри гравця у дану гру
Difficulty_id	Унікальний ідентифікатор відносної складності гри
Hltb_single	Час потрібний на разове проходження гри
Hltb_complete	Час потрібний на виконанання 100% активностей у грі
Peak_players	Рекордна кількість одночасних гравців
Peak_date	Дата рекорду
Owners	Приблизна кількість власників гри

Таблиця 3.1.14 Gameplay_Fact

На основі спроектованої моделі було створено сховище даних відеоігор, що включає одинадцять таблиць вимірів та три таблиці фактів реалізованих за допомогою нижче наведених скриптів використовуючи PostgreSQL версії 16

```
CREATE SCHEMA IF NOT EXISTS dwh;
DO
$body$
BEGIN
```

```
-- ORANGE BLOCK
```
Створення таблиці виміру «Операційна система»
```
CREATE TABLE IF NOT EXISTS dwh.dim_os(
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
```

Створення таблиці виміру «Джерело оцінки»
```
CREATE TABLE IF NOT EXISTS dwh.dim_score_source(
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
```

Створення таблиці виміру «Ігрова платформа»
```
CREATE TABLE IF NOT EXISTS dwh.dim_platform(
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
-- LIGHT BLUE BLOCK
```
Створення таблиці виміру «Гра»
```
CREATE TABLE IF NOT EXISTS dwh.dim_game(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
```
Створення таблиці виміру «Дата»
```
CREATE TABLE IF NOT EXISTS
    dwh.dim_date( id BIGINT PRIMARY
    KEY,
    year SMALLINT,
    month SMALLINT,
    day SMALLINT
);
--GREEN BLOCK
```
Створення таблиці виміру «Регіон»
```
CREATE TABLE IF NOT EXISTS
    dwh.dim_region( id SERIAL PRIMARY
    KEY,
    current_name VARCHAR(255) UNIQUE,
    previous_name VARCHAR(255)
);
--VIOLET BLOCK
```
Створення таблиці виміру «Технологія» CREATE
```
TABLE IF NOT EXISTS dwh.dim_technology( id
BIGSERIAL PRIMARY KEY,
    name VARCHAR(255),
    type VARCHAR(255),
    CONSTRAINT type_tech_unique
UNIQUE(name,type) );
```

Створення таблиці виміру «Видавець»

```
CREATE TABLE IF NOT EXISTS
    dwh.dim_publisher( id BIGSERIAL PRIMARY
    KEY,
    name VARCHAR(255) UNIQUE
);
```

Створення таблиці виміру «Жанр»

```
CREATE TABLE IF NOT EXISTS
    dwh.dim_genre( id SERIAL PRIMARY
    KEY,
    name VARCHAR(255) UNIQUE
);
```

Створення таблиці виміру «Розробник» CREATE

```
TABLE IF NOT EXISTS dwh.dim_developer( id
BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
```

Створення таблиці виміру «Рівень складності»

```
CREATE TABLE IF NOT EXISTS
dwh.dim_difficulty( id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE
);
--FACT BLOCK
```

Створення фактової таблиці «Гравці» CREATE

```
TABLE IF NOT EXISTS dwh.fact_players(
    id BIGSERIAL PRIMARY KEY,
    game_id INT REFERENCES dwh.dim_game(id),
    region_id INT REFERENCES dwh.dim_region(id),
    players NUMERIC(7,2)
);
```

Створення фактової таблиці «Оцінка»

```
CREATE TABLE IF NOT EXISTS dwh.fact_score(
    id BIGSERIAL PRIMARY KEY,
    game_id BIGINT REFERENCES dwh.dim_game(id),
    score_source_id INT REFERENCES
    dwh.dim_score_source(id), score NUMERIC(5,2) CHECK
    (score <= 100.00)
);
```

Створення фактової таблиці «Ігролад»

```
CREATE TABLE IF NOT EXISTS dwh.fact_gameplay(
    id BIGSERIAL PRIMARY KEY,
    game_id BIGINT REFERENCES dwh.dim_game(id),
    release_date BIGINT REFERENCES
```

```
    dwh.dim_date(id), achievements_count INT,
    avg_play_time NUMERIC(10,2),
    mdn_play_time NUMERIC(10,2),
    difficulty_id INT REFERENCES
    dwh.dim_difficulty(id), hltb_single NUMERIC(10,2),
    hltb_complete NUMERIC(10,2),
    peak_players INT CHECK (peak_players >= 0),
    peak_date BIGINT REFERENCES dwh.dim_date(id),
    owners INT CHECK (owners >= 0)
);
--GRAY BLOCK
CREATE TABLE IF NOT EXISTS dwh.src_os(
    game_id BIGINT REFERENCES dwh.dim_game(id),
    os_id INT REFERENCES dwh.dim_os(id),
    CONSTRAINT src_os_pkey PRIMARY
KEY(game_id,os_id) );
CREATE TABLE IF NOT EXISTS dwh.gmp_pub(
    game_id BIGINT REFERENCES dwh.dim_game(id),
    publisher_id INT REFERENCES dwh.dim_publisher(id),
CONSTRAINT gmp_pub_pkey PRIMARY KEY(game_id,publisher_id) );
CREATE TABLE IF NOT EXISTS dwh.scr_pltf(
    title_id BIGINT REFERENCES dwh.fact_score(id),
    platform_id INT REFERENCES dwh.dim_platform(id),
CONSTRAINT scr_pltf_pkey PRIMARY KEY(title_id,platform_id) );
CREATE TABLE IF NOT EXISTS dwh.gmp_genr(
    game_id BIGINT REFERENCES dwh.dim_game(id),
    genre_id INT REFERENCES dwh.dim_genre(id),
    CONSTRAINT gmp_genr_pkey PRIMARY
KEY(game_id,genre_id) );
CREATE TABLE IF NOT EXISTS dwh.gmp_dev(
    game_id BIGINT REFERENCES dwh.dim_game(id),
    developer_id BIGINT REFERENCES dwh.dim_developer(id),
    CONSTRAINT gmp_dev_pkey PRIMARY
KEY(game_id,developer_id) );
CREATE TABLE IF NOT EXISTS dwh.gmp_tech(
    game_id BIGINT REFERENCES dwh.dim_game(id),
    technology_id BIGINT REFERENCES dwh.dim_technology(id),
    CONSTRAINT gmp_tech_pkey PRIMARY
KEY(game_id,technology_id) );
END
$body$
LANGUAGE plpgsql
```

3.2 Розробка ETL процесів

Для проектування та подальшої роботи з сховищем даних потрібно врахувати комплекс методів, які реалізовують процес переносу початкових даних в аналітичний формат, який буде підтримуватись в сховищі даних та не

порушуватиме цілісність системи. Для цього опишемо основні функції ETL процесів, що включають:

Процес завантаження.

Цей етап передбачає стягування даних довільної якості в ETL для подальшої обробки. Вхідні дані можуть бути в джерелах різних типів і форматів, бути створені в різних додатках, і, крім того, можуть використовувати різне кодування. Саме тому в подальшому для вирішення завдань аналізу дані повинні бути перетворені в єдиний універсальний формат, щоб запобігти вмісту різних факторів, які ускладнюють коректний аналіз даних.

Процес валідації даних.

На цьому етапі головною задачею є перевірка даних на коректність і повноту, після чого складається звіт про помилки для виправлення та узагальнення даних. Первинне очищення даних в процесі ETL носить переважно технічний характер, в якому основне завдання - підготувати дані до завантаження в сховище. Вторинна очистка в аналітичній системі є користувальницькою та спрямована на підготовку даних до вирішення конкретної аналітичної задачі. Завдяки чому обидва етапи очищення однаково важливі і необхідні.

Перетворення даних.

Даний етап ETL процесу потрібний для підготовки даних до розміщення в сховище даних і приведення їх до найбільш зручного виду для подальшого аналізу. При цьому повинна враховуватися вимога рівня якості даних. Тому в процесі перетворення можуть бути задіяні найрізноманітніші інструменти, які можуть бути як і найпростішими засобами ручного редагування даних так і реалізація досить складних методів обробки та очищення даних.

Завантаження даних в сховище.

Після того як дані отримано з різних джерел і виконано перетворення, агрегація і очищення даних, здійснюється останній етап ETL, тобто завантаження даних в сховище. Процес завантаження полягає в перенесенні

даних з проміжних таблиць в структури сховища даних. Саме тому, щоб пришвидшити завантаження даних потрібно заздалегідь перевірити дані на повноту та коректність, а після спроектувати та обрати найоптимальніший процес заповнення сховища. При завантаженні даних в сховище переноситься не вся інформація з джерел, а тільки та, яка була змінена протягом проміжного часу, що пройшов з попереднього завантаження.

3.3 Завантаження даних за допомогою ETL процесів

```
query = 'SELECT * FROM stage.play_time_by_player'
df = pd.read_sql_query(query, engine)

#Fix data types
toType(df,object,'game')
if 'action_type' in df.columns:
    toType(df,object,'action_type')
toType(df,float,'time')
#Dicard not
if 'action_type' in df.columns:
    df = df[df['action_type'] == 'play']
    df.drop('action_type', axis=1, inplace=True)

#Aggregate
df = df.groupby('game')['time'].mean().reset_index()

#Round time
df['time'] = df['time'].round(1)
df.to_sql('play_time_by_player', schema='stage', con=engine, if_exists='replace',
index=True, index_label='id')
query = 'SELECT * FROM stage.genres'
df = pd.read_sql_query(query, engine)

printNulls(df)
#Fix data types
toType(df,object,'game','developer','genres')

#Remove summary column
if 'summary' in df.columns:
    df.drop('summary',axis=1,inplace=True)

#Parse array into separate rows
df = df.rename(columns={'genres':'genre'})
df = parseArraish(df,'genre')

df['developer'] = df['developer'].replace(to_replace=np.nan, value="['Unknown']",
regex=True)
df = parseArraish(df,'developer')

df.dropna(subset='release_date',inplace=True)
convertDate(df,'release_date')
```

```
df = onlyEarliestRelease(df,'release_date')

df.drop('id',axis=1,inplace=True)
df = df.reset_index(drop=True)

df.to_sql('genres', schema='stage', con=engine, if_exists='replace', index=True,
index_label='id')


query = 'SELECT * FROM stage.mdn_play_time'
df = pd.read_sql_query(query, engine)

df = parseString(df, 'genres',';')
df = parseString(df, 'platforms',';')
df = parseString(df, 'developer',';')
df = parseString(df, 'publisher',';')

df[['start', 'end']] = df['owners'].str.split('-', expand=True)
df['owners'] = (df['start'].astype(int) + df['end'].astype(int)) / 2
df.drop(['start', 'end'], axis=1, inplace=True)
toType(df,int,'owners')
toType(df,float,'avg_play_time')
toType(df,float,'median_play_time')

convertDate(df,'release_date')
df = onlyEarliestRelease(df,'release_date')
df = df.rename(columns={'platforms': 'os'})

df.drop('id',axis=1,inplace=True)
df = df.rename(columns={'genres':'genre'})
df.to_sql('mdn_play_time', schema='stage', con=engine, if_exists='replace',
index=True, index_label='id')
query = 'SELECT * FROM stage.metacritic_review'
df = pd.read_sql_query(query, engine)

toType(df,float,'metascore')

df.drop('id',axis=1,inplace=True)
df.to_sql('metacritic_review', schema='stage', con=engine, if_exists='replace',
index=True, index_label='id')


query = 'SELECT * FROM stage.all_steam'
df = pd.read_sql_query(query, engine)

df = df.dropna(subset={'all_time_peak_date'})
df = df.rename(columns={'release':'release_date'})

df['technologies'] = df['technologies'].str.split('; ')
df = df.explode('technologies')
df[['technology_type','technology']] =
df['technologies'].str.split('.',expand=True)
df.drop('technologies',axis=1,inplace=True)

toType(df,float,'rating')
convertDate(df,'release_date')
df = onlyEarliestRelease(df,'release_date')
```

```
convertDate(df,'all_time_peak_date')
df = onlyEarliestRelease(df,'all_time_peak_date',True)
toType(df,int,'all_time_peak')

df.drop('id',axis=1,inplace=True)
df.to_sql('all_steam', schema='stage', con=engine, if_exists='replace', index=True,
index_label='id')


query = 'SELECT * FROM stage.regions'
df = pd.read_sql_query(query, engine)

missing_percentage = df.isna().mean() * 100

columns_to_check = ['rating', 'developer']

drop_nulls_overlim(df,columns_to_check,0.3)

df.dropna(subset='game',inplace=True)
df.dropna(subset='publisher',inplace=True)
df.dropna(subset='year_of_release',inplace=True)

toType(df,float,'na_players','jp_players','eu_players','other_players','global_play
ers')

toType(df,float,'year_of_release')
toType(df,int,'year_of_release')
convertDate(df,'year_of_release')

df.drop('id',axis=1,inplace=True)
df = df.rename(columns={'year_of_release':'release_date'})
df = onlyEarliestRelease(df,'release_date')
df.to_sql('regions', schema='stage', con=engine, if_exists='replace', index=True,
index_label='id')


query = 'SELECT * FROM stage.time_to_beat'
df = pd.read_sql_query(query, engine)

columns_to_check = ['description','tags','voiceovers','published_hltb_date']
drop_nulls_overlim(df,columns_to_check,0.3)
df.dropna(subset='gfq_difficulty',inplace=True)
df.dropna(subset='hltb_single',inplace=True)
df.dropna(subset='hltb_complete',inplace=True)
df.dropna(subset='igdb_uscore',inplace=True)
df.dropna(subset='gfq_rating',inplace=True)
columns_to_check =
['grnk_score','igdb_score','igdb_single','gfq_rating','igdb_uscore']
drop_nulls_overlim(df,columns_to_check,0.3)

df.fillna({'achivements': 0}, inplace=True)

convertToCommonScore(df,'gfq_rating',5)

if 'platforms' in df.columns:
    df = df.rename(columns={'platforms':'os'})
```

```
df = parseString(df,'os',',')
mapToCommon(df,'os',os_map)

df = parseString(df,'languages',',')
if 'languages' in df.columns:
    df = df.rename(columns={'languages':'language'})

df = parseString(df,'developers',',')
if 'developers' in df.columns:
    df = df.rename(columns={'developers':'developer'})

df = parseString(df,'publishers',',')
if 'publishers' in df.columns:
    df = df.rename(columns={'publishers':'publisher'})

toType(df,float,'hltb_single','hltb_complete','gfq_rating','igdb_uscore','achivemen
ts')
toType(df,int,'achivements')

df = df.drop_duplicates(['game', 'achivements'])

df.drop('id',axis=1,inplace=True)
df.to_sql('time_to_beat', schema='stage', con=engine, if_exists='replace',
index=True, index_label='id')
```

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Вивантаження та підготовка даних для аналізу

Усі аналітичні операції над даними здійснювалися в середовищі Jupyter Notebook. Нижче наведено пояснення процесу перенесення датасету [2].

```
db_username = os.getenv("DB_USERNAME")
db_password = os.getenv("DB_PASSWORD")
db_host = os.getenv("DB_HOST")
db_port = os.getenv("DB_PORT")
db_name = os.getenv("DB_NAME")

engine = create_engine(
    f"postgresql://{db_username}:{db_password}@{db_host}:{db_port}/{db_name}"
)
```

Рис 4.1.1 Налаштування зовнішнього доступу до сховища даних

```
query = """WITH game_with_4_ss AS (SELECT game_id
FROM dwh.fact_score
WHERE score_source_id IN (1, 2, 3, 4)
GROUP BY game_id
HAVING COUNT(DISTINCT score_source_id) = 4)
SELECT
fg.achievements_count AS achievements,
dd.day,
dd.month,
dd.year,
fg.avg_play_time,
fg.mdn_play_time,
dif.name AS difficulty,
dgen.name AS genre,
fg.hltb_single AS time_to_beat,
fg.hltb_complete AS time_to_beat_100,
fg.peak_players,
dev.name AS developer,
dp.name AS publisher,
dos.name AS os,
dt.type AS technology_type,
dt.name AS technology_name,
dss.name AS reviewer,
fs.score
FROM dwh.fact_gameplay as fg
INNER JOIN dwh.dim_game AS dg ON dg.id = fg.game_id
INNER JOIN dwh.dim_date AS dd ON dd.id = fg.release_date
INNER JOIN dwh.dim_difficulty AS dif ON dif.id = fg.difficulty_id
INNER JOIN dwh.fact_score AS fs ON fg.game_id = fs.game_id
INNER JOIN dwh.dim_score_source AS dss ON dss.id = fs.score_source_id
INNER JOIN dwh.gmp_dev AS gmpd ON gmpd.game_id = fg.game_id
INNER JOIN dwh.dim_developer AS dev ON gmpd.developer_id = dev.id
INNER JOIN dwh.gmp_pub AS gmpp ON gmpp.game_id = fg.game_id
INNER JOIN dwh.dim_publisher AS dp ON gmpp.publisher_id = dp.id
INNER JOIN dwh.gmp_genr AS gmpg ON gmpg.game_id = fg.game_id
INNER JOIN dwh.dim_genre AS dgen ON gmpg.genre_id = dgen.id
INNER JOIN dwh.src_os AS srcos ON srcos.game_id = fg.game_id
INNER JOIN dwh.dim_os AS dos ON srcos.os_id = dos.id
INNER JOIN dwh.gmp_tech AS gmpt ON gmpt.game_id = fg.game_id
INNER JOIN dwh.dim_technology AS dt ON gmpt.technology_id = dt.id
INNER JOIN game_with_4_ss ON game_with_4_ss.game_id = fg.game_id
WHERE fg.hltb_single IS NOT NULL
AND fg.owners IS NOT NULL
AND fg.avg_play_time IS NOT NULL
AND fg.mdn_play_time IS NOT NULL
AND fg.difficulty_id IS NOT NULL
AND fg.peak_players IS NOT NULL
ORDER BY dd.year DESC
"""
df = pd.read_sql_query(query,engine)
```

Рис 4.1.2 Запит на завантаження вибірки для аналізу

У парі з ознакою technology_name вибірка містить ознаку technology_type. Окремо вони не становлять жодної цінності для аналізу, тому було об’єднано воєдино [2]

```
df['technology'] = df[['technology_type','technology_name']].apply(lambda x: '/'.join(x), axis=1)
df.drop(columns={'technology_type','technology_name'}, axis=1, inplace=True)
```

Рис 4.1.3 Маніпуляції над ознакою technology

На випадок помилки на етапі очищення, дані вибірки перевірено на базову цілісність.

```
print(f'Вибірка містить {df.shape[0]} записів та {df.shape[1]-1} досліджуваних ознак')
print(f'З яких 9 є числовими, а 8 - якісними\n')

#перевірка на наявність пропусків у даних
potential_gaps = df.isnull()

print('Кількість порожніх значень за колонками:')
for col in potential_gaps.columns.values.tolist():
    print(potential_gaps[col].value_counts())
    print('')

#перевірка на дублікати
print(f'Загальна кількість неунікальних рядків: {df.duplicated().sum()}\n')

#огляд показників кореляції між цільовим значенням та числовими предикторами
corr = df._get_numeric_data().corr()
print('Показники кореляції числових даних та цільової величини:')
print(corr['score'])
```

Рис 4.1.4 Обстеження структури даних

```
Вибірка містить 250523 записів та 17 досліджуваних ознак
З яких 9 є числовими, а 8 - якісними

Загальна кількість неунікальних рядків: 0

Показники кореляції числових даних та цільової величини:
achievements        0.040419
day                 0.077891
month              -0.125799
year                0.044170
avg_play_time       0.113675
mdn_play_time       0.204944
time_to_beat        0.087012
time_to_beat_100    0.013429
peak_players        0.075396
score               1.000000
```

Рис 4.1.5 Результати обстеження вхідних

Числові ознаки в цілому дуже слабко корелюють з цільовою величною ‘score’. Для подальшої роботи буде взята трійка предикторів з найбільшими показниками. Виявлено підстави детальніше дослідити вплив 3-х показників: ‘avg_play_time’, ‘mdn_play_time’, ‘time_to_beat’

4.2 Підбір ознак в якості майбутніх предикторів

```
print(df[['avg_play_time','mdn_play_time', 'time_to_beat']].corr())
print('')
```

Рис 4.2.5 Обчислення взаємної кореляції

```
coef, p = pearsonr(df['mdn_play_time'],df['score'])
print(f'За критерієм Пірсона mdn_play_time має коефіцієнт кореляції: {coef} зі значущістю: {p}')

coef, p = pearsonr(df['time_to_beat'],df['score'])
print(f'За критерієм Пірсона time_to_beat має коефіцієнт кореляції: {coef} зі значущістю: {p}')
```

Рис 4.2.6 Перевірка значущості отриманих коефіцієнтів

Видно що предиктори avg_play_time та mdn_play_time сильно корелюють між собою, тому не має підстав використовувати обидва.

Критерій Пірсона підтвердив значущість обчислених до цього коефіцієнтів кореляції.

Згідно наведених вище міркувань, у подальшому будуть використовуватися ознаки mdn_play_time та time_to_beat.

Щоб зрозуміти основні властивості якісних ознак перед відбором, було побудовано низку візуалізацій. Смужкова діаграма має переваги над діаграмою розмаху за рахунок кращої інтерпритованості та чіткого змалювання кількості елементів у кожному класі [6]

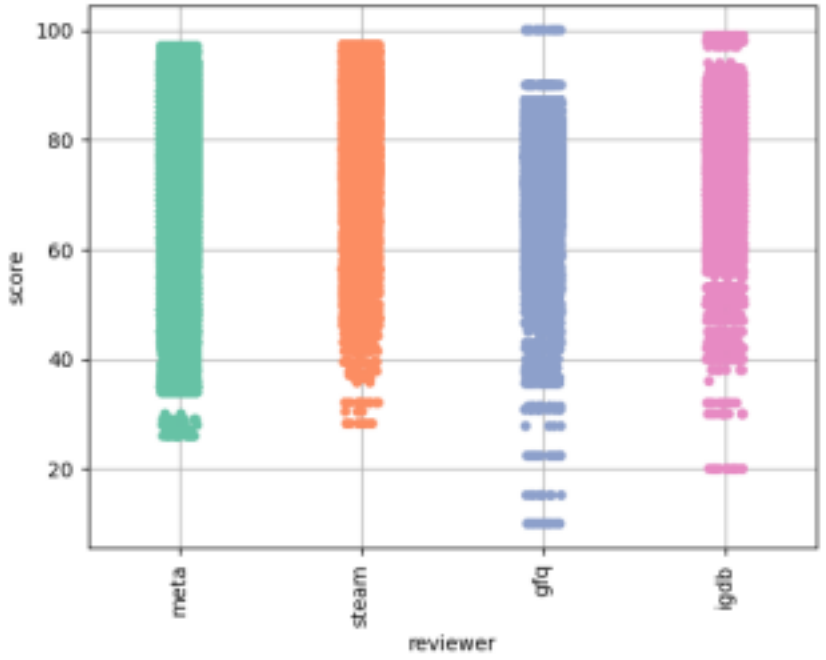

Рис 4.2.7 Смужкова діаграма для ознаки reviewer

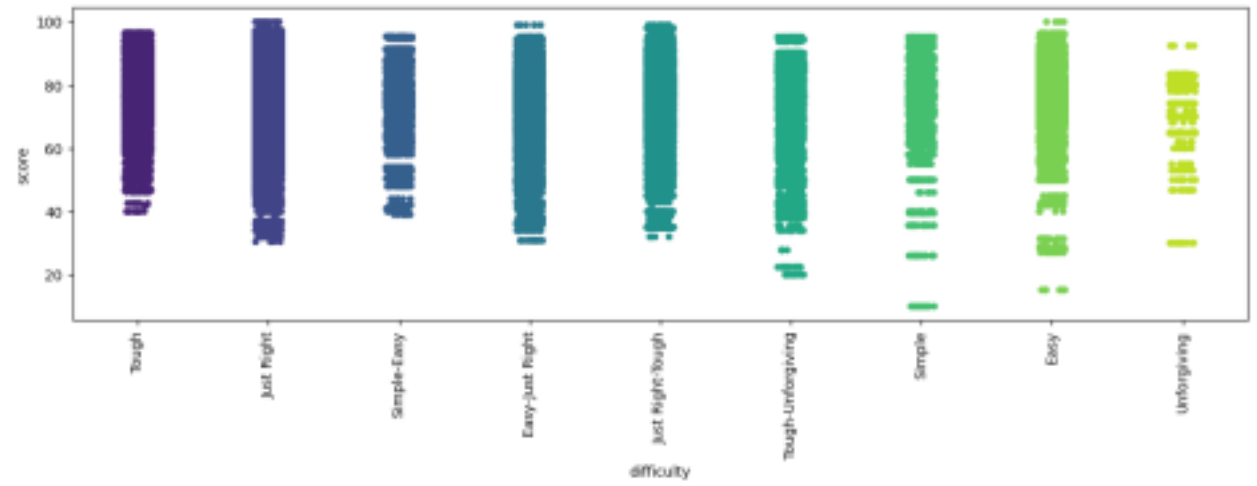

Рис 4.2.8 Смужкова діаграма для ознаки difficulty

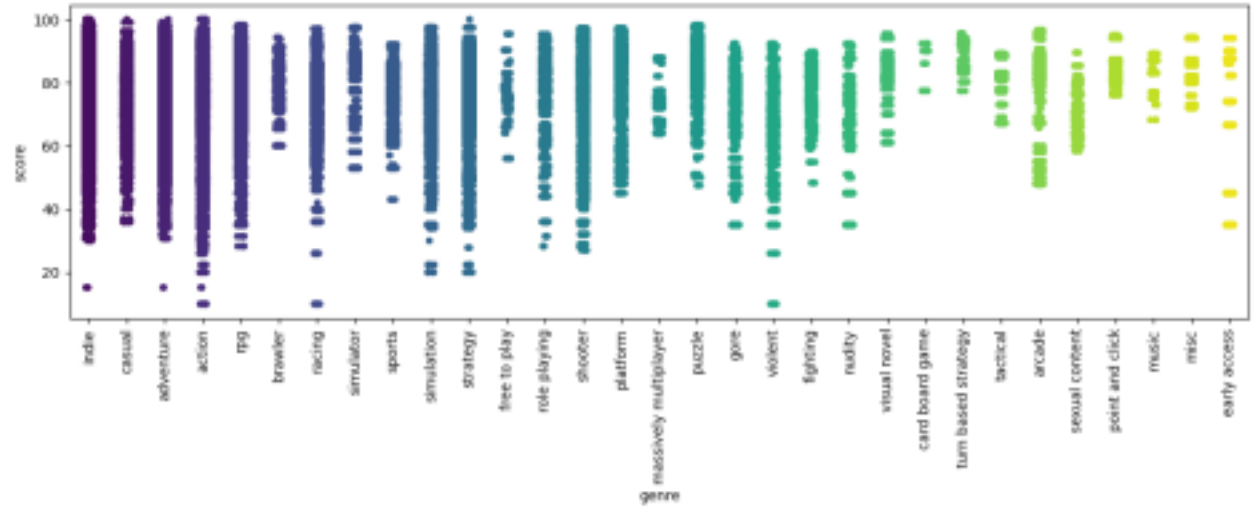

Рис 4.2.9 Смужкова діаграма для ознаки genre

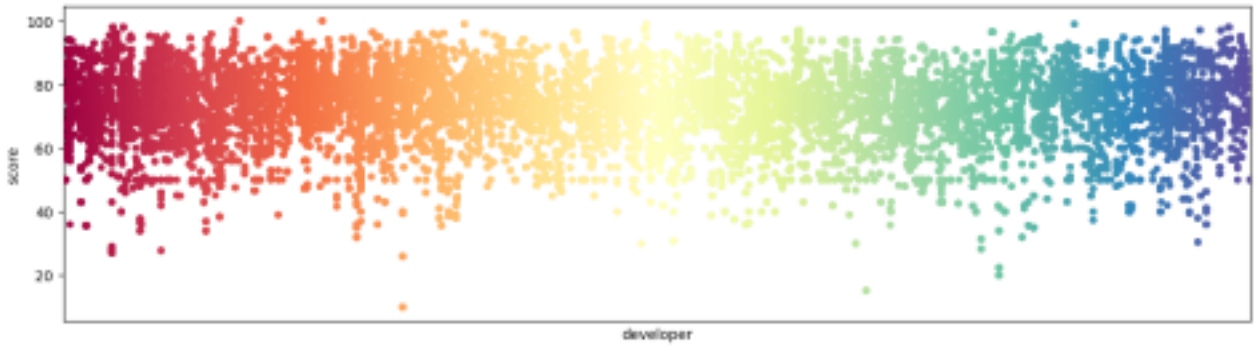

Рис 4.2.10 Смужкова діаграма для ознаки developer

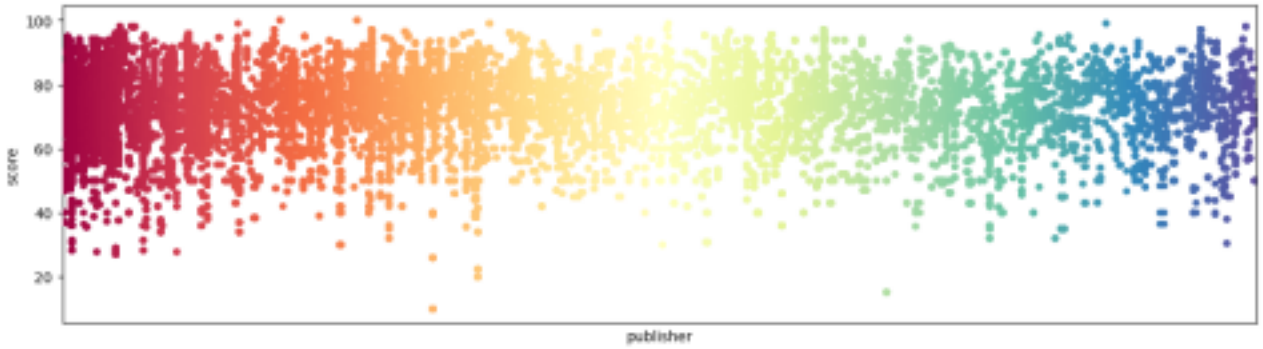

Рис 4.2.11 Смужкова діаграма для ознаки publisher

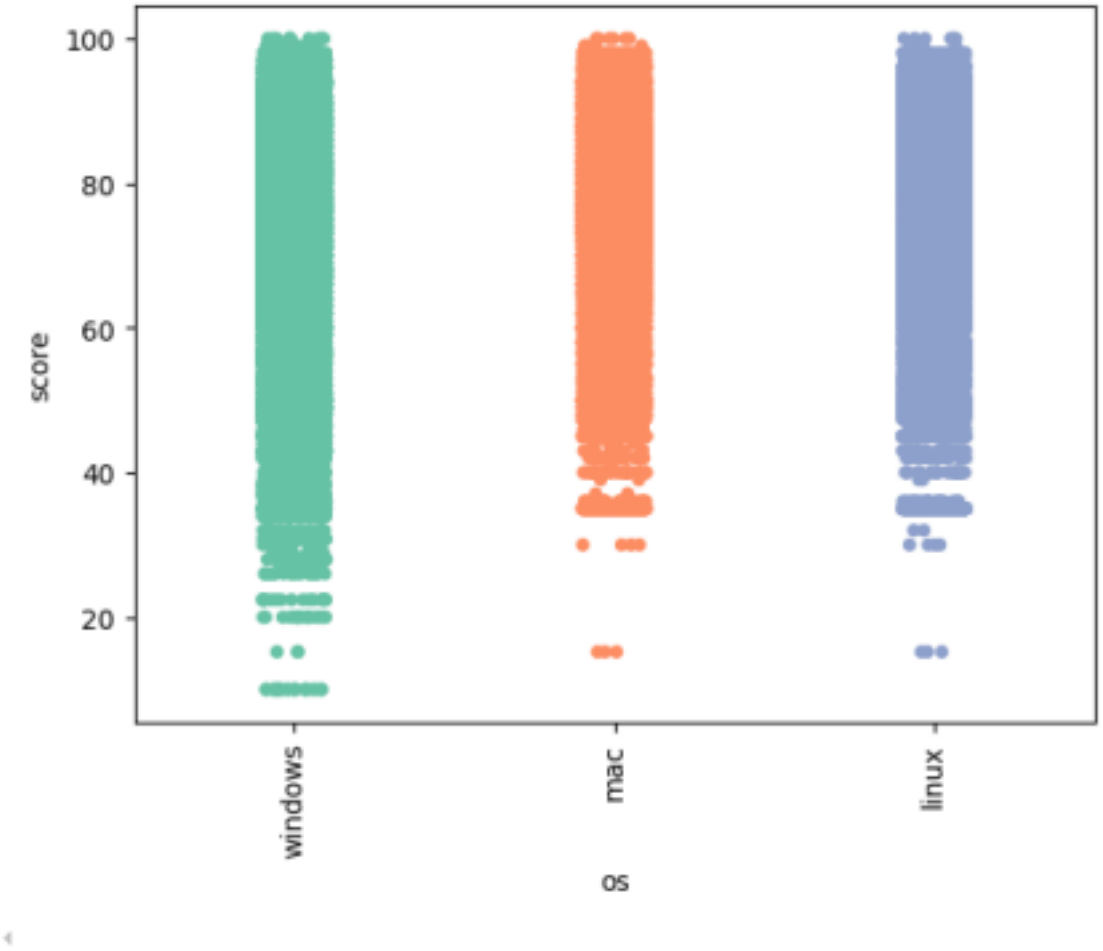

Рис 4.2.12 Смужкова діаграма для ознаки os

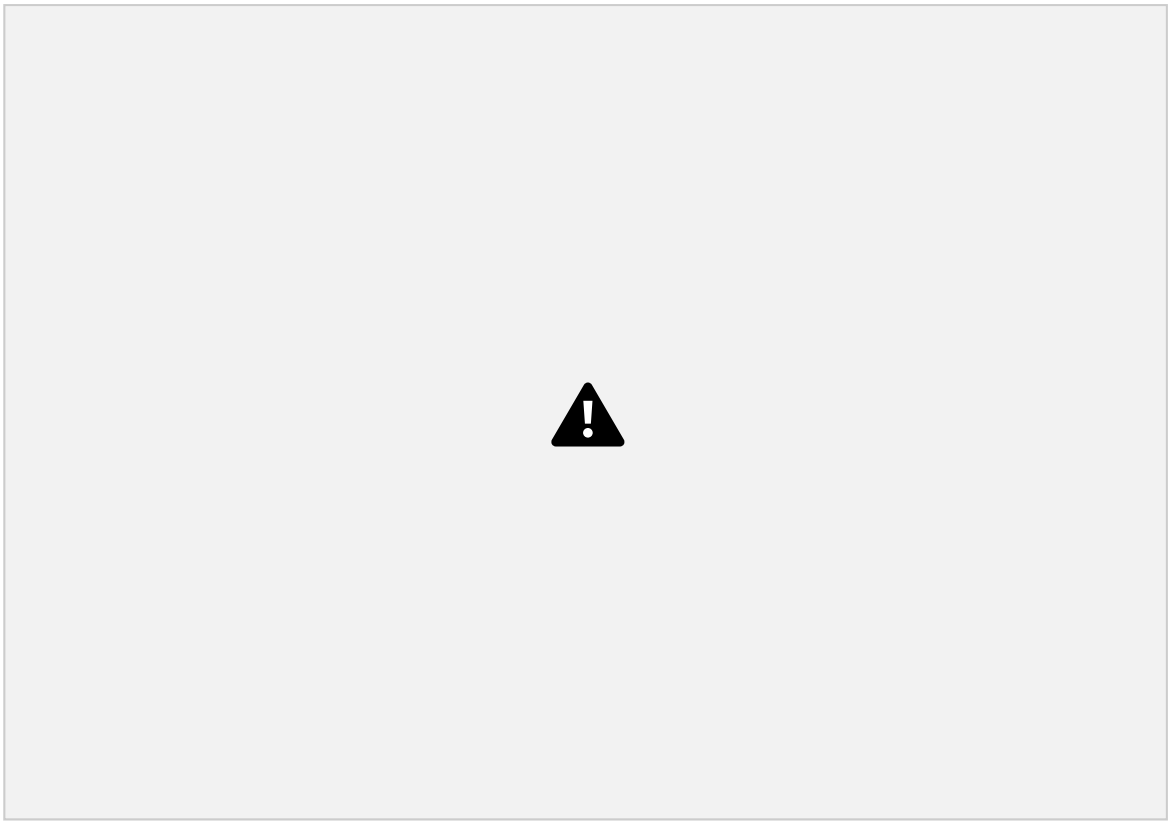

Рис 4.2.13 Смужкова діаграма для ознаки technology

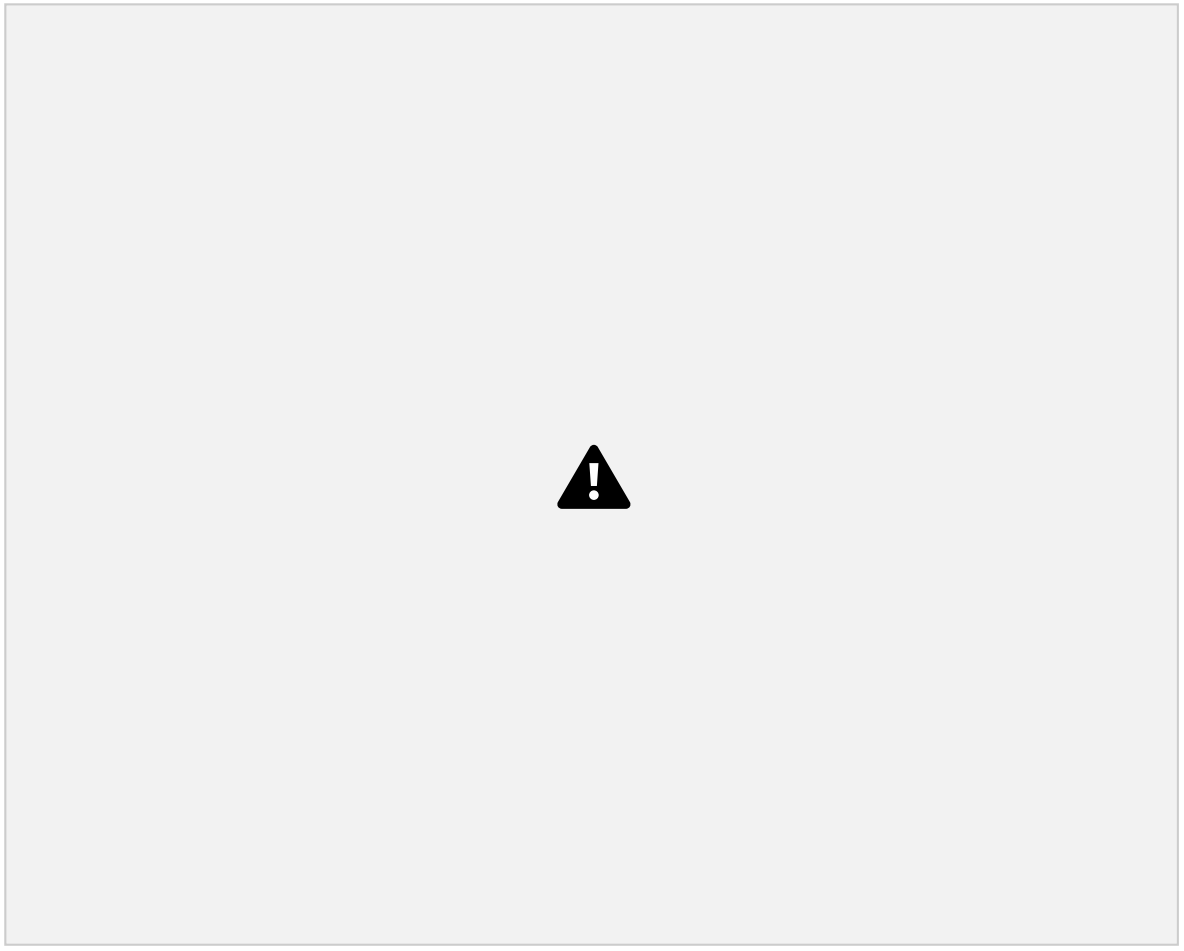

Рис 4.2.14 Смужкова діаграма для ознаки platform

За отриманими візуалізаціями можна припустити, що ознаки: difficulty, genre,

developer, publisher, technology і platform підійдуть у ролі предикторів. Дане припущення необхідно перевірити чисельними методами. Для використання методу ANOVA ознаки мусять мати нормально розподілені класи. Контроль типу розподілу виконано за критерієм Шапіро-Уілка [8].

Рис 4.2.15.1 Статистика Шапіро-Уілка для ознаки genre

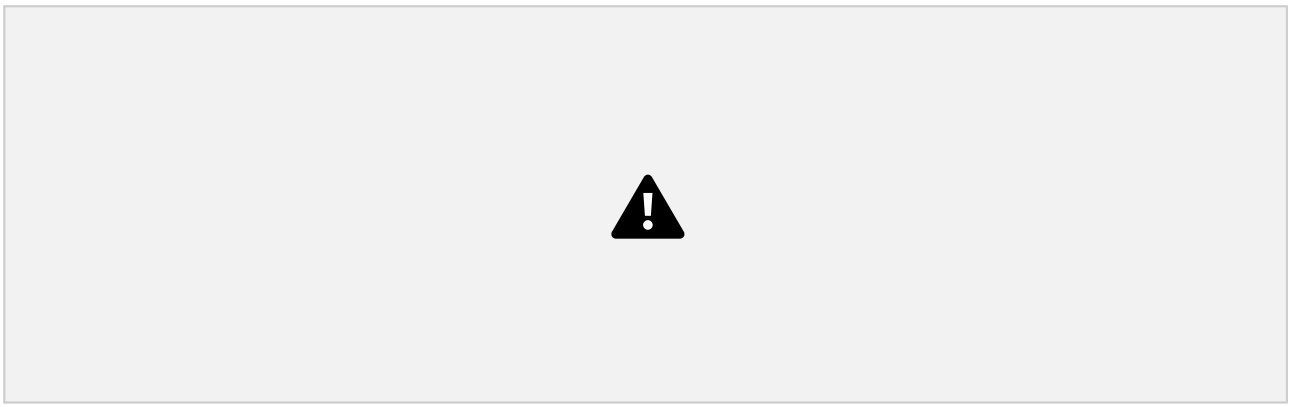

Рис 4.2.15.2 Репрезентативний уривок з набору значень статистики Шапіро Уілка для ознаки genre

Рис 4.2.16.1 Статистика Шапіро-Уілка для ознаки difficulty

Рис 4.2.16.2 Набір значень статистики Шапіро-Уілка для ознаки difficulty

Ідентичні дії повторено для інших категоріальних ознак. У всіх випадках нульова гіпотеза критерію була відкинута — підкласи не розподілені за нормальним законом.

Для таких даних підійде непараметричний метод Крускала-Уоліса

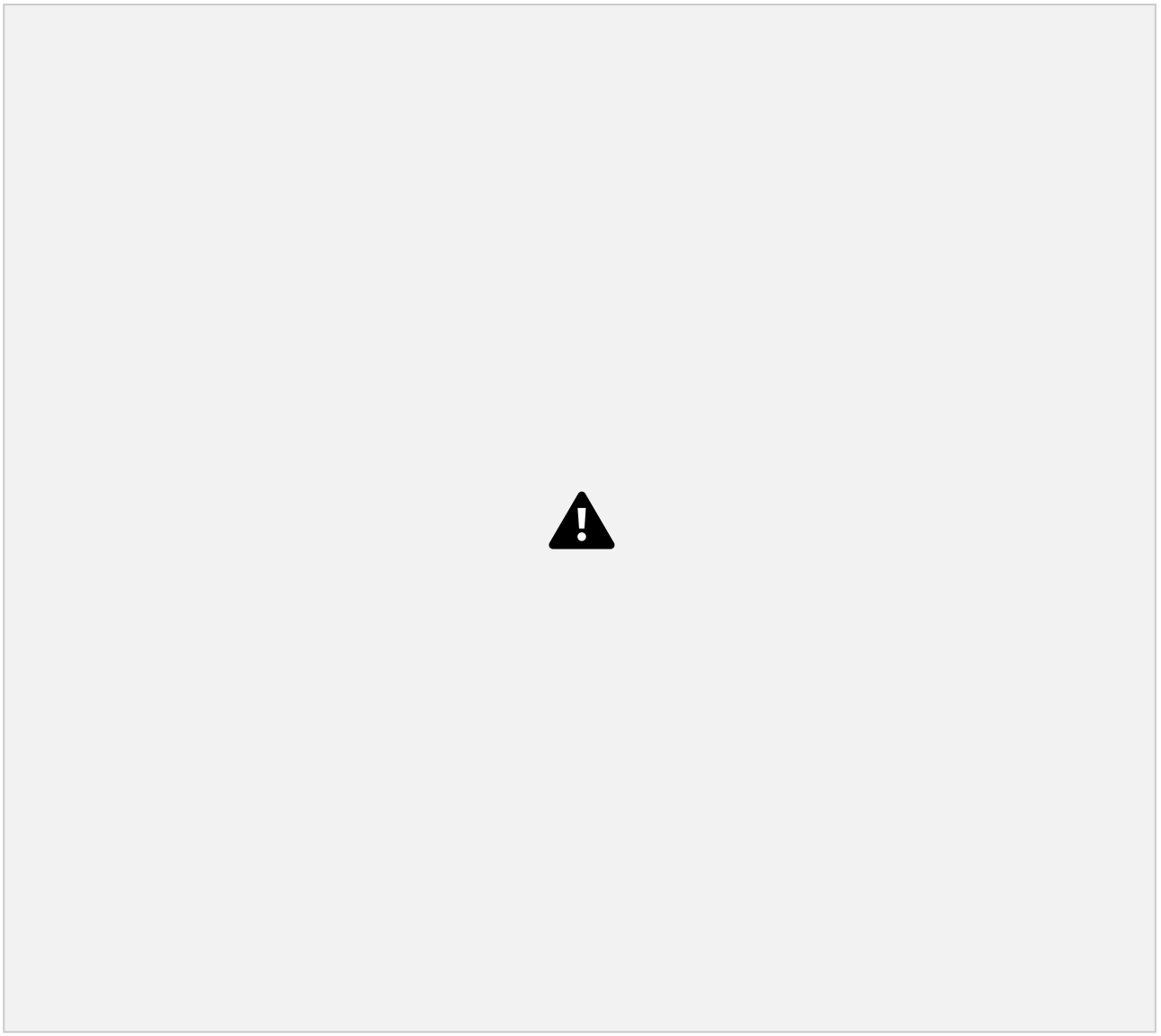

Рис 4.2.17 Тест Крускала для всіх якісних ознак

Рис 4.2.18 Результати тесту Крускала

Тест Крускала, хоч і не здатний прямо вказувати коефіцієнт кореляції ознак, утім дає чітке розуміння того, що всі протестовані ознаки мають принаймні слабкий значущий вплив на цільову ознаку.

Якщо відсортувати показники за спаданням впливу вийде такий топ:

1. Розробник
2. Видавець

3. Технології
4. Рівень складності
5. Жанр
6. Ресурс
7. Платформа
8. Операційна система

Для співставлення з даним тестом через регресор дерева рішень обчислено коефіцієнти важливості.

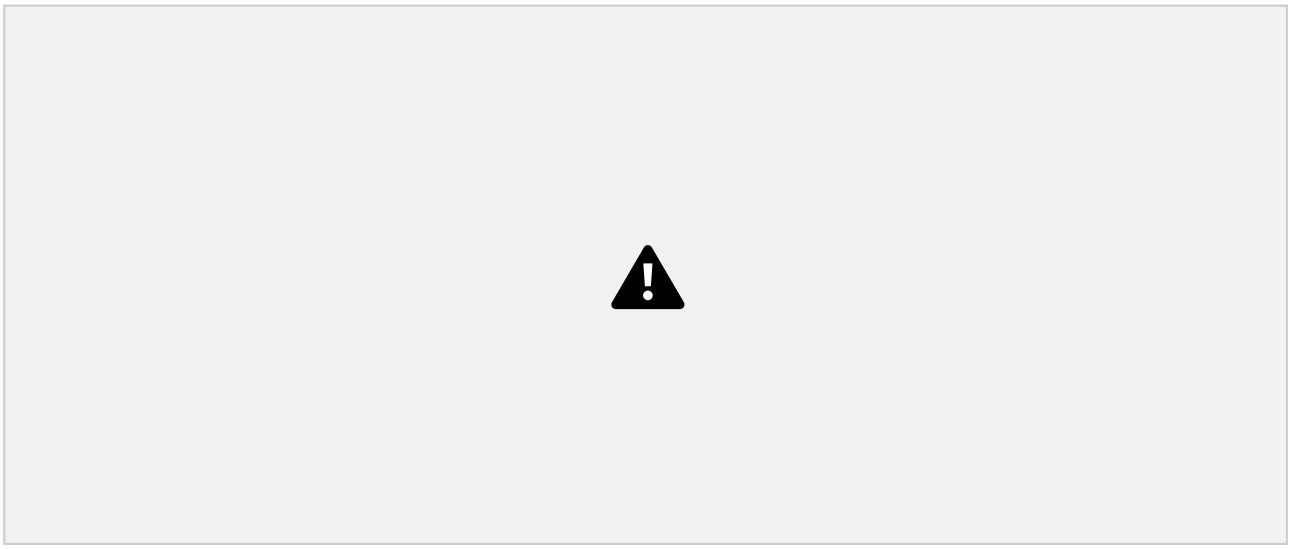

Рис 4.2.19 Оцінка важливості деревом рішень

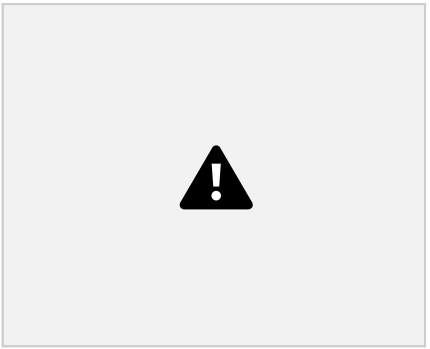

Рис 4.2.20 Топ-10 показників за коефіцієнтом важливості

За попередніми міркуваннями було обрано такі числові предиктори: mdn_play_time, time_to_beat.

Комбінуючи тест Крускала та оцінки важливості, було обрано такі якісні предиктори: developer, publisher, difficulty, technology.

Ці 6 ознак стануть стартовою множиною предикторів для проведення випробовувань різних моделей.

4.3 Обґрунтування вибору методів інтелектуального аналізу даних Для першої стадії підбору методів аналізу було обрано 5 концептуально різних моделей.

1. Множинна лінійна модель, що покаже степінь лінійності вибірки та наявність мультиколінеарності
2. Поліноміальна регресія, як покращена версія звичайної лінійної моделі
3. Дерево рішень, як класичний метод на основі дерев
4. Випадковий ліс — більш стійка до перенавчання за рахунок вбудованої регуляризації та нечутлива до мультиколінеарності модель 5. XGBoost — найбільш просунута модель, добре адаптована до великих обсягів даних, теж здатна проводити регуляризацію.

Рис 4.3.1 Формування базової тренувальної/тестової вибірки

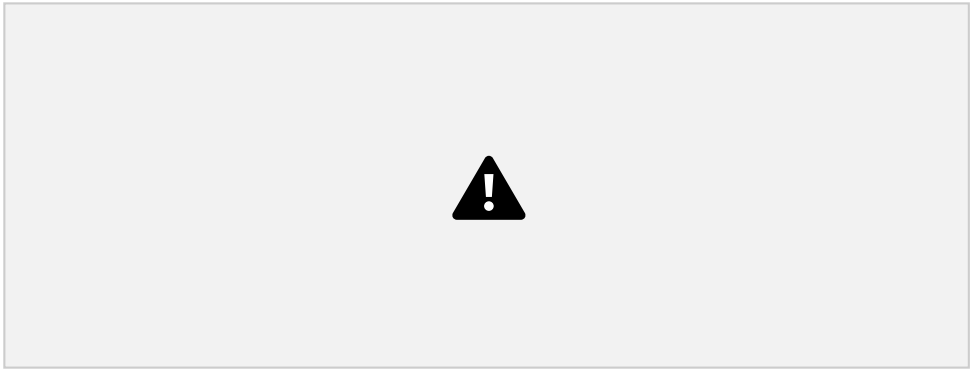

Рис 4.3.2 Результати навчання моделей на базовій версії вибірки Далі множинна лінійна та поліноміальна регресії тестуватися не будуть, через відчутну часову складність та слабку результативність.

Рис 4.3.3 Формування вибірки з урахуванням платформи

Рис 4.3.4 Результати навчання моделей вибірці з урахуванням платформи

Рис 4.3.5 Формування вибірки з урахуванням платформи, жанру

Рис 4.3.6 Результати навчання моделей вибірці з урахуванням платформи, жанру

Рис 4.3.7 Формування вибірки з урахуванням платформи, жанру, операційної системи

Рис 4.3.8 Результати навчання моделей вибірці з урахуванням платформи, жанру, операційної системи

За результатами практичної перевірки моделей на вибірках із різним складом параметрів, було встановлено, що додатковий предиктор platform покращує покриття моделей майже на 6 відсотків.

Тому буде доданий до робочої множини предикторів.

Параметри os та genre навпаки лише погіршують показники моделей, ймовірно внаслідок перенавчання. Вони не братимуть участі у подальшому підборі моделей.

Для перевірки впливу ресурсу, що виставляє оцінку, було проведено окреме випробовування з урахуванням оновленої основної множини та показника reviewer.

Рис 4.3.9 Формування вибірки з урахуванням ресурсу-критика

Рис 4.3.10 Результати навчання моделей вибірці з урахуванням ресурсу-критика

На основі цього експерименту можна побачити, що ресурс потужно впливає на оцінку гри, додаючи до покриття моделей ще 20%

Це дає змогу отримати дуже високе значення опису цільової величини. Для спрощення - можливість прогнозувати оцінку з точністю +- 3 бали.

Незважаючи на це, з точки зору бізнес аналізу включення фактора ресурсу до моделей є не дуже логічним. Тому предиктор reviewer не буде враховано при подальшій роботі.

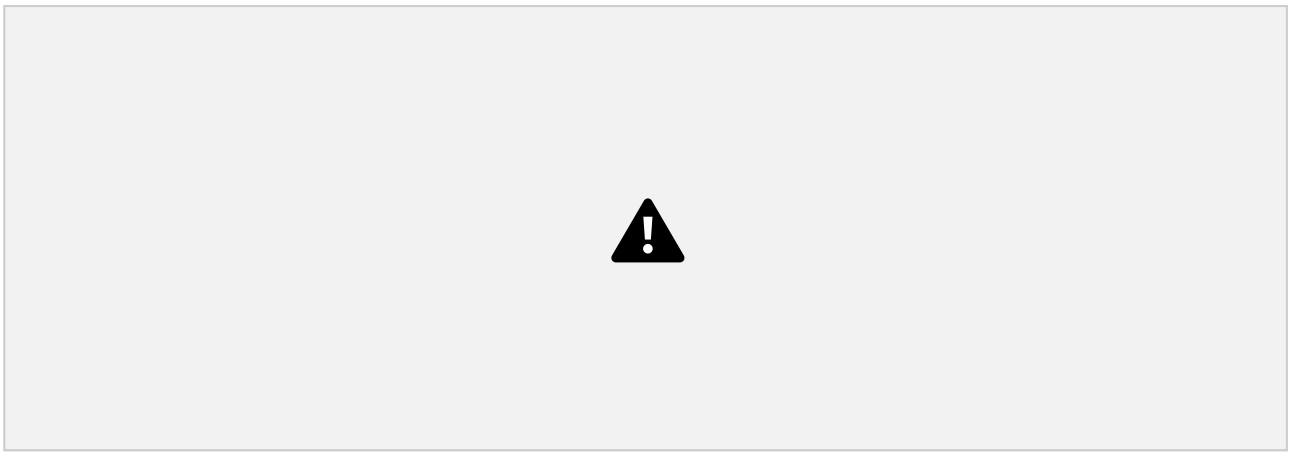

Рис 4.3.11 Результати навчання оновленого списку моделей Для порівняння було застосовано ще дві перспективні моделі KNN regressor та LightGBM. Серед 5 непоганих моделей було обрано три найкращих: RandomForest, XGBoost та LGMB.

Слід пересвідчитись у стійкості моделей за допомогою крос-валідації.

Рис 4.3.12 Крос-валідація випадкового лісу

Рис 4.3.13 Середнє значення метрики випадкового лісу протягом крос-валідації

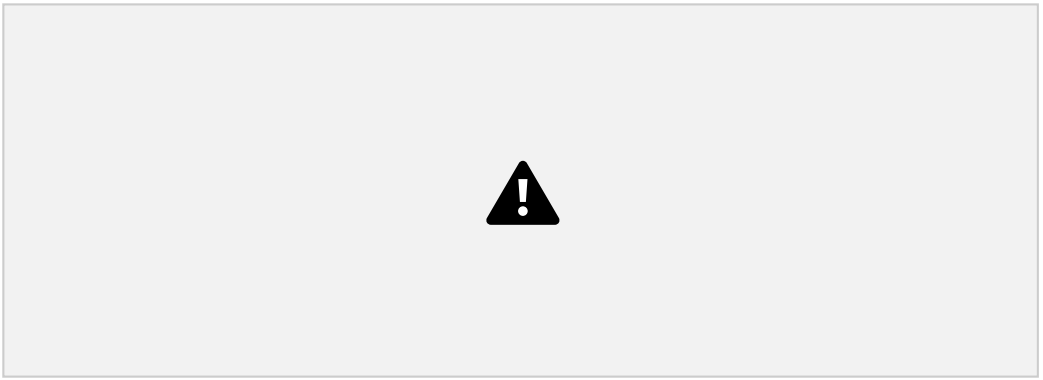

Рис 4.3.14 Крос-валідація XGBoost[4]

Рис 4.3.15 Середнє значення метрики XGBoost протягом крос-валідації

Рис 4.3.16 Крос-валідація LightGBM[5]

Рис 4.3.17 Середнє значення метрики LightGBM протягом крос-валідації

Кожна з 3х моделей показала хороший результат крос-валідації, що дає підстави вважати їх стійкими і добре адаптованими до даних.

4.4 Налаштування параметрів обраних моделей

Під час крос-валідація було помічено прояв додаткової практичної переваги моделі LightGBM — швидкодію. Тому саме для цієї моделі буде здійснена спроба підбору гіперпараметрів шляхом пошуку за сіткою.

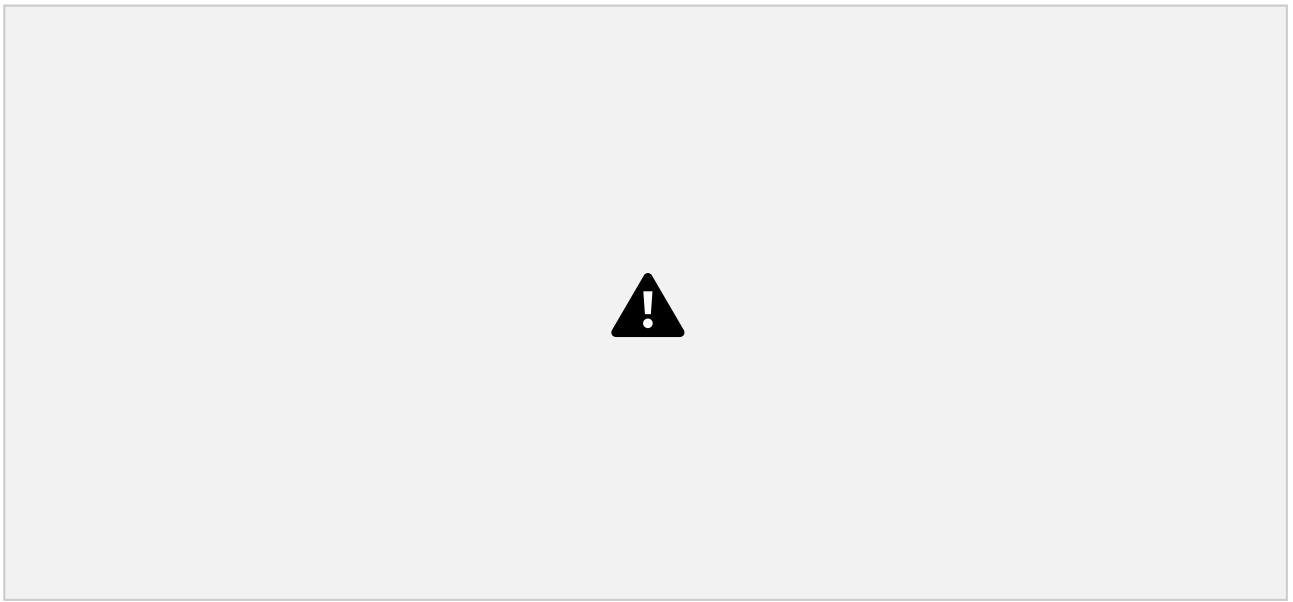

Рис 4.4.1 Підбір параметрів за сіткою для моделі LightGBM

Рис 4.4.2 Демонстрація метрик кращого зразка моделі LightGBM

Налаштування можна вважати успішним, адже фактично вдалося покращити значення коефіцієнту детермінації.

Для двох інших моделей налаштування не проводилося по причині апаратних обмежень. Навіть для найшвидшої моделі час проведення пошуку становив 70 хв.

4.5 Візуальне порівняння відібраних моделей

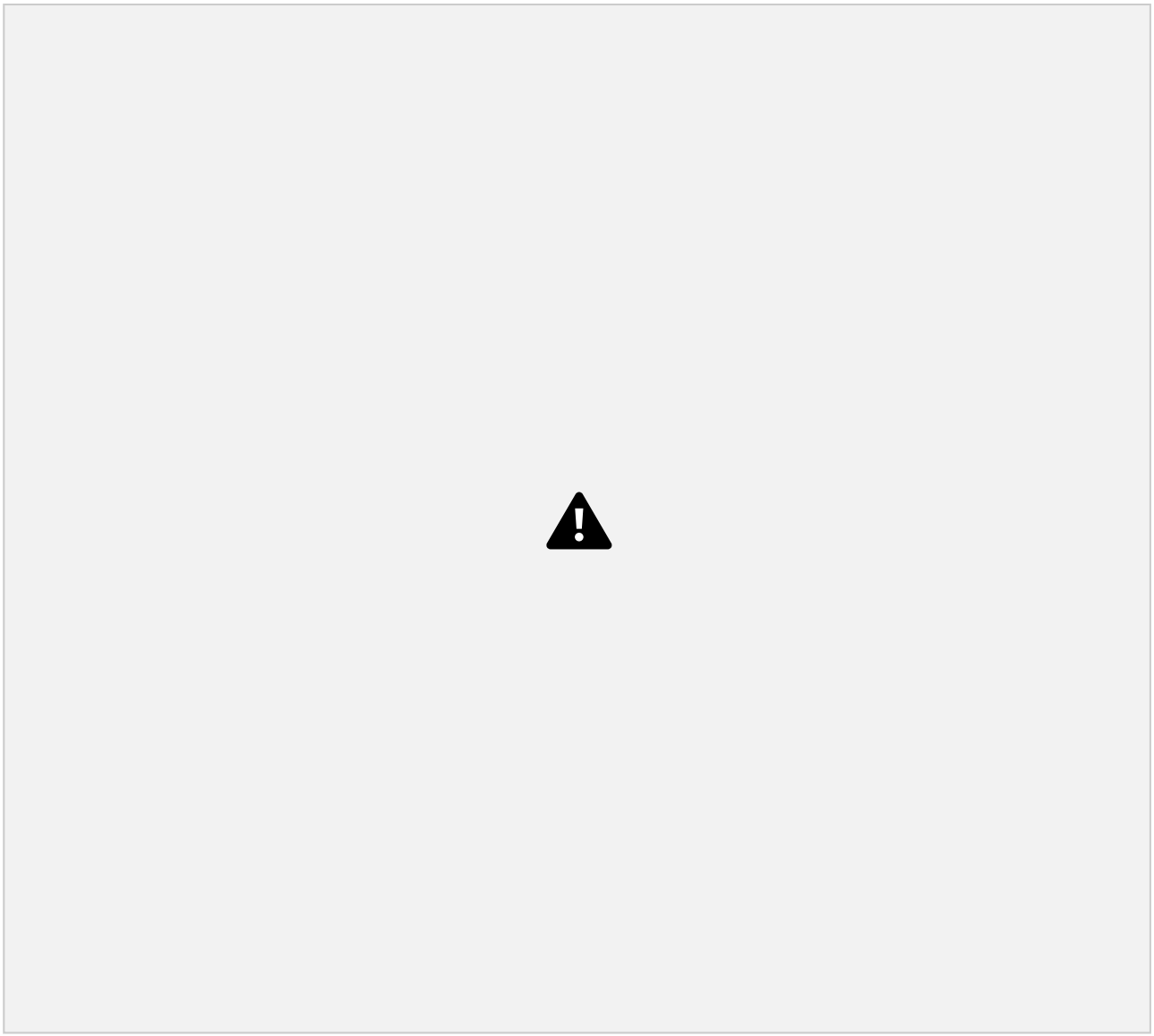

Рис 4.5.1 Порівняльна діаграма фактичних та прогнозованих значень

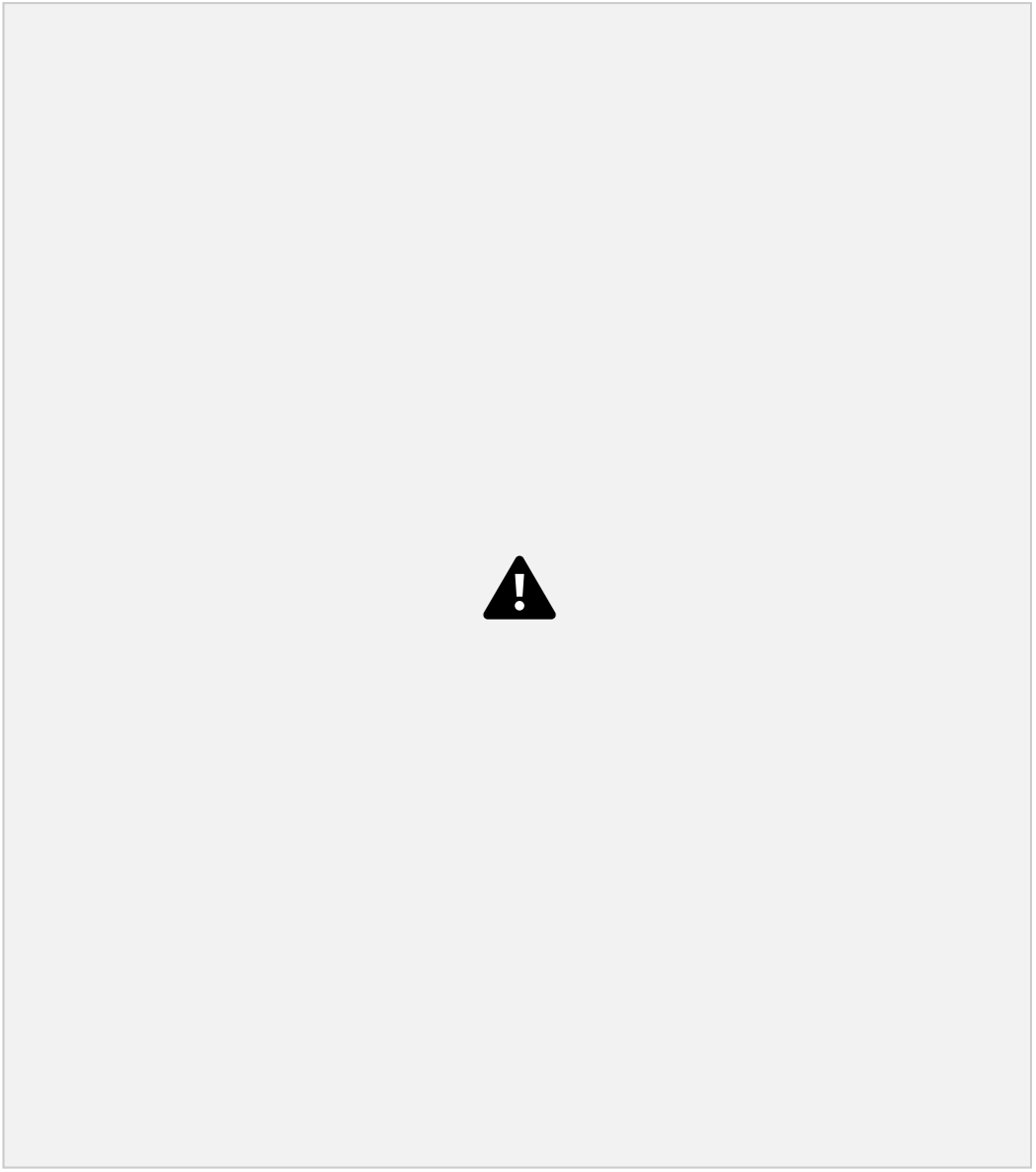

Рис 4.5.2 Порівняльна діаграма середньої квадратичної похибки моделей[3]

У цілому отримані графіки не демонструють істотних відмінностей між моделями, але дозволяють виявити лідера – модель Light GBM. Вона в усьому трохи краще за конкуруючі моделі і додатково вирізняється відмінною швидкодією.

ВИСНОВКИ

Під час дослідження, проведеного на основі підготовленого тематичного сховища даних, вдалося успішно виділити підмножину важливих для прогнозу ознак. При цьому процес відбору включав фільтрування за бізнес складовою, щоб максимізувати прикладне значення аналізу. Застосування цієї підмножини

дозволило натренувати серію моделей різних типів. Помітно, що моделі на основі дерев демонструють набагато кращі результати при налаштуваннях за замовчуванням, ніж інші типи моделей.

Крім того, було виявлено, що метрики ігрового процесу, такі як час проходження гри або пікова кількість гравців, практично не впливають на значення оцінки при розгляді у великому масштабі, оскільки вони стають майже незалежними. Жанрова спрямованість також не несе вагомої інформації, оскільки кожен жанр має свою масу прихильників, яка ефективно корегує оцінку. Зате вага платформи виявилася несподівано високою. Це каже про те, що фахівцям не рекомендується приступати до нового проекту без попереднього аналізу спектру популярних платформ.

Цікавим відкриттям стало те, що приблизно 20% всієї варіативності оцінки залежить від ресурсу, що її виставляє. Таким чином GameFaQ та IGDB більш схильні до критичних, дуже низьких або навпаки високих оцінок, Steam має багато інді-проектів з високими балами, тому медіана оцінки там найвища, а Metacritic в свою чергу вирізняється жорсткішою модерацією, тому проекти від поважних студій мають там стабільно вищу оцінку.

Загалом, результати цього дослідження можуть допомогти розробникам початківцям отримати ідеї для прогнозування поведінки своїх гравців, а звичайним любителям індустріїзрозуміти фундаментальні причинно-наслідкові зв’язки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація Python 3.12 – Режим доступу: https://docs.python.org/3.12/uk/>
2. Документація Pandas – Режим доступу: https://pandas.pydata.org/pandas docs/stable/index.html#translation
3. Документація Matplotlib – Режим доступу: https://matplotlib.org/stable/users/translation.html
4. Документація XGBoost – Режим доступу: https://xgboost.readthedocs.io/en/stable/> (немає української версії)

5. Документація LightGBM – Режим доступу: https://lightgbm.readthedocs.io/en/latest/index.html (немає української версії)
6. Документація Seaborn – Режим доступу: https://seaborn.pydata.org/index.html (немає української версії) 7. Документація Scikit-sklearn – Режим доступу: https://scikit-learn.org/stable/ (немає української версії)

ДУ

Тексти програмного коду прогнозування оцінки відеогри за даними SteamSpy, Metacritic, IGDB, GameFAQ

(Найменування програми (документа))

HDD

(Вид носія даних)

348 арк, 795 Кб

(Обсяг програми (документа), арк.,

студента групи ІП-21 ІІ курсу

Створення таблиць зони обробки:

```
with engine.connect() as connection:
    query = text(f"""
CREATE SCHEMA IF NOT EXISTS Stage;

DROP TABLE IF EXISTS
stage.metacritic_review; CREATE TABLE
stage.metacritic_review( id SERIAL PRIMARY
KEY,
metascore TEXT,
game TEXT,
platform TEXT
);

DROP TABLE IF EXISTS stage.requirements;
CREATE TABLE stage.requirements(
id SERIAL PRIMARY KEY,
steam_game_id TEXT,
pc_req TEXT,
mac_req TEXT,
linux_req TEXT,
min_req TEXT
);

DROP TABLE IF EXISTS stage.all_steam;
CREATE TABLE stage.all_steam(
id SERIAL PRIMARY KEY,
game TEXT,
link TEXT,
release TEXT,
rating TEXT,
publisher TEXT,
developer TEXT,
technologies TEXT,
all_time_peak TEXT,
all_time_peak_date TEXT
);

DROP TABLE IF EXISTS stage.play_time_by_player;
CREATE TABLE stage.play_time_by_player( id
SERIAL PRIMARY KEY,
game_id TEXT,
game TEXT,
action_type TEXT,
time TEXT
);

DROP TABLE IF EXISTS stage.genres;
CREATE TABLE stage.genres(
id SERIAL PRIMARY KEY,
game TEXT,
release_date TEXT,
developer TEXT,
```

```
genres TEXT,
summary TEXT
);

DROP TABLE IF EXISTS stage.regions;
CREATE TABLE stage.regions(
id SERIAL PRIMARY KEY,
game TEXT,
platforms TEXT,
year_of_release TEXT,
genre TEXT,
publisher TEXT,
na_players TEXT,
eu_players TEXT,
jp_players TEXT,
other_players TEXT,
global_players TEXT,
developer TEXT,
rating TEXT
);

DROP TABLE IF EXISTS stage.mdn_play_time;
CREATE TABLE stage.mdn_play_time( id
SERIAL PRIMARY KEY,
steam_game_id TEXT,
game TEXT,
release_date TEXT,
developer TEXT,
publisher TEXT,
platforms TEXT,
genres TEXT,
avg_play_time TEXT,
median_play_time TEXT,
owners TEXT
);

DROP TABLE IF EXISTS stage.time_to_beat;
CREATE TABLE stage.time_to_beat( id
SERIAL PRIMARY KEY,
achivements TEXT,
description TEXT,
developers TEXT,
gfq_difficulty TEXT,
gfq_rating TEXT,
grnk_score TEXT,
hltb_complete TEXT,
hltb_single TEXT,
igdb_score TEXT,
igdb_single TEXT,
igdb_uscore TEXT,
languages TEXT,
game TEXT,
platforms TEXT,
published_hltb_date TEXT,
publishers TEXT,
tags TEXT,
voiceovers TEXT
```

```
);
""")
    connection.execute(query)
    connection.commit()
```