МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра комп’ютерних наук

КУРСОВА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни Інтелектуальний аналіз даних
на тему:

«Візуалізація продажу відеоігор»

студента III курсу групи КН-20-1
спеціальності 122 «Комп’ютерні науки»
Дяченко Валерії Володимирівни
(прізвище, ім’я та по-батькові)

Керівник ст. викл. каф. КН Марчук Г. В.

Дата захисту: " ___ " _____ 2023 р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ Граф М.С.
(підпис) (прізвище та ініціали)
_____ Марчук Г. В.
(підпис) (прізвище та ініціали)
_____ Коротун О.В.
(підпис) (прізвище та ініціали)

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерних наук
Освітній рівень: бакалавр
Спеціальність 122 «Комп'ютерні науки»

«ЗАТВЕРДЖУЮ»
Зав. кафедри
_____ Граф М.С.
"____"_____2023р.

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТУ
Дяченко Валерії Володимирівні

1. Тема роботи: Візуалізація продажу відеоігор, керівник курсової роботи: Марчук Г. В.
2. Строк подання студентом: "_____"____2023р.
3. Вхідні дані до роботи: Використовуючи різні алгоритми провести візуальний аналіз продажу ігор, представити графіки і діаграми, зробити висновки на основі отриманих даних.
4. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
1. Аналіз проблематики, методів та засобів вирішення задачі;
2. Практична реалізація алгоритмів інтелектуального аналізу даних
3. Аналіз результатів підсумкові графіки, таблиці
5. Перелік графічного матеріалу(з точним зазначенням обов'язкових креслень)
   a. Презентація
   b. Репозиторій:

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3			

7. Дата видачі завдання " " березня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсової роботи	Строк виконання етапів роботи	Примітки
1	Постановка задачі		
2	Пошук та огляд теоретичної частини		
3	Опрацювання літературних джерел		
4	Формулювання технічного завдання		
5	Розробка програмної частини		
6	Опис результатів		
7	Написання пояснювальної записки		
8	Захист		

Студент _____ Дяченко В. В.
(підпис) (прізвище та ініціали)

Керівник проекту _____ Марчук Г. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсової роботи на тему «Візуалізація продажу відеоігор». Складається з реферату, вступу, трьох розділів за змістом, висновків після них, загального висновку, та списку використаної літератури/додатку.

Текстова частина викладена на 41 сторінці друкованого тексту.

Пояснювальна записка має 14 сторінок додатку. Список використаних джерел містить 10 найменувань, і займає 1 сторінку.

В роботі наведено 27 рисунків. Загальний обсяг роботи - 59 сторінок.

У першому розділі була проаналізована поставлена задача, як результат обрано необхідний інструментарій та алгоритми для виконання.

У другому розділі була побудована модель та практично реалізовані відповідні алгоритми і методи.

У третьому розділі були графічно представлені результати роботи, а також проведений їх аналіз.

Загальний висновок містить у собі результати виконаної роботи.

У додатку представлений повний код реалізації алгоритмів.

Ключові слова: АНАЛІЗ, ДАНІ, ВІДЕОІГРИ, ПРОДАЖІ, PYTHON, ВІЗУАЛІЗАЦІЯ, ГРАФІКИ.

ЗМІСТ

ВСТУП

Актуальність теми. Останнє десятиріччя об’єм отримуваної інформації зростає щоденно. Величезні обсяги даних зберігаються в різних форматах та джерелах (включаючи бази даних, хмарні сховища, соціальні мережі та платформи і т.д.). Отримання та аналіз цих даних став необхідністю як для створення нових технологій, так і покращення ефективності/оптимізації вже існуючих. Аналіз допомагає приймати важливі та змістовні рішення, так як практично кожна сфера діяльності потребує глибоке розуміння зв’язків і відповідних паттернів, що формуються на основі отриманих даних. Таким чином можна впевнено сказати, що грамотний та інформативний аналіз забезпечує успіх будь-якого починання. І ігрова індустрія не є винятком.

Віртуальні розваги суттєво впливають на економіку та культуру. Тому розуміння ринкових тенденцій має важливе значення для розробників, видавництв, інвесторів та інших зацікавлених сторін. Аналіз ринку дозволяє зробити висновки та покращити стратегії розвитку/маркетингу компаній в цій індустрії.

Об’єктом дослідження є набір даних «Video Game Sales», його динаміка та особливості.

Предметом дослідження є візуальний аналіз наявних даних та вплив різних факторів (таких як жанр, рік, платформа, розробник) на продажі у різних країнах.

Метою дослідження є аналіз та візуалізація продажу відеоігор з використанням різних методів інтелектуального аналізу. Як результат виявлення тенденцій, залежностей та закономірностей, для підведення підсумків, що дозволять зробити обґрунтовані рекомендації для покращення стратегій маркетингу, розвитку і управління.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз задачі, засобів та методів її вирішення

Існують різні види аналізу та представлення інформації. Враховуючи мету роботи нас цікавить саме візуальний аналіз даних. З назви зрозуміло, що даний варіант представлення інформації використовує графічні елементи для наочного зображення наявних даних. А саме графіки та діаграми, інфографіки, теплові карти, графи, панелі управління і т.д.

Вибір конкретних елементів залежить від типу даних, характеру аналізу та мети візуального представлення. Наприклад, у випадку з продажом можна використовувати: лінійні графіки, стовпчикові діаграми, кругові діаграми, теплові карти, матрицю кореляції та графіки розсіювання.

Для їх створення нам знадобиться відповідний інструмент, а саме – matplotlib та Seaborn. Це дві популярні бібліотеки для візуалізації даних в мові програмування Python. Вони є гарним варіантом для роботи так як надають широкий спектр можливостей для реалізації поставленої задачі. Прості у використанні, гнучкі, підтримують різноманітні види графіків, міжплатформені – усе це переваги.

Для запуску та виконання коду Python використаємо платформу від Google - Google Colab. Вона базується на середовищі Jupyter Notebook і дозволяє користувачам створювати та виконувати код прямо у браузері без необхідності встановки та налаштування середовища на своєму комп'ютері. Google Colab безкоштовний, використовує хмарне середовище, містить багато популярних бібліотек, дозволяє працювати з іншими користувачами та підтримує використання графічних прискорювачів (GPU) для обчислень, що особливо корисно при виконанні задач машинного або глибинного навчання.

Для кращої візуалізації даних також можна та необхідно використовувати розвідувальний аналіз, кластерний аналіз, асоціативний аналіз та прогнозування за допомогою регресі.

Розвідувальний аналіз даних є першим кроком у процесі і дозволяє зрозуміти основні характеристики та властивості набору даних. Він допомагає вивчати їх структуру, перевіряти якість та цілісність, а також робити перші висновки про можливі залежності та взаємозв'язки.

Кластерний аналіз дозволяє групувати дані на основі певних характеристик. Це допомагає виявити внутрішню структуру даних та зрозуміти залежності між об'єктами. Також стає легше візуалізувати великі набори даних, розділяючи їх на схожі групи або кластери. На основі кластерного аналізу можна розбити користувачів на групи з подібними вподобаннями та пропонувати рекомендації в межах кожної.

Асоціативний аналіз виявляє приховані асоціації між різними елементами в наборі даних. Це може допомогти виявити нетипові або підозрілі зв'язки між елементами даних.

Моделі регресії в свою чергу є одним з ключових інструментів у інтелектуальному аналізі даних. Основна мета - побудова математичної моделі, яка може прогнозувати значення залежної змінної на основі незалежних. У нашому випадку це може бути прогнозування майбутніх продажів відеоігор на основі наявних даних, враховуючи такі фактори, як жанр, розробник, рік випуску тощо.

1.2 Методи і алгоритми інтелектуального аналізу даних для рішення

Для візуалізації даних можуть використовуватись різні формули та математичні вирази. Кожен графік будується на основі своєї.

- Формула лінійного графіка: $y = mx + b$

  y - значення на осі Y, x - значення на осі X, m - нахил лінії (коефіцієнт нахилу), b - зсув лінії (перетин з віссю Y).
- Формула кругової діаграми: відсоток = (значення / загальна сума значень) * 100.
- Формула стовпчикової діаграми: Висота стовпчика = значення даних

Значення даних відображаються на вертикальній осі (*Y*), в той час як на горизонтальній (*X*) записані категорії або мітки, що відповідають кожному стовпчику.

- Формула теплової карти: колір = (значення - мінімальне значення) / (максимальне значення - мінімальне значення)

  Отриманий числовий результат використовується для визначення кольору або інтенсивності на тепловій карті. Зазвичай це кольорова шкала, де маленькі значення відображаються світлими кольорами, а великі - темними.
- Формула графіка розсіювання: формально не є математичною формулою, а лише методом візуалізації залежності між двома змінними. Точки на графіку розташовуються відповідно до значень двох змінних (*x, y*) і показують їх взаємозв'язок.
- Формула матриці кореляції за Пірсоном: використовується для вимірювання ступеня лінійної залежності між двома змінними. Має наступний вигляд:

  $r = \sum((x - \bar{y})(y - \dot{y}))/\sqrt{(\sum(x - \bar{y})^2 * \sum(y - \dot{y})^2)}$

  r - коефіцієнт кореляції за Пірсоном, *x, y* - значення змінних, $\bar{y}$, $\dot{y}$ - середні значення змінних, Σ - сума значень за всіма спостереженнями.

  Вона дозволяє обчислити коефіцієнт кореляції, який може приймати значення в діапазоні від -1 до 1. Значення 1 вказує на повну позитивну лінійну залежність, -1 вказує на повну негативну лінійну залежність, а значення 0 вказує на відсутність лінійної залежності між змінними.
- Формула оцінки щільності KDE (Kernel Density Estimation) виглядає наступним чином:

  $KDE(x) = (1/nh) * \sum K((x - x^i)/h)$

x - точка, для якої обчислюється оцінка щільності, x_i - точки з набору даних, K - ядро, яке використовується для згладжування, h - параметр ширини ядра (bandwidth), n - кількість точок в наборі даних.

Загальна ідея полягає у використанні ядра K для створення згладженої функції щільності, яка наближає розподіл даних. Кожна точка x має свій внесок до оцінки щільності, пропорційний відстані між x і xi, нормованої за допомогою ширини ядра h.

- Формула градієнта кольору: RGB = (R, G, B), де R, G і B - значення червоного, зеленого та синього каналів відповідно.

Що до кластерного аналізу, існує кілька алгоритмів та функцій, які можуть бути використані для його проведення. Наприклад, K-means. Це один з найпоширеніших алгоритмів. Він розділяє дані на K кластерів шляхом мінімізації суми квадратів відстаней між точками даних та центрами кластерів. Алгоритм складається з двох основних кроків: призначення точок до кластерів і перерахунок центроїдів.

Спочатку задається кількість кластерів K, яку необхідно сформувати. Далі обираються початкові центроїди (точки, що вибрані випадково або на основі певних критеріїв). Розраховується відстань (наприклад, Евклідова) між кожною точкою і всіма центроїдами, а згодом вона призначається до кластеру з найближчим центроїдом. Дані дії повторюються поки всі точки не будуть призначені. Після цього переходимо до перерахунку центроїдів. Обчислюється нове розташування, що відображає середнє значення всіх призначених точок. Розрахунки продовжуються поки центроїди не стабілізуються або не досягнеться максимальна кількість ітерацій.

Переваги K-means у тому, що це досить простий у реалізації та ефективний алгоритм. Він добре працює з великим обсягом даних і може бути застосований у різних галузях.

Звісно, в нього присутні й недоліки. Він потребує точного зазначення параметру K (к-ті кластерів), що може бути складним завданням, особливо

якщо немає явних підказок. Також вибір початкових центроїдів може вплинути на кінцевий результат, тому іноді для одного набору даних формуються різні кластери. Ще один недолік це чутливість до викидів, тобто окремих об'єктів, які віддалені від інших точок. Вони можуть вплинути на розташування центроїдів та призначення об'єктів до кластерів. Тому потрібно обов’язково враховувати особливості вхідних даних при використанні алгоритму K-means.

Асоціативний аналіз також має багато алгоритмів і функцій. Наприклад Apriori. Він є одним з найбільш відомих та широко застосовуваних методів у галузі. Використовується для виявлення цікавих асоціацій та залежностей між різними елементами в наборі даних. Основним принципом алгоритму Apriori є використання поняття підтримки (support) та правил підтримки (support rules).

Підтримка визначає, наскільки часто певний набір елементів зустрічається в наборі даних. Це міра «популярності» або «поширеності» певної комбінації. Підтримка може бути обчислена як відношення кількості транзакцій, що містять дану комбінацію, до загальної кількості транзакцій. Вона виражається у відсотках або десятковому значенні. Правила в свою чергу це комбінації елементів, які мають достатньо високу підтримку. Вони складаються з двох частин: антецеденту (англ. antecedent) і консеквенту (англ. consequent). Антецедент - це множина елементів у транзакції, а консеквент – ті, що з'являються внаслідок.

Алгоритм Apriori складається з:

- Визначення мінімального значення підтримки.
- Генерації itemsets. (враховуючи мінімальну підтримку)
- Визначення правил, які відповідають умовам.
- Повторення кроків 2 та 3 для знаходження більш складних асоціацій/правил.

Цей алгоритм досить простий у реалізації та чудово працює з великим обсягом даних. Звісно через це може бути вимогливим до ресурсів та не дуже продуктивним.

Для прогнозування за допомогою регресі можуть бути використані різні моделі. Наприклад, проста лінійна регресія, множинна регресія, або складніші такі як «випадковий ліс».

Лінійна регресія є одним з найпростіших і поширених методів регресійного аналізу. Проста у використанні, швидко обчислюється та добре підходить для прогнозування. Проте не може моделювати складні взаємозв'язки між багатьма змінними. А також чутлива до викидів або аномальних значень, що можуть спотворити результати.

Множинна регресія в свою чергу дозволяє враховувати вплив кількох незалежних змінних одночасно, тому робить більш повний і точний прогноз. Хоча при занадто великій к-ті незалежних змінних у моделі може виникнути проблема перенасиченості (overfitting) або мультиколінеарності.

«Випадковий ліс» чудово працює з великою кількістю даних без потреби в попередній обробці. Має властивість виявляти та контролювати перенавчання, а також прогнозує значення навіть при наяві складних неоднорідностей у даних. Однак, через це може бути обчислювально витратним при великій к-ті інформації.

Висновки до першого розділу

Провівши ретельний аналіз, в першому розділі, було визначено алгоритми та інструментарій для найкращого виконання поставленої задачі. Було розглянуто основні складові візуального аналізу та обрано інструментарій для роботи із ними. Також описано деякі формули і алгоритми інтелектуального аналізу даних для рішення. Враховано їх переваги та недоліки.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

2.1 Характеристика джерела даних для проведення аналізу

Для проведення подальшого аналізу скористуємось даними із джерела https://www.kaggle.com. А саме набором даних про продаж відеоігор «Video Game Sales». Обраний датасет включає у себе інформацію про понад 16.5 тисяч відеоігор. Продажі кожної перевищують 100 000 копій. Він був згенерований шляхом обробки джерела vgchartz.com.

Поля набору містять:

- Rank - рейтинг загальних продажів;
- Name - назва гри;
- Platform – платформа випуску ігор (наприклад, ПК, PS4 тощо);
- Year - рік виходу гри;
- Genre - жанр гри;
- Publisher - видавець гри;
- NA_Sales - продажі в Північній Америці (у мільйонах);
- EU_Sales - продажі в Європі (у мільйонах);
- JP_Sales - продажі в Японії (у мільйонах);
- Other_Sales - продажі в решті світу (у мільйонах);
- Global_Sales – загальний обсяг продажів.

Завантаження датасету (рис. 2.1):

```
vgs = pd.read_csv(io.StringIO(uploaded['vgsales.csv'].decode('utf-8')))
```

Назви колонок, кількість непустих значень та типи даних (рис. 2.2):

```
vgs.info()
```

Дані наявні у датасеті (рис. 2.3):

```
vgs.head()
```

Рис.2.1. Завантаження датасету.

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
```

Рис.2.2. Результат виконання команди info().

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

Рис.2.3. Результат виконання команди head (5).

2.2 Побудова моделі аналізу і практична реалізація

Після завантаження та перегляду обраного датасету починаємо аналіз.

Імпорт необхідних бібліотек:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Для кращого розуміння форми розподілу даних (структури) проведемо розвідувальний аналіз, а саме отримаємо загальну інформацію, перевіримо на наявність пропущених значень або дублікатів:

```
# Перевірка на наявність пропущених значень
print(vgs.isnull().sum())

# Перевірка наявності дублікатів
print(vgs.duplicated().sum())

# Перевірка статистичних показників для числових стовпців
print(vgs.describe())
```

Якщо присутні нульові значення, позбуваємось їх шляхом заповнення або видалення:

```
# Заміна пропущених значень у стовпці "Year" на середнє значення
mean_year = vgs['Year'].mean()
vgs['Year'].fillna(mean_year, inplace=True)

# Видалення рядків з пропущеними значеннями у стовпці "Publisher"
vgs.dropna(subset=['Publisher'], inplace=True)

# Перевірка на наявність пропущених значень після заміни та видалення
print(vgs.isnull().sum())
```

Далі будуємо графік KDE (Kernel Density Estimation) для стовпця "Global_Sales":

```
sns.kdeplot(vgs['Global_Sales'])
# Встановлення меж осі x
plt.xlim(-5, 5)
# Розподілення позначок рівномірно
ticks = np.linspace(-5, 5, 9)
plt.xticks(ticks)
```

Так як наші дані мають відхилення у розподілі, за допомогою міжквартильного розмаху знайдемо усі значення, що є викидами:

```
data = vgs['Global_Sales']
Q1 = np.percentile(data, 25)  # Обчислюємо перший квартиль (25%)
Q3 = np.percentile(data, 75)  # Обчислюємо третій квартиль (75%)
IQR = Q3 - Q1  # Обчислюємо міжквартильний розмах (IQR)
lower_bound = Q1 - 1.5 * IQR  # Обчислюємо нижню межу
```

```
upper_bound = Q3 + 1.5 * IQR  # Обчислюємо верхню межу
# Вибираємо значення, які виходять за межі
outliers = data[(data < lower_bound) | (data > upper_bound)]
```

Медіана і середнє значення:

```
median_sales = vgs['Global_Sales'].median()
mean_sales = vgs['Global_Sales'].mean()
```

Візуалізуємо викиди для усіх типів продажів за допомогою «ящиків з вусами» (box plot):

```
from matplotlib import pyplot as plt
# Визначаємо стовпці, які містять дані про продажі
sales_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']
# Створюємо фігуру та вісі
fig, ax = plt.subplots(figsize=(10, 6))
# Створюємо "ящик з вусами" із використанням boxplot
bp = ax.boxplot(vgs[sales_columns], patch_artist=True, notch=True, vert=False, showfliers=True)
# Встановлюємо кольори для кожного "ящика" в залежності від стовпця продажів
colors = [plt.cm.plasma(i / len(sales_columns)) for i in range(len(sales_columns))]
for patch, color in zip(bp['boxes'], colors):
    patch.set(facecolor=color)
# Встановлюємо кольори для викидів в залежності від стовпця продажів
outlier_colors = [plt.cm.plasma((i+1) / (len(sales_columns)+1)) for i in range(len(sales_columns))]
for i, flier in enumerate(bp['fliers']):
        flier.set(markerfacecolor=outlier_colors[i], marker='o', alpha=0.5)
```

Кореляція:

```
# Обчислюємо кореляційну матрицю
correlation_matrix = vgs.corr(numeric_only=True)
# Створюємо маску для скриття верхньої трикутної частини матриці
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
# Налаштовуємо властивості графіку
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, mask=mask)
```

Тепер перейдемо до візуального аналізу. Побудова декількох лінійних графіків для зображення загальних продажів за роки та к-ть випущених ігор на різних платформах (оберемо більш значущі, випуск не менше 100 ігор):

```
    # Групуємо дані за платформами та роками і підраховуємо кількість ігор
    platform_sales = vgs.groupby('Platform')['Year'].value_counts().unstack(fill_value=0)
    # Відбираємо лише ті платформи, які мають загальну кількість випущених ігор більше 100
    platform_sales = platform_sales.loc[platform_sales.sum(axis=1) >= 100]
    # Визначаємо кількість підграфіків для розміщення
    num_platforms = len(platform_sales)
    num_subplots = int(np.ceil(num_platforms / 5))
    # Створюємо фігуру та відповідну кількість підграфіків
    fig, axes = plt.subplots(num_subplots, 1, figsize=(10, 6*num_subplots))
    # Ітеруємося по підграфіках
    for i, ax in enumerate(axes):
        start = i * 5
        end = start + 5
        # Вибираємо підмножину платформ для поточного підграфіка
        platform_subset = platform_sales.iloc[start:end]
        # Генеруємо кольори для кожної платформи
            colors = plt.colormaps['plasma'](np.linspace(0, 1, len(platform_subset.index)))
         # Малюємо лінії та заповнюємо простір між ними для кожної платформи
                    for j, (platform, color) in enumerate(zip(platform_subset.index, colors)):
                                        ax.plot(platform_subset.columns, platform_subset.loc[platform], linestyle='-', color=color)
                                    ax.fill_between(platform_subset.columns, platform_subset.loc[platform], color=color, alpha=0.3)
```

Так як даних досить багато, для візуалізації продажів за кожен рік можна використати теплові карти (heatmap).

Яка частка ігор випущена видавцями:

```
    #Створення сводної таблиці для підрахунку загальних продажів за роком та видавництвом
    table_sales = pd.pivot_table(vgs, values=['Global_Sales'], index=['Year'], columns=['Publisher'], aggfunc='sum', margins=False)
    #Вибір топ-10 видавництв за продажами
    top_10_publishers = table_sales['Global_Sales'].sum().sort_values(ascending=False)[:10]
    #Створення таблиці з продажами топ-10 видавництв
    table_top_publishers = table_sales['Global_Sales'][top_10_publishers.index]
```

Також необхідно порівняти продажі у різних регіонах (за жанром та видавцем):

```
    # Список регіонів для візуалізації
    regions = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales',
'Global_Sales']
    # Створення сітки графіків
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))
    axes = axes.flatten()
    # Прохід по кожному регіону
    for i, region in enumerate(regions):
        # Групування продажів за жанром
                                        genre_sales                =
vgs.groupby('Genre')[region].sum().reset_index()
        # Побудова графіку стовпчиків
             sns.barplot(x='Genre',   y=region,   data=genre_sales,
ax=axes[i], palette='viridis')
        axes[i].set_xlabel('', fontsize=0)
        axes[i].set_ylabel('', fontsize=0)
        axes[i].set_title('{}'.format(region), fontsize=12)
          axes[i].set_xticklabels(genre_sales['Genre'], rotation=90,
fontsize=8)
        # Форматування осі Y для відображення значень з роздільниками
тисяч
         axes[i].yaxis.set_major_formatter(ticker.StrMethodFormatter(
'{x:,.0f}'))
        plt.xticks(rotation=90, fontsize=8)
```

Щоб дізнатися частка яких даних у різних категоріях переважає використовуємо кругові діаграми (обираємо значення % яких не менше 1):

```
    # Обчислюємо кількість ігор кожного видавця
    publisher_counts = vgs['Publisher'].value_counts()
    # Фільтруємо видавців, які складають менше 1% від загальної
кількості
    publisher_counts    =    publisher_counts[publisher_counts    /
publisher_counts.sum() >= 0.01]
    # Задаємо кольори для секторів графіка
    colors = sns.color_palette('coolwarm', len(publisher_counts))
    # Задаємо значення "викиду" для секторів
    explode = [0.1] * len(publisher_counts)
    # Задаємо властивості секторів
    wedgeprops = {'linewidth': 1, 'edgecolor': 'white'}
    # Розміщуємо сектори на графіку
    patches,    _    =    plt.pie(publisher_counts,    colors=colors,
explode=explode, startangle=90, wedgeprops=wedgeprops)
    # Задаємо однакові масштаби осей для отримання кругової форми
графіку
    plt.axis('equal')
    # Збираємо підписи для легенди
    total = sum(publisher_counts)
    percentages = [(count / total) * 100 for count in publisher_counts]
    # Фільтруємо значення відсотків менше 1%
    labels = ['{0} - {1:1.1f}%'.format(label, percentage) for label,
percentage in zip(publisher_counts.index, percentages) if percentage >=
1]
```

Графік розсіювання:

```
    # Побудова діаграми розсіювання за допомогою функції scatterplot з використанням даних зі змінними vgs
    sns.scatterplot(data=vgs, x='Year', y='Platform', hue='Platform', palette=palette, s=100, alpha=0.7)
```

Відсоток ігор різних жанрів випущених однією студією:

```
    sales_by_publisher_genre = filtered_data.groupby(['Publisher', 'Genre'])['Global_Sales'].sum().reset_index() # Обчислюємо загальні продажі по жанрах для кожного видавця
    sales_by_publisher_genre['Percentage'] = (sales_by_publisher_genre['Global_Sales'] / sales_by_publisher_genre.groupby('Publisher')['Global_Sales'].transform('sum')) # Обчислюємо відсоток продажів по жанрах для кожного видавця
    heatmap_data = pd.pivot_table(sales_by_publisher_genre, values='Percentage', index='Publisher', columns='Genre', fill_value=0) # Створюємо таблицю для побудови теплової карти
```

За допомогою метода Apriori визначаємо пари елементів, що зустрічаються у вибірці найчастіше:

```
    # Конвертуємо категоріальні ознаки в рядки
    vgs[categorical_features] = vgs[categorical_features].astype(str)
    # Створюємо список транзакцій
    transactions = vgs[categorical_features].values.tolist()
    # Ініціалізуємо об'єкт TransactionEncoder
    te = TransactionEncoder()
    # Застосовуємо TransactionEncoder до списку транзакцій і отримуємо бінарну матрицю
    te_ary = te.fit_transform(transactions)
    # Створюємо DataFrame з бінарною матрицею і назвами колонок
    df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
    min_support = 0.01
    # Знаходимо часті набори за алгоритмом Apriori
    frequent_itemsets = apriori(df_encoded, min_support=min_support, use_colnames=True)
    # Додаємо колонку з довжиною набору
    frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
    # Фільтруємо набори за довжиною = 2
    frequent_itemsets = frequent_itemsets[frequent_itemsets['length'] == 2]
```

І візуалізуємо результат:

```
    # Побудова горизонтальної стовпчастої діаграми
    plt.figure(figsize=(10, 6))
    plt.barh(range(len(frequent_itemsets2)),
frequent_itemsets2['support'],  tick_label=[',  '.join(itemset)  for
itemset in frequent_itemsets2['itemsets']], color=colors)
```

Реалізуємо метод кластерного аналізу K-means:

```
    from sklearn.cluster import KMeans
    # Визначаємо ознаки, які будемо використовувати для кластеризації
    features = ['NA_Sales', 'EU_Sales', 'Global_Sales']
    X_encoded = df_filtered[features]
    # Визначаємо кількість кластерів, які хочемо сформувати
    k = 5
    kmeans = KMeans(n_clusters=k, n_init='auto', random_state=42)
    # Виконуємо кластеризацію з використанням K-means
    kmeans.fit(X_encoded)
    # Отримуємо мітки кластерів для кожного зразка
    labels = kmeans.predict(X_encoded)
    # Додаємо колонку 'Cluster' до вихідного DataFrame з отриманими
мітками кластерів
    df_filtered['Cluster'] = labels
    print(labels)
    print(df_filtered[['NA_Sales',    'EU_Sales',    'Global_Sales',
'Cluster']])
```

Візуалізуємо за допомогою бібліотеки PCA:

```
    # Ініціалізуємо об'єкт PCA з двома компонентами
    pca = PCA(n_components=2)

    # Застосовуємо PCA до даних для зменшення розмірності до 2D
    X_pca = pca.fit_transform(X_encoded)

    # Візуалізація результатів на графіку розсіювання
    plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis')
    plt.xlabel('PCA Dimension 1')
    plt.ylabel('PCA Dimension 2')
    plt.title('Cluster Visualization using PCA')
    plt.colorbar()
    plt.show()
```

Метод agg для проведення аналізу групових статистик, таких як середнє значення, медіана:

```
numeric_columns = ['NA_Sales', 'EU_Sales', 'Global_Sales']
grouped_stats = df_filtered.groupby('Cluster')[numeric_columns].agg(['mean', 'median'])
# Виведення обчислених статистичних показників
print(grouped_stats)
```

А також подальша візуалізація результату:

```
# Створення фігури та двох підграфіків
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10), sharex=True)
clusters = grouped_stats.index
features = ['Other_Sales', 'Global_Sales']
colors = ['blue', 'orange']
x_pos = np.arange(len(clusters))
bar_width = 0.1
opacity = 0.7
# Побудова графіку середніх значень
for i, feature in enumerate(features):
    values_mean = grouped_stats[(feature, 'mean')].values.flatten()
    ax1.bar(x_pos + i * bar_width, values_mean, color=colors[i], width=bar_width, label=feature, alpha=opacity)
# Побудова графіку медіанних значень
for i, feature in enumerate(features):
    values_median = grouped_stats[(feature, 'median')].values.flatten()
    ax2.bar(x_pos + i * bar_width, values_median, color=colors[i], width=bar_width, label=feature, alpha=opacity)
```

Дані про кластери :

```
for a in range(5):
    cluster_data = df_filtered[df_filtered['Cluster'] == a]
    most_common_values = cluster_data[features].mode().iloc[0]
    max_sales = cluster_data['Global_Sales'].max()
```

Модель RandomForestRegressor для передбачення змінної 'Global_Sales' на основі наявних даних:

```
sales_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales','Global_Sales','Rank']
features = [col for col in df_filtered.columns if col not in sales_columns]
target = 'Global_Sales'
X = df_filtered[features]
y = df_filtered[target]
```

```
    # Ініціалізація LabelEncoder для перетворення категоріальних ознак на числові значення
    label_encoder = LabelEncoder()
    X_encoded = X.copy()
    for feature in features:
        # Перетворення категоріальної ознаки на числове значення
        X_encoded[feature] = label_encoder.fit_transform(X[feature])
    # Розділення даних на тренувальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.1, random_state=1000)
    # Ініціалізація моделі RandomForestRegressor з 200 деревами
    model = RandomForestRegressor(n_estimators=200, random_state=300)
    model.fit(X_train, y_train)
    # Прогнозування значень на тестовому наборі
    y_pred = model.predict(X_test)
    # Обчислення точності моделі
    accuracy = model.score(X_test, y_test)
    print(f'Accuracy: {accuracy}')
```

Візуалізація передбачень:

```
    plt.figure(figsize=(10, 6))
    plt.scatter(range(len(y_test)), y_test, color='blue', label='Фактичні', alpha=0.8)  # Графік фактичних значень
    plt.scatter(range(len(y_test)), y_pred, color='red', label='Прогнозовані', alpha=0.4)  # Графік прогнозованих значень
```

Висновки до другого розділу

У другому розділі було надано повну характеристику джерела даних. Далі побудовано модель аналізу та реалізовано її на практиці з використанням обраного програмного забезпечення. Як результат отримано усі необхідні складові для візуального представлення даних та підведення підсумку.

РОЗДІЛ 3. АНАЛІЗ РЕЗУЛЬТАТІВ, ПІДСУМКОВІ ГРАФІКИ ТА ТАБЛИЦІ

3.1 Результати проведеного аналізу в таблицях діаграмах і графіках

Щоб підібрати підхід до аналізу для початку було вирішено розглянути структуру даних та як вони розподілені у наборі. Перевіривши набор на наявність нульових значень, бачимо що є деякі проблеми. А саме пусті значення у роках та розробниках. Щоб вирішити це видаляємо значення без розробника та замість пустих років ставимо середнє значення.

```
Rank            0
Name            0
Platform        0
Year          271
Genre           0
Publisher      58
NA_Sales        0
EU_Sales        0
JP_Sales        0
Other_Sales     0
Global_Sales    0
dtype: int64
0
                Rank          Year      NA_Sales      EU_Sales      JP_Sales  \
count   16598.000000  16327.000000  16598.000000  16598.000000  16598.000000
mean     8300.605254   2006.406443      0.264667      0.146652      0.077782
std      4791.853933      5.828981      0.816683      0.505351      0.309291
min         1.000000   1980.000000      0.000000      0.000000      0.000000
25%      4151.250000   2003.000000      0.000000      0.000000      0.000000
50%      8300.500000   2007.000000      0.080000      0.020000      0.000000
75%     12449.750000   2010.000000      0.240000      0.110000      0.040000
max     16600.000000   2020.000000     41.490000     29.020000     10.220000

        Other_Sales  Global_Sales
count  16598.000000  16598.000000
mean       0.048063      0.537441
std        0.188588      1.555028
min        0.000000      0.010000
25%        0.000000      0.060000
50%        0.010000      0.170000
75%        0.040000      0.470000
max       10.570000     82.740000
```

Рис.3.1. Основна інформація про дані.

Що до розподілу, його зробили за допомогою KDE (вона ж ядерна оцінка щільності). На отриманому графіку можна побачити, що дані мають скос вправо (right-skewed distribution). Вони асиметричні та мають більш широкий діапазон менших значень. Різке зростання означає, що є кілька дуже популярних продуктів, які складають велику частку глобальних продажів.

Розтягнені значення, в свою чергу, вказують на наявність продуктів з меншим внеском у продаж, але вони все ще значною кількістю.

Це відображає різні ринкові умови. Несподіваний успіх або зростання популярності певних продуктів з часом, також жанрів, платформ, або інших факторів, що впливають на продажі.

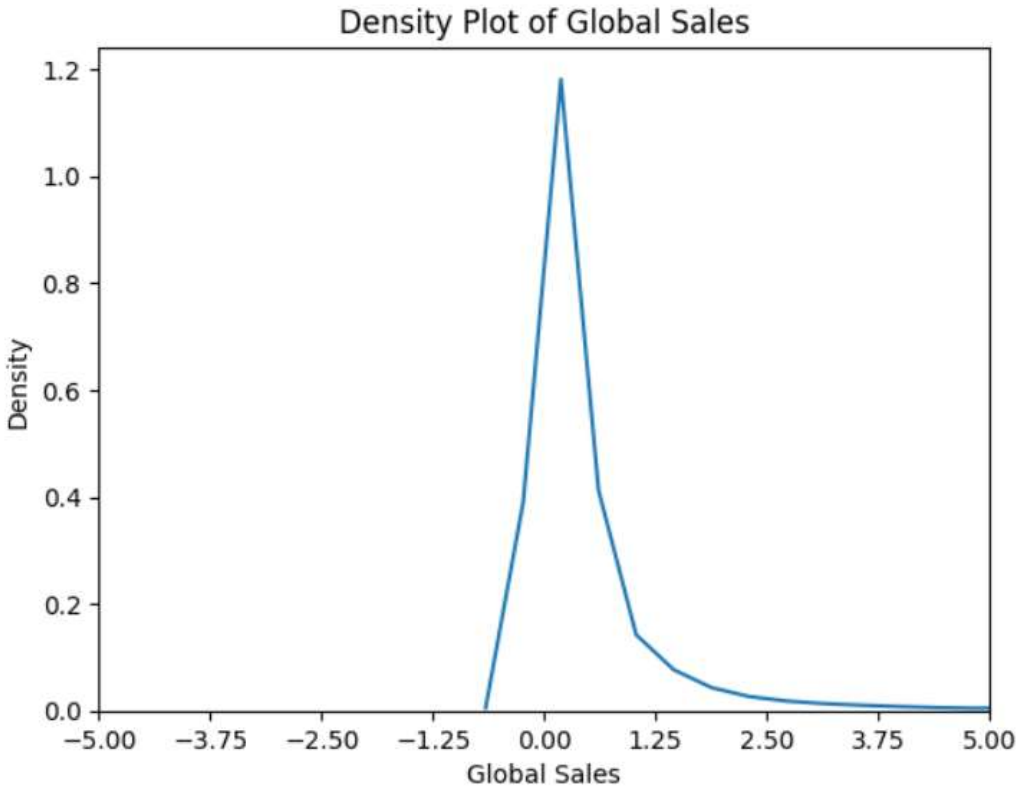


Рис.3.2. Ядерна оцінка щільності для стовпця "Global_Sales".

Так як відомо, що є досить популярні продукти зі значеннями відмінними від інших, є сенс переглянути дані на наявність викидів. Для цього використаємо «коробку з вусами» (box plot). На отриманому графіку можемо побачити, що к-ть аномальних значень дуже висока. Настільки, що неможливо розглянути коробки з центральними та розділовими характеристиками (мінімальним/максимальним значенням, медіаною, міжквартильним діапазоном).

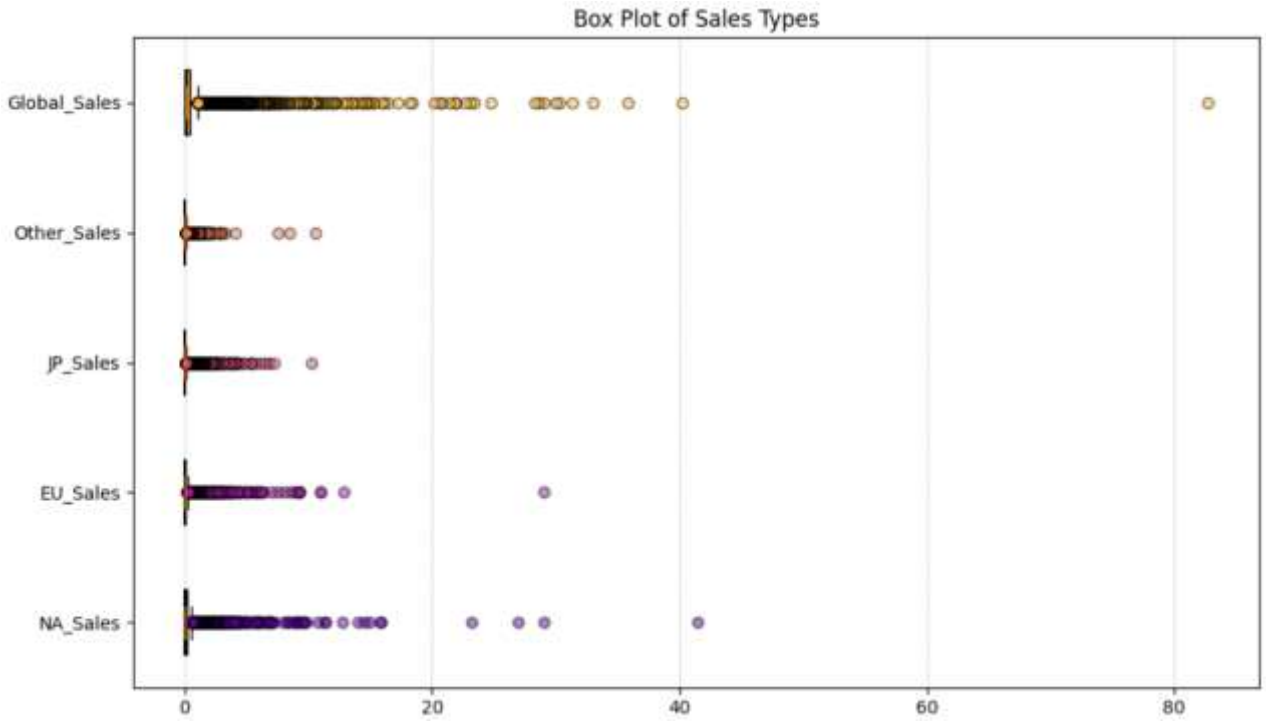


Рис.3.3. Викиди для усіх типів продажів.

Розрахувавши їх окремо, дійсно, бачимо, що к-ть викидів сягає 1900 значень з 15 тисяч.

```
Outliers:
0       82.74
1       40.24
2       35.82
3       33.00
4       31.37
        ...
1888     1.09
1889     1.09
1890     1.09
1891     1.09
1892     1.09
Name: Global_Sales, Length: 1893, dtype: float64
```

Рис.3.4. Викиди даних у стовпці "Global_Sales".

Як і показала ядерна оцінка щільності, це відбувається через дуже велику к-ть даних з малим значенням, що впливають на весь датасет.

```
Median Global Sales: 0.17
Mean Global Sales: 0.5374406555006628
```

Рис.3.5. Медіана та середнє значення.

Так як проводиться аналіз продажів недоречно видаляти або усереднювати значення продажів для отримання нормального розподілу. Так втрачаються важливі дані про «хіти» продажів, популярність та тенденції ринку. Тому продовжимо аналіз та перейдемо до розгляду залежності між парами змінних. Матриця кореляції надає інформацію про ступінь лінійної залежності числових значень. За її допомогою можна побачити досить логічні зв'язки між даними (рис.3.6).

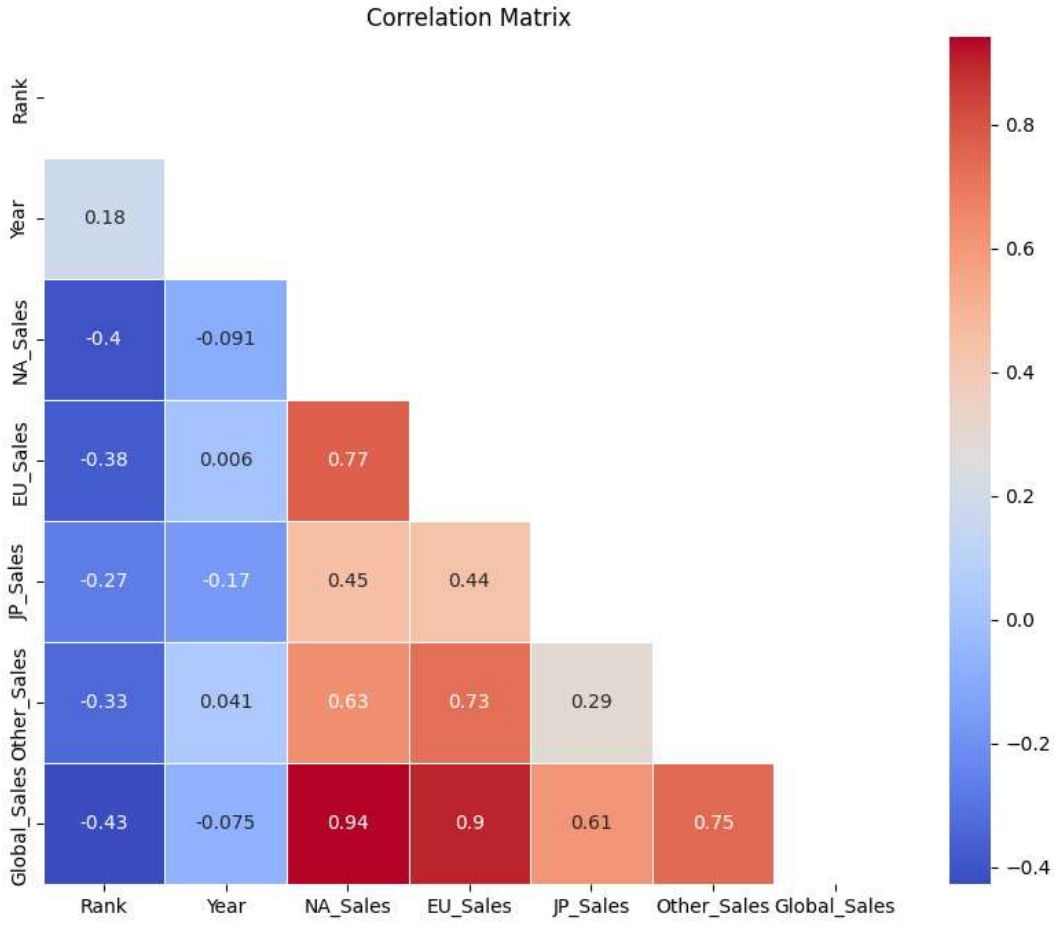


Рис.3.6. Матриця кореляції.

1. Чим вищий ранг гри у наборі – тим більші її продажі.
2. Чим вище усі типи продажів – тим більше глобальні.
3. Чим новіша гра – тим вище її ранг.
4. Більше всіх вплив на глобальні продажі має американський ринок.
5. Японські ігри з роками втрачають продажі більше за інших.

Тепер переглянемо загальну тенденцію прибутку від ігор на лінійному графіку (рис.3.7). З роками, хоча і не плавно, кошти збільшуються (спад у кінці наявний через відсутність даних з 2016 року). Як висновок ігрова індустрія набирає все більшу популярність починаючи з 2000-х років.

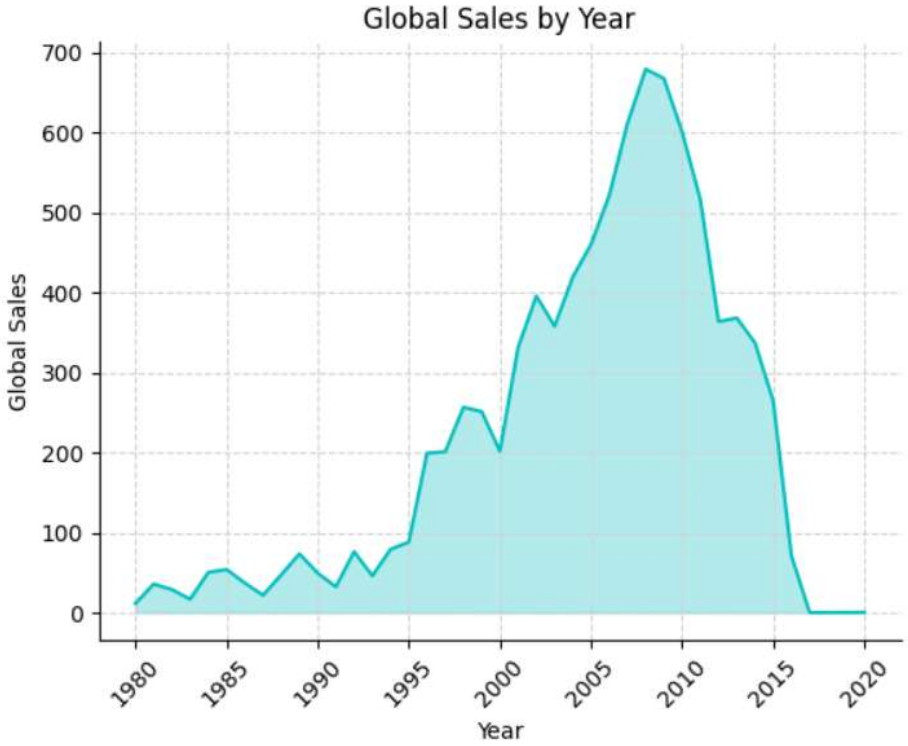
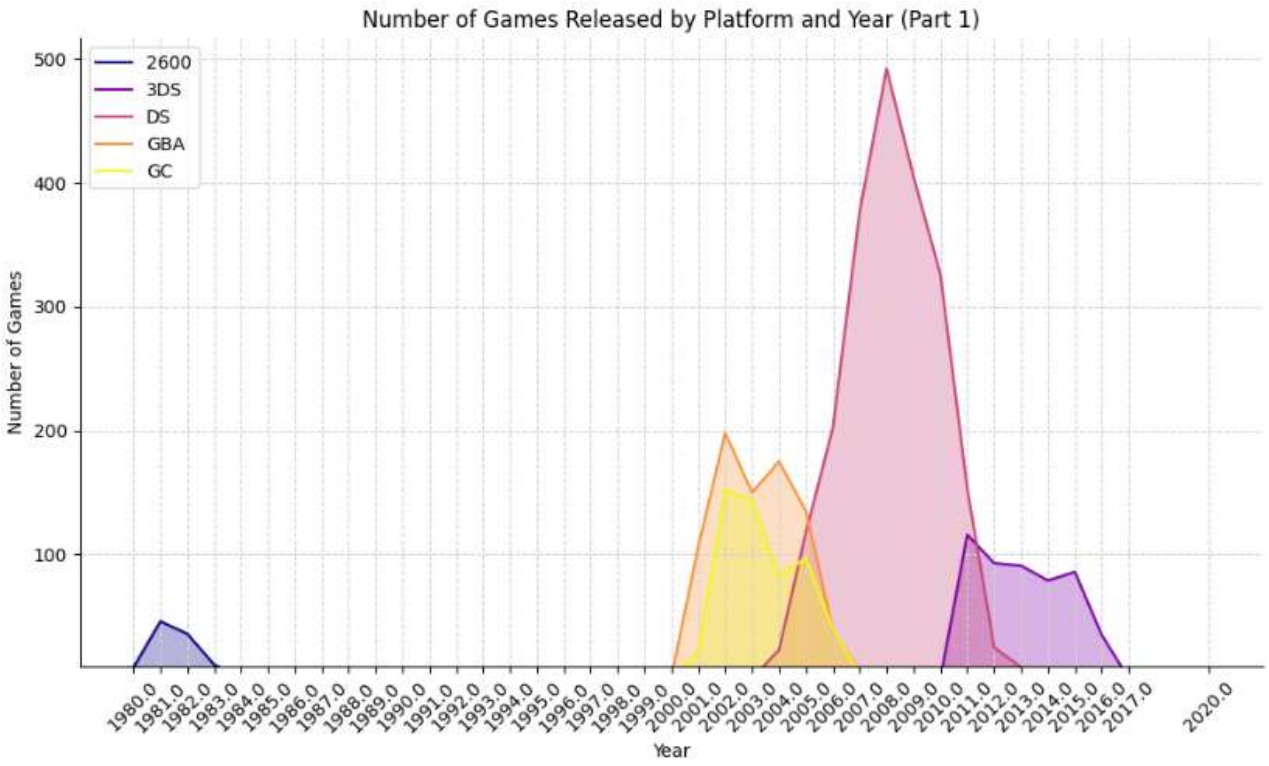


Рис.3.7. Продажі за рік (у мільйонах).

У датасеті наявно багато різних платформ. Тому було вирішено зобразити лише ті, що мають як мінімум 100 релізів. На усіх графіках (рис.3.8-3.9) спостерігається приріст виходу ігор на різні платформи з 1990-х років і основна популярність у 2005-2010. Станом на 2023 рік, можна сказати, що усі консолі наявні в датасеті – застарілі, тому для сучасних продажів не матимуть великої значущості.



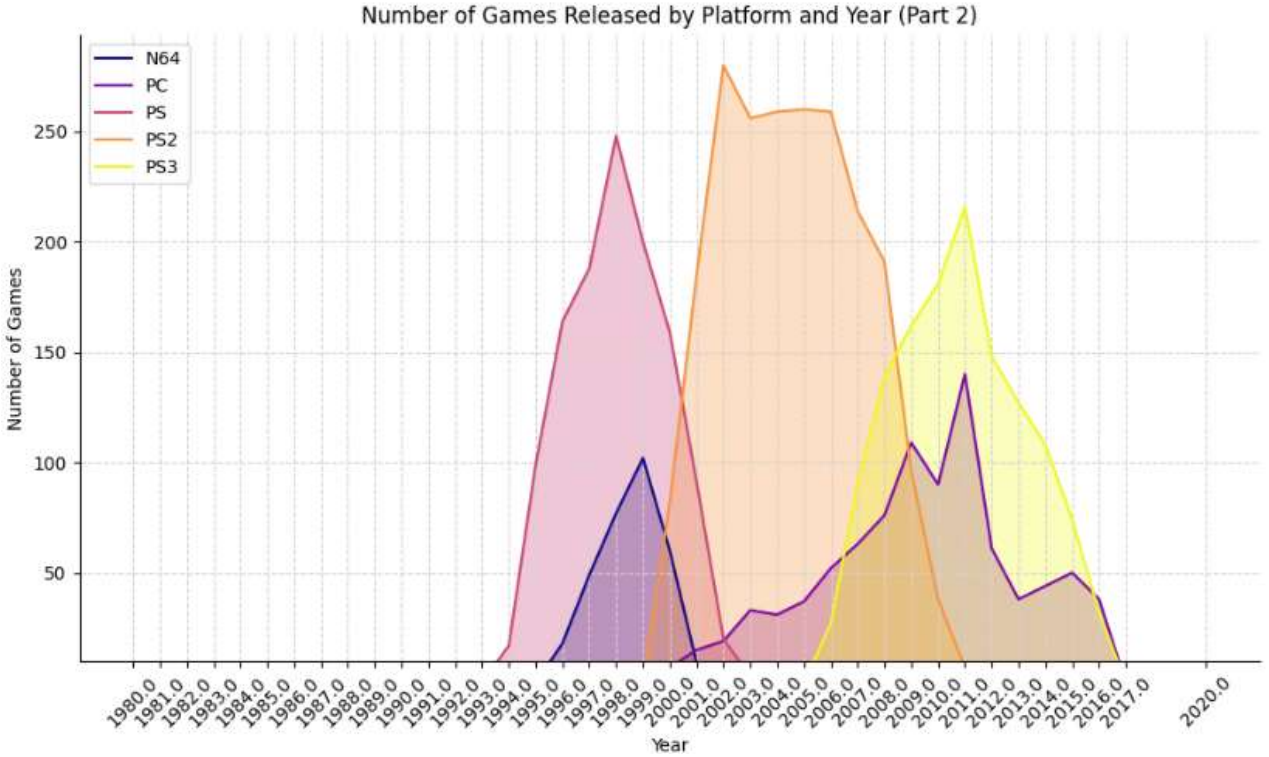
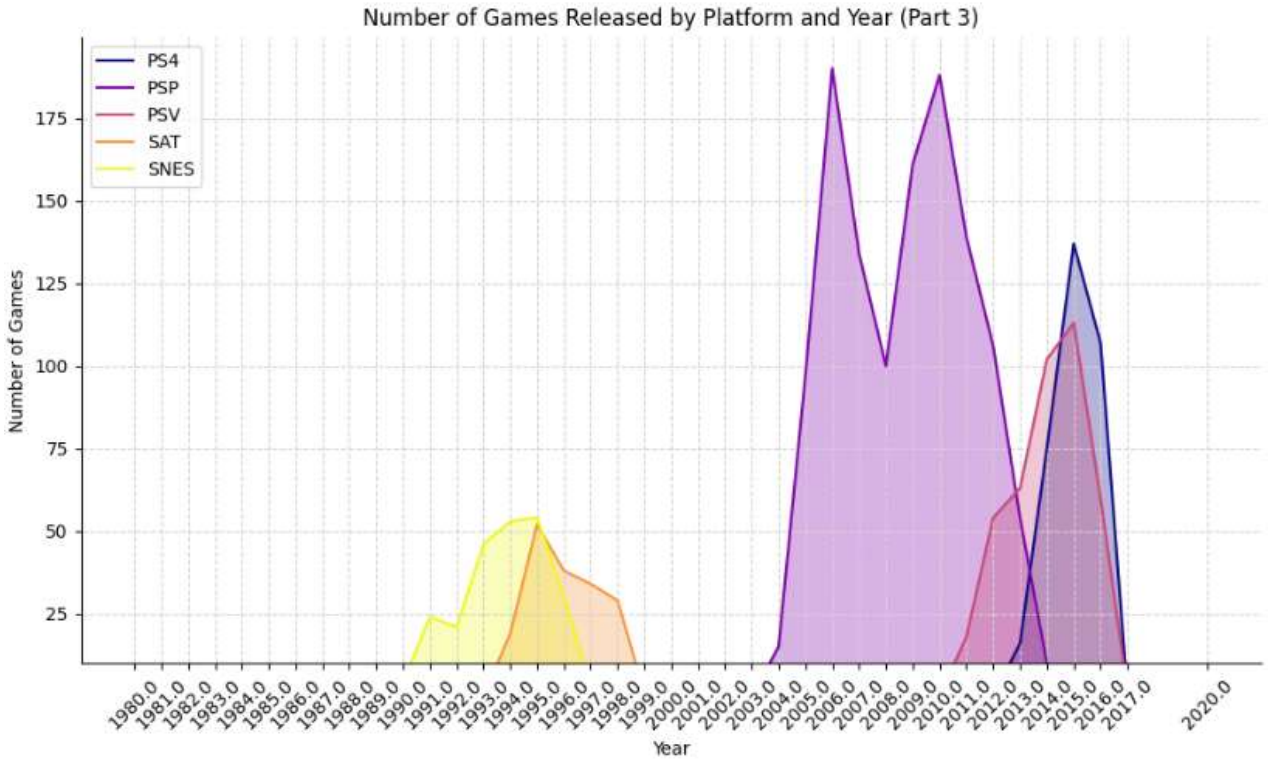


Рис.3.8. Продажі на різних платформах (частина1-2).



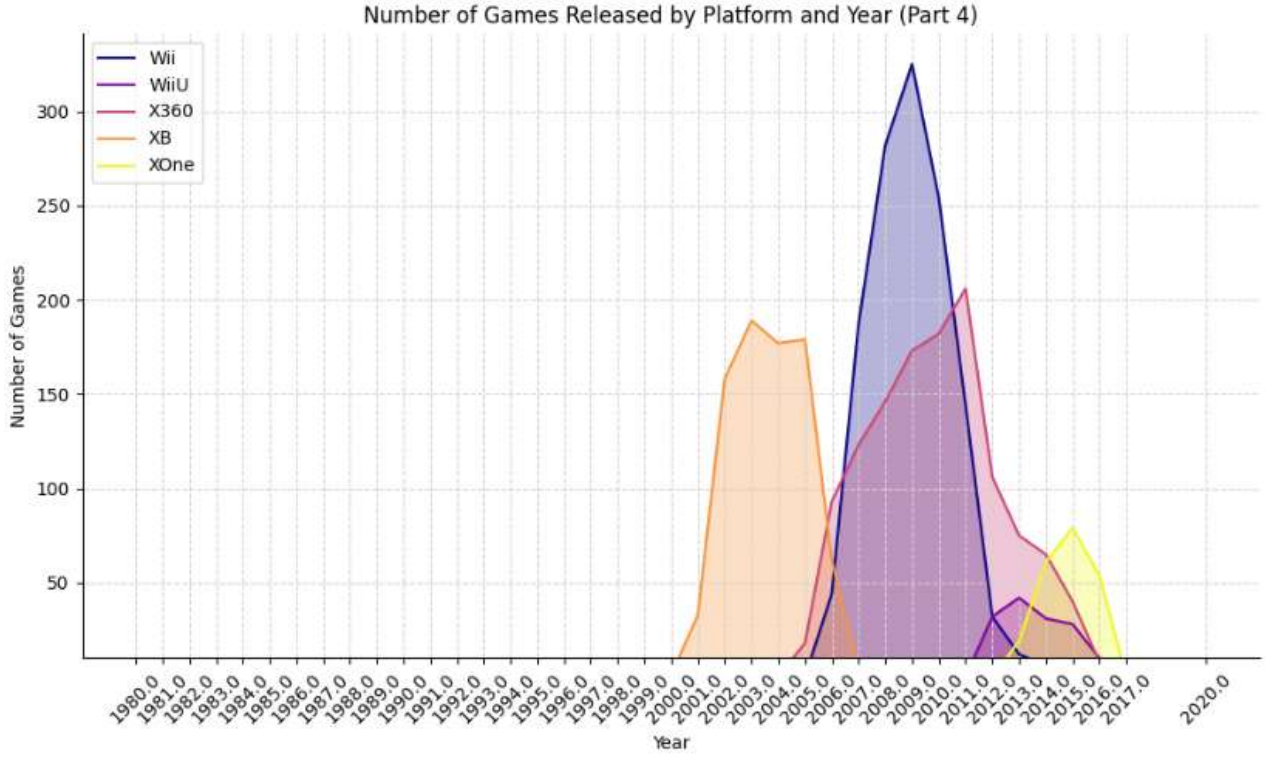


Рис.3.9. Продажі на різних платформах (частина3-4).

Що до жанрів, найдорожче продаються спортивні або багатокористувацькі ігри (рис.3.10). Також популярні платформери, «мікс». Найменшу зацікавленість користувачів викликає «стартегія».

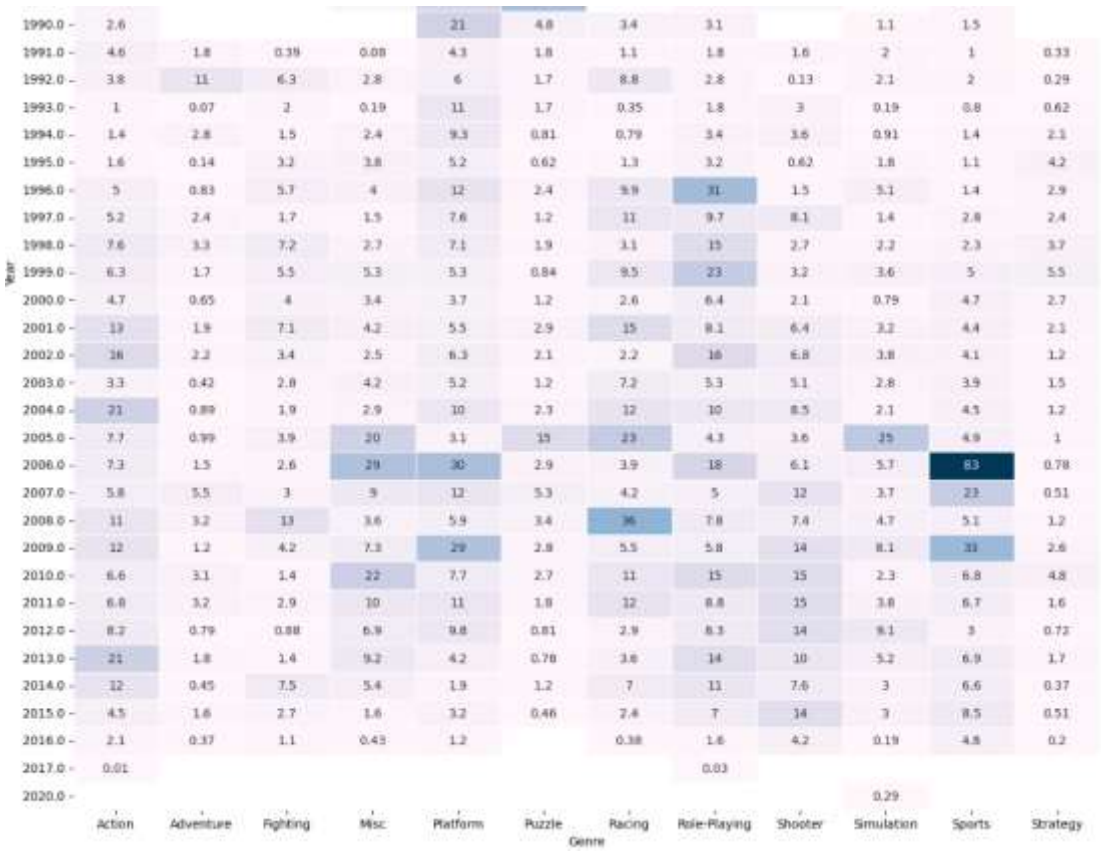


Рис.3.10. Найбільші продажі різних жанрів у рік (частково).

Також чітко видно приріст випуску продукції з роками.

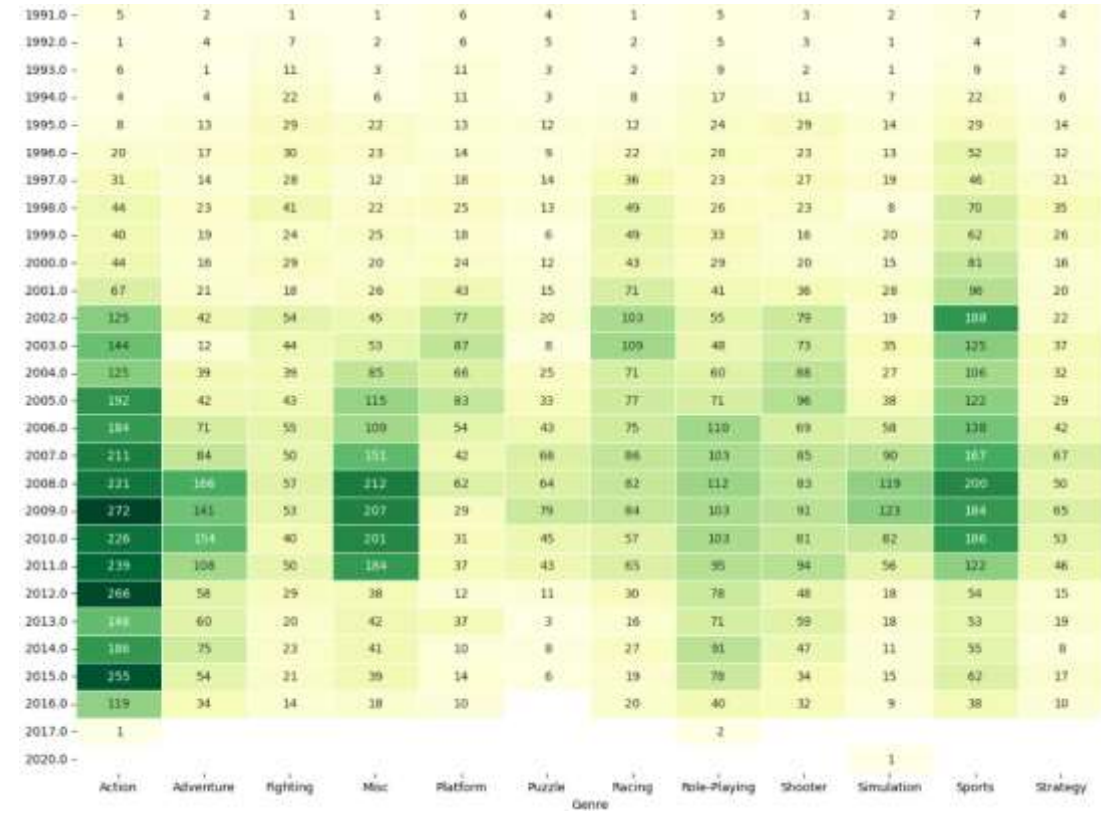


Рис.3.11. К-ть ігор різних жанрів у рік (частково).

Найактивнішими студіями в останні роки є: Nintendo, Electronic Arts, Activision Blizzard, Konami entertainment, Ubisoft. Усі ці компанії – гіганти розважального ринку, що мають достатньо коштів на щорічний випуск AAA проектів, або своїх консолей.

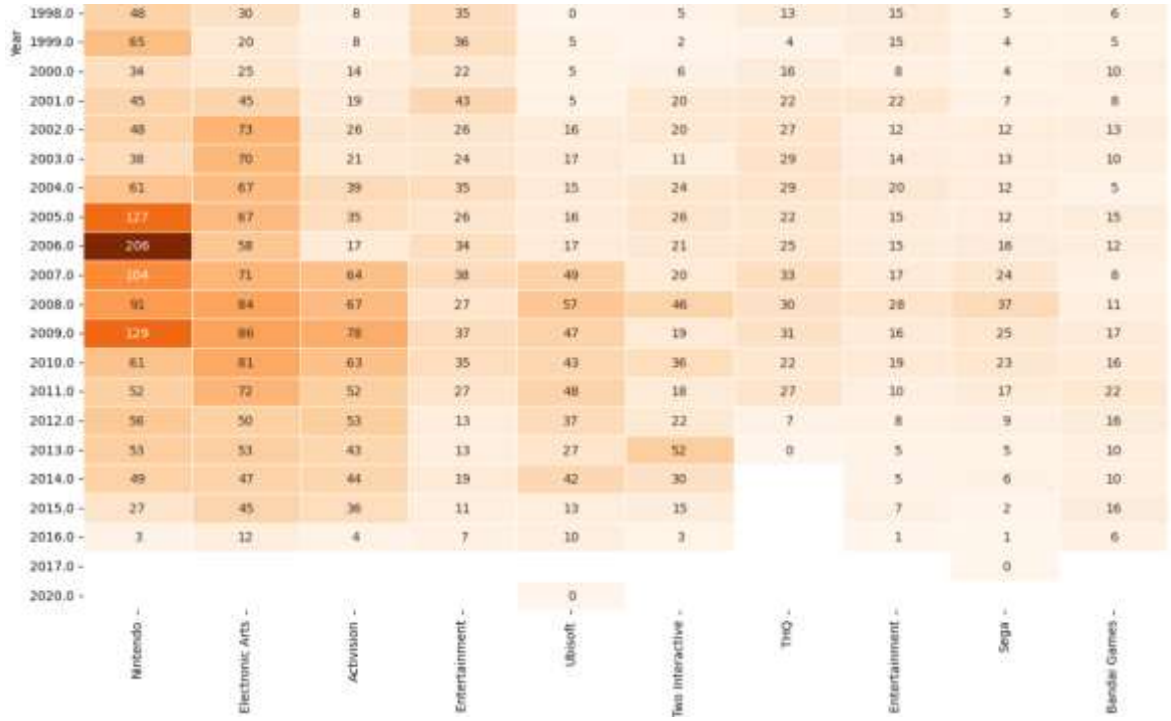


Рис.3.12. К-ть ігор різних студій у рік (частково).

Розглядаючи продажі, не дивно, що жанр, який найчастіше всього випускають останні роки – “action”. Це беззаперечний лідер у всіх країнах, окрім Японії. Також варто звернути увагу на спортивні ігри, шутери, котрі посідають друге і тертє міцся відповідно. Більшість таких ігор або мультиплеєрні або розраховані на взаємодію з іншими. Тож можна стверджувати, що користувачам веселіше грати разом.

Звісно, можна побачити, що це не стосується Японії. Вона також зробила великий внесок у ігрову індустрію та має багату історію, яка пішла іншою дорогою. Перше місце посідають рольові ігри. Даний жанр загалом має непагані продажі по світу і є досить популярним.

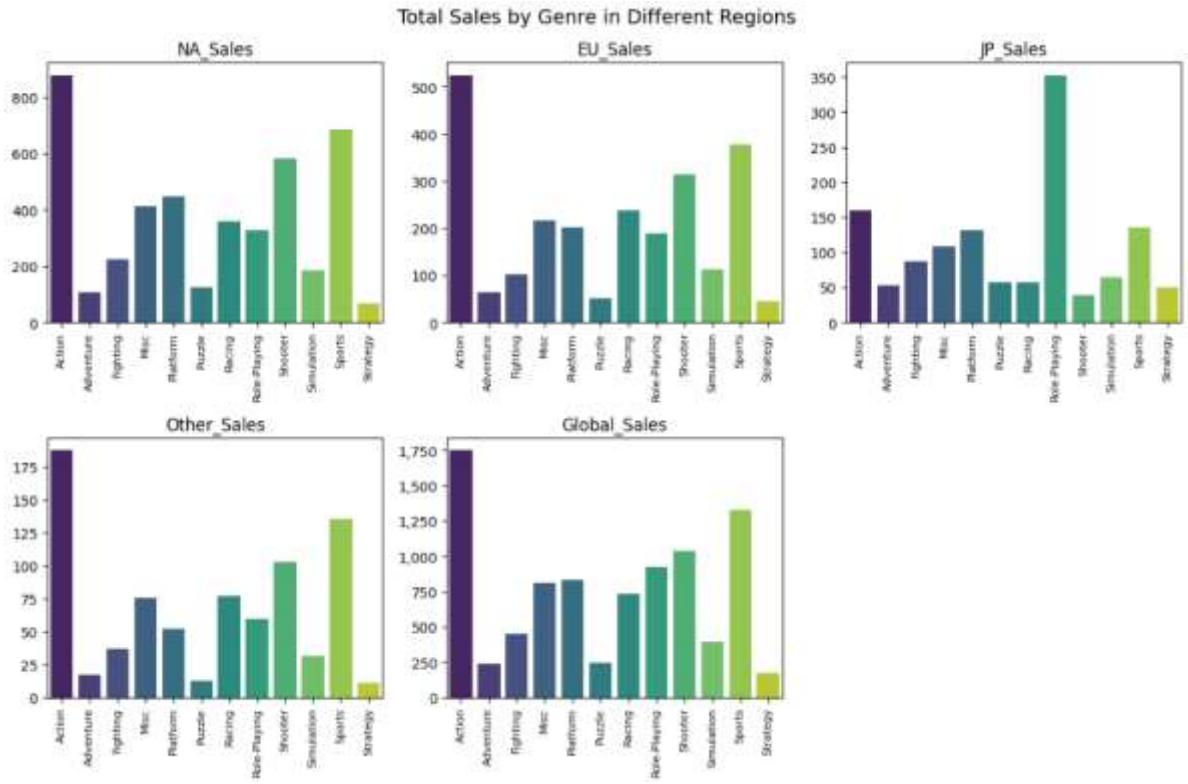


Рис.3.13. Продажі у різних регіонах (за жанром).

Що до розробників, найбагатшим серед усіх є Nintendo. Потрібно розуміти, що даний видавець не портує свої ігри на інші платформи. Тому необхідно спочатку придбати їх консоль, а потім і продукцію, за сталою ціною без знижок. Як можна побачити на початку 21 століття така стратегія давала гарні результати та принесла немало прибутку компанії.

Також показали себе такі відомі студії як Electronic Arts та Activision Blizzard. Перші зосереджені на створенні лінійок багатокористувацьких ігор, другі ж показали еталон якості та неординарності на початку нульових років, чим зробили собі ім’я та отримали славу.

Іншими відомими компаніями з великими продажами є Ubisoft і Sony Computer Entertaiment. Ubisoft також створюють франшизи, хоча мають дещо інший підхід ніж Electronic Arts. Sony Computer Entertaiment – це японська ігрова філія величезної компанії Sony, яка розробляє не лише ігри. А прикладає руку до створення всесвітньо відомих фільмів, музики та є крупним виробником техніки.

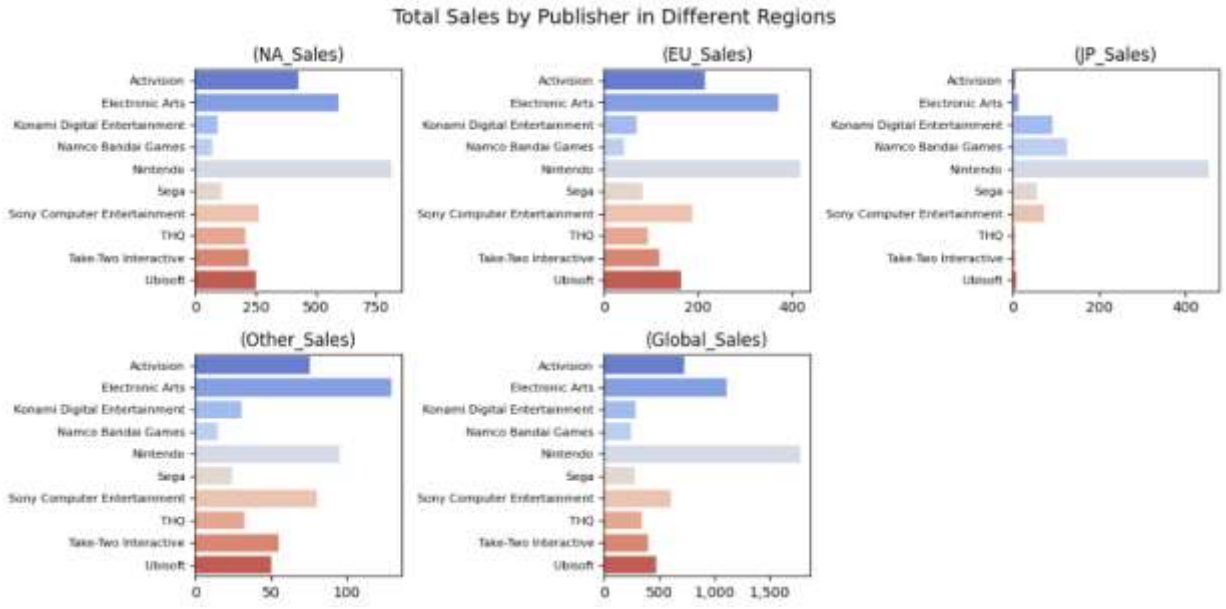


Рис.3.14. Продажі у різних регіонах (за видавцем).

Розглядаючи діаграми на рис.3.15. можемо побачити найбільші частки жанрів, платформ та видавців, що присутні у датасеті.

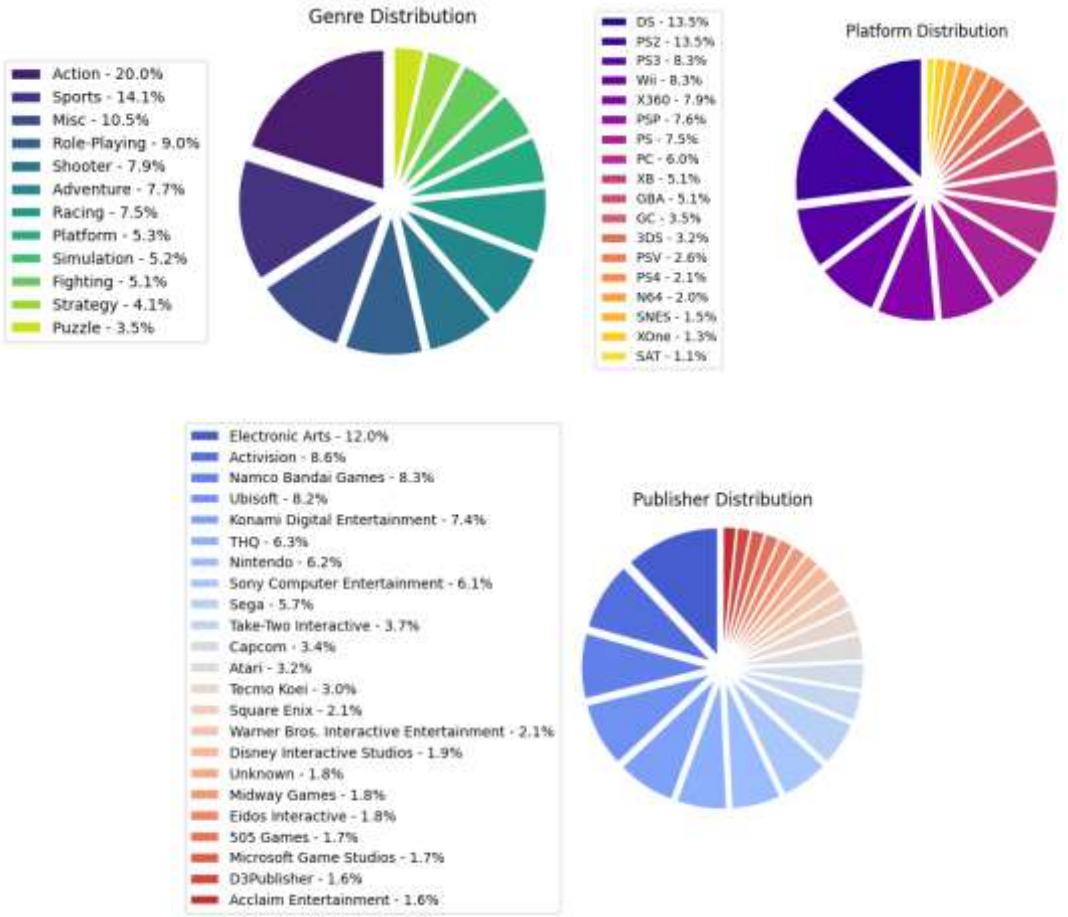


Рис.3.15. Частка різних жанрів, платформ, видавців від загальної.

Насправді платформ набагато більше аніж було зазначено, це підтверджує графік розсіювання на рис.3.16. Не усі вони популярні, а про більшість недостатньо нової інформації. Для прикладу ПК платформа. Раніше вона не згадувалась, тому що стала популярною лише в останні роки. У кінці 20 - початку 21 століття не усі могли дозволити собі мати персональний комп'ютер, тому і грали на портативних консолях, які наразі є лише в коллекціях або навіть музеях.

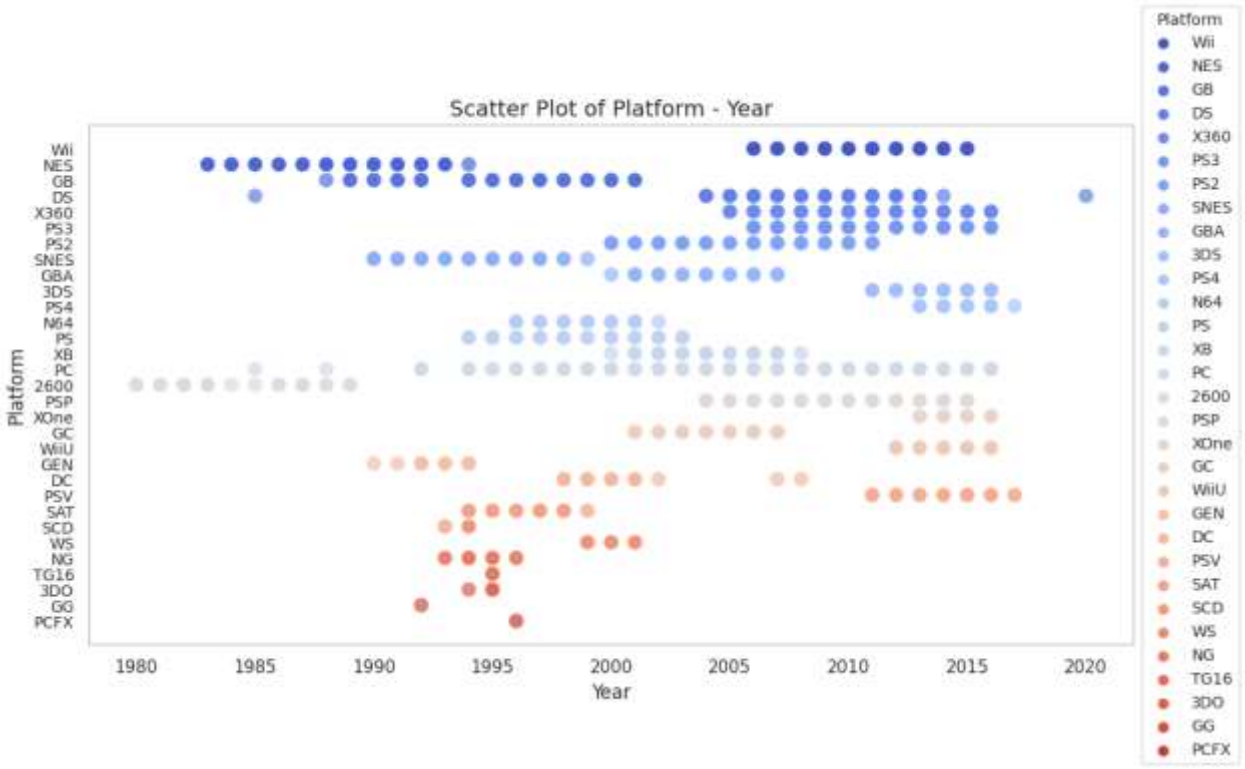


Рис.3.16. Графік розсіювання, платформи та рік виходу.

На наступному графіку (рис.3.17) зображено % ігор різних жанрів у найуспішніших компанії. Як не дивно більшість з них популярні: платформер, шутер, action, рольові, спортивні ігри. Також присутня значна частка ігор, які важко віднести до одного жанру.

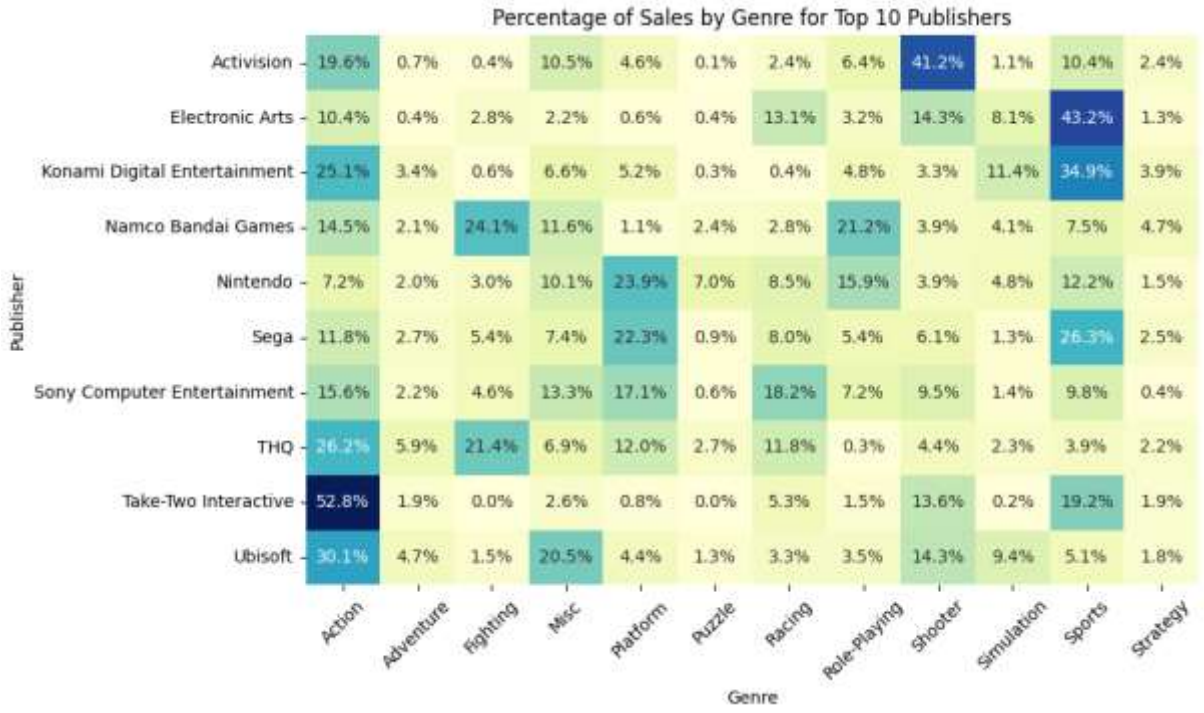


Рис.3.17. Жанри, які випускають компанії (у відсотках).

Раніше вже було проаналізовано більші частки значень датасету, але це було окремо для кожної категорії. Щоб переглянути які значення та пари значень зустрічаються нацчастіше, використовуємо асоціативний аналіз і візуалізуємо результат (рис.3.18- 3.19).

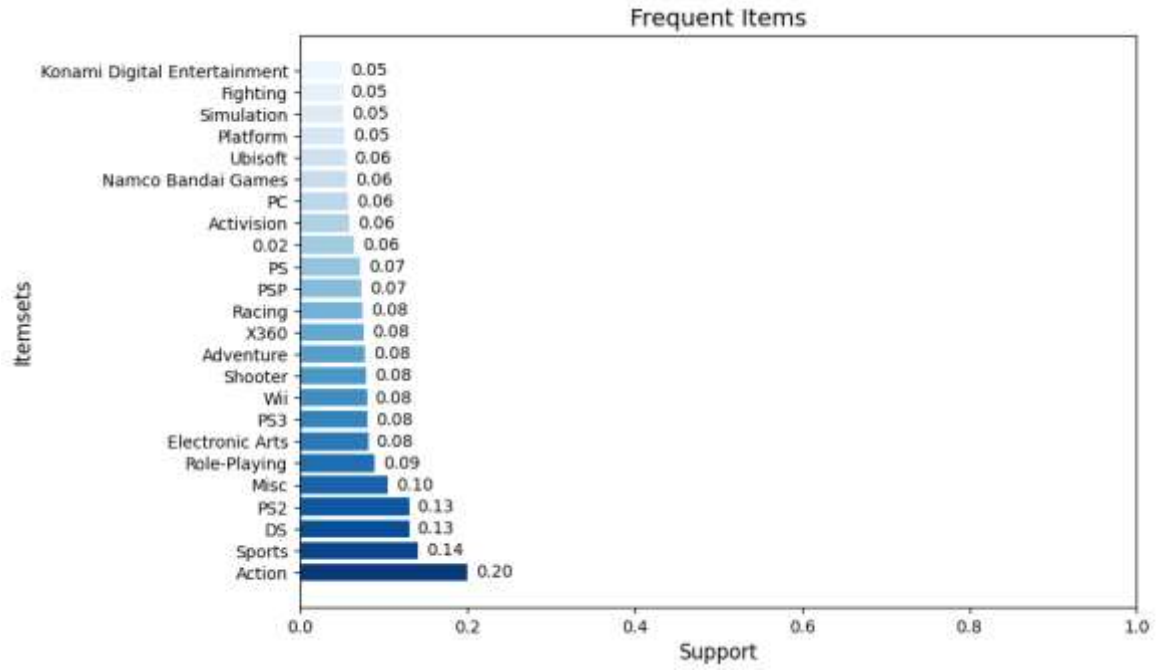


Рис.3.18. Частозустрічаємі елементи за результатом асоціативного аналізу.

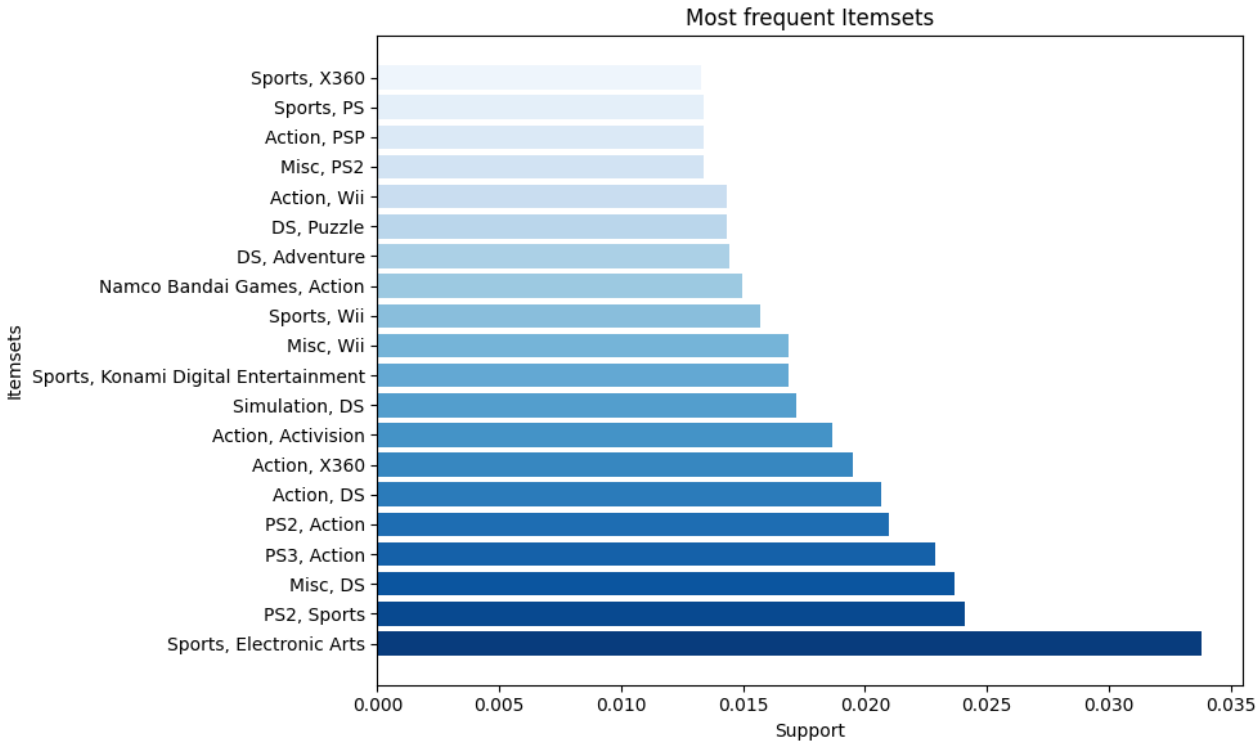


Рис.3.19. Частозустрічаємі пари елементів за результатом асоціативного аналізу.

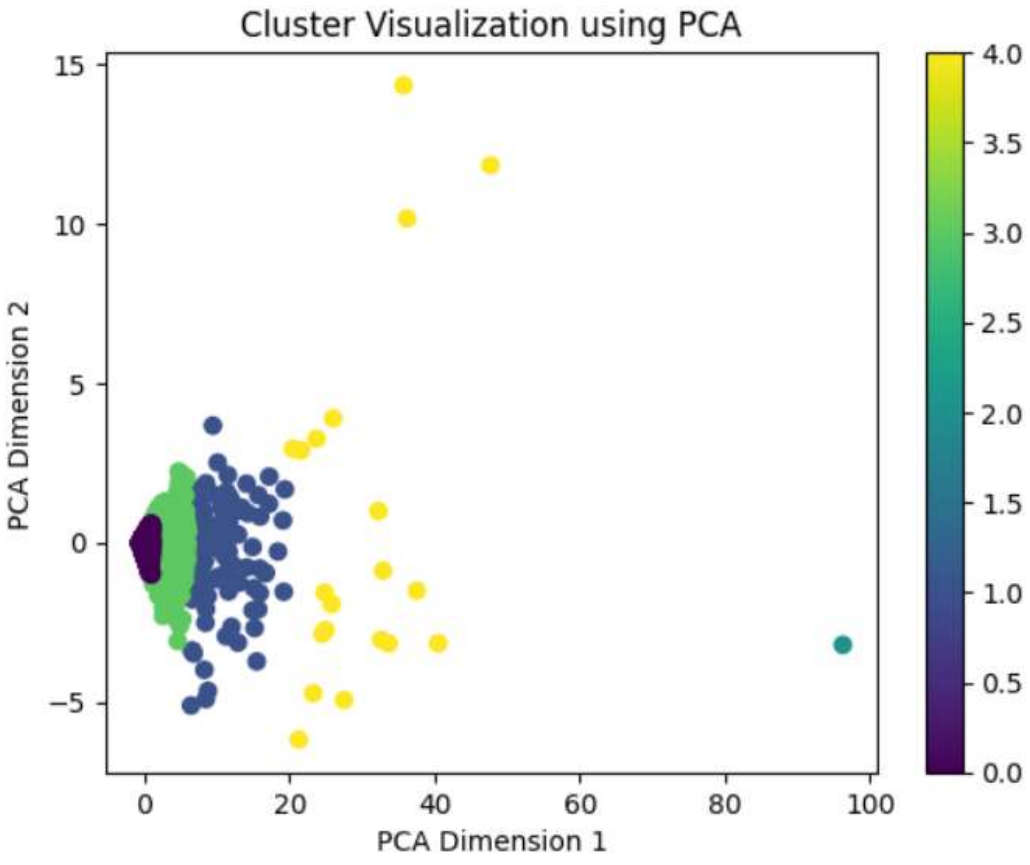


Рис.3.20. Візуалізація розподілу кластерів.

За результатами кластерного аналізу можна побачити, що починаючи від першого до останнього щільність даних зменшується. Останній кластер складається лише з одного значення. Це свідчить про дуже рідкісний випадок, коли гра має надзвичайно високі продажі порівняно з іншими. У першому кластері ж навпаки, маємо багато значень. І найбільше з них має всього 1.7 мільйона пробутку від глобальних продажів, тому можна зробити висновок що він включає ігри, які не здобули широкої популярності або не були комерційно успішними що підтверджує графік (рис.3.22). Інші кластери мають середні значення продажів у помірних межах. Вони можуть охоплювати ігри, які мають помірний комерційний успіх, спрямовані на певну аудиторію або нішевий ринок.

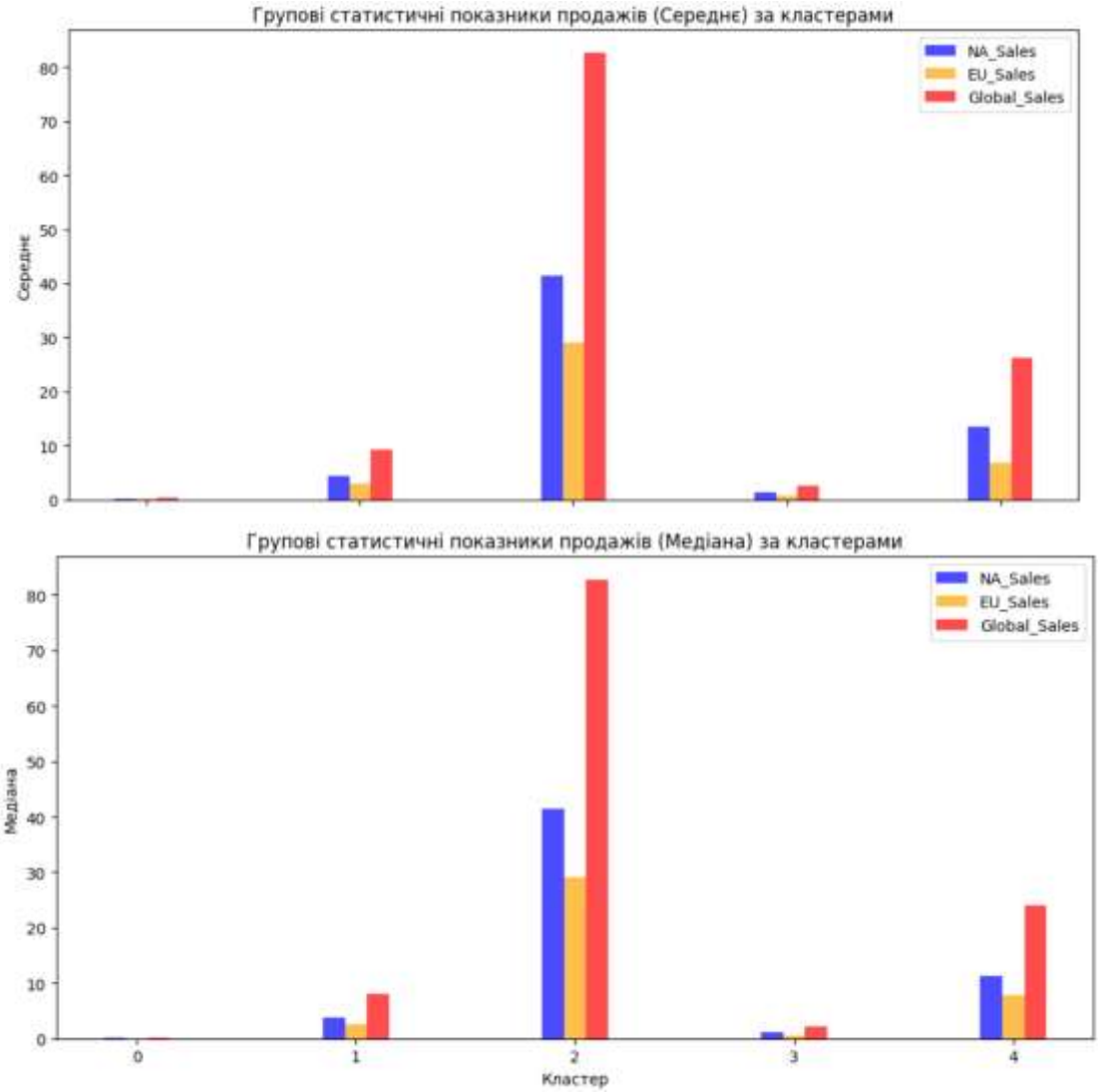


Рис.3.21. Середнє значення та медіана для продажів, розділених на кластери.

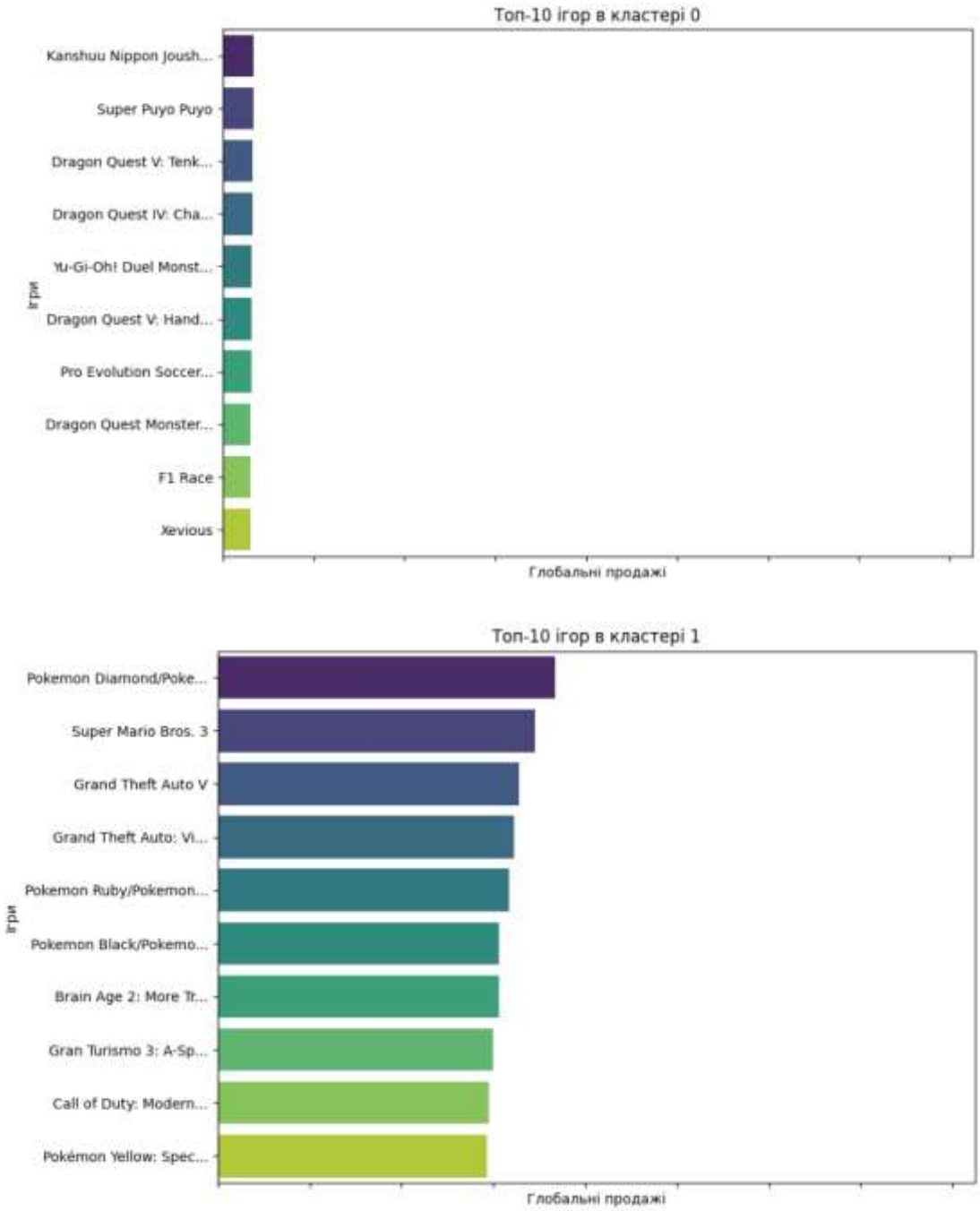


Рис.3.22. Порівняння продажів з різних кластерів.

Останнім візуалізоване передбачення за допомогою регресії. Це дуже корисно для визначення майбутніх продажів на основі отриманих. Хоча і присутньо багато характеристик ігор (дата виходу, платформа, видавник), все одно глобальні продажі передбачити не так просто. Існує багато зовнішніх факторів, що не включені у датасет. Тому отримана модель може описати лише половину. Хоча якщо на основі глобальних продажів передбачати продажі у різних країнах – результат чудовий (90%).

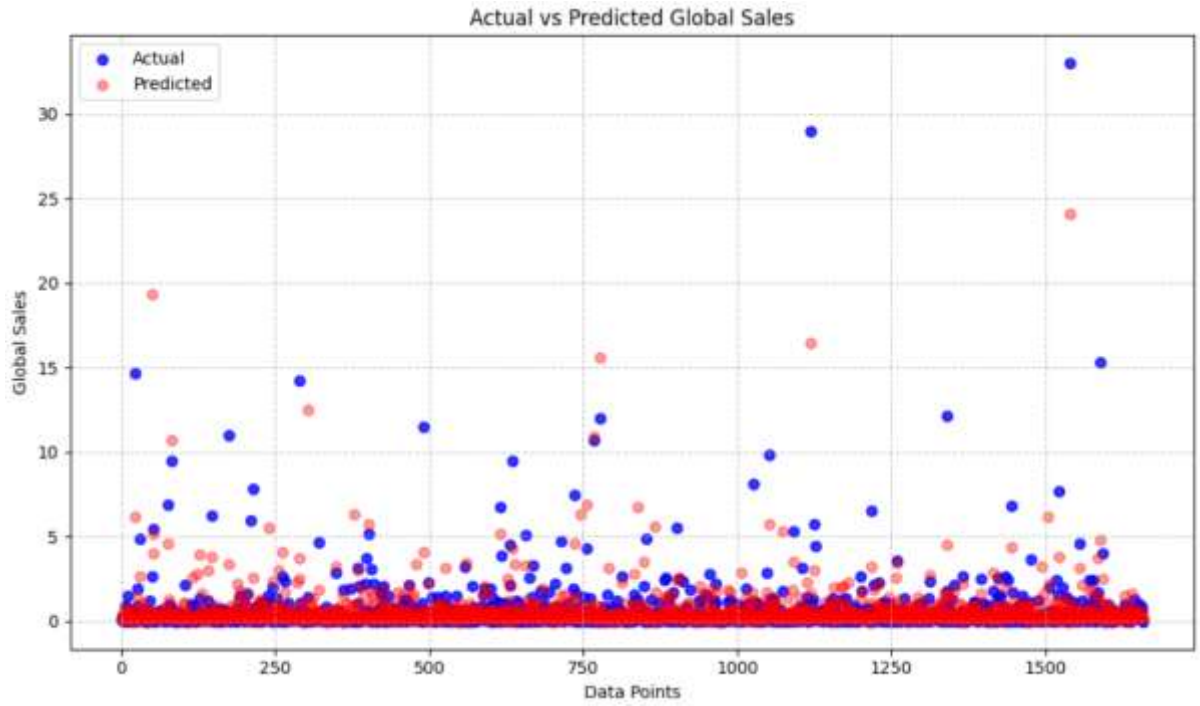


Рис.3.23. Передбачені дані Global_sales за допомогою моделі та справжні значення.

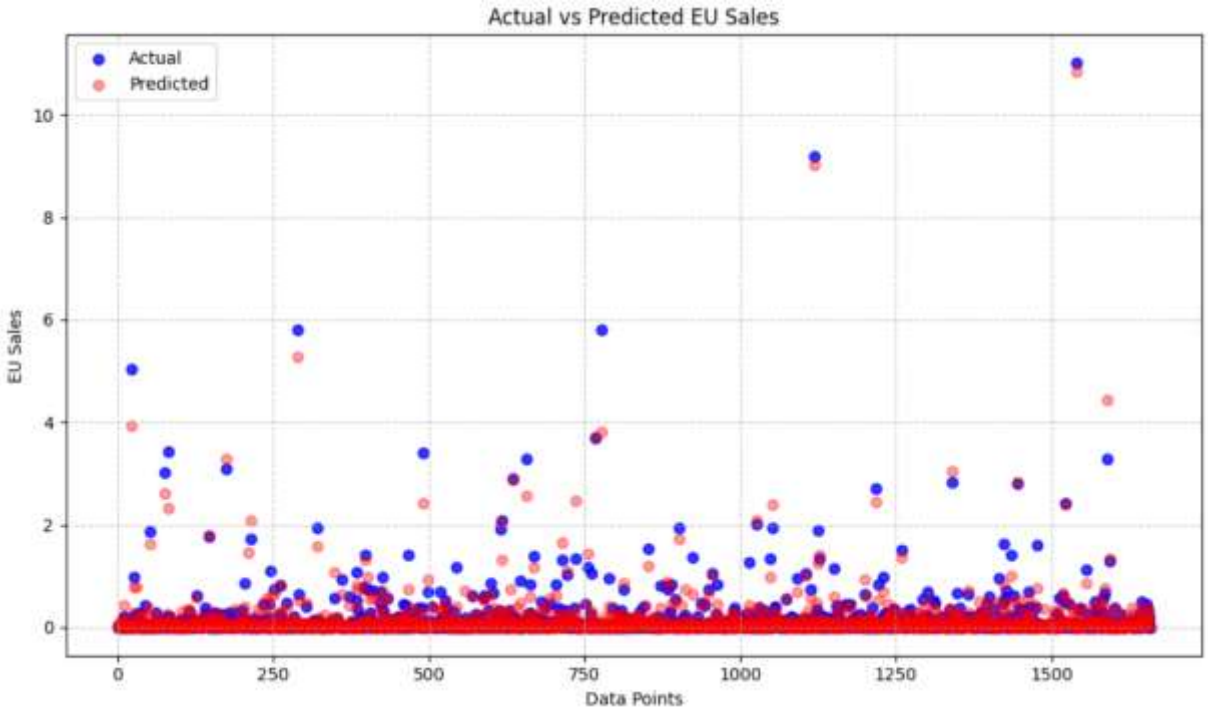


Рис.3.24. Передбачені дані EU_sales за допомогою моделі та справжні значення.

3.2 Аналіз результатів

За результатами візуального аналізу можна сказати, що у житті є багато різних умов, які так чи інакше впливають на продажі. Іноді це можуть бути і зовнішні чинники, що зовсім не пов’язані з іграми на пряму. Через такі «незаплановані» події виникає зазначена неоднорідність, і продажі або злітають до небачених висот, або навіть не покривають витрати.

Починаючи з нульових років ігрова індустрія набирає усе більшу популярність. Нові ігри отримують більше уваги користувачів та як результат приносять все більше коштів.

Останнім часом багатокористувацькі та “action” жанри домінують на ринку. Гонки, шутери, спорт – усі ігри де присутній кооператив, мультиплеєр або взаємозв’язок з іншими. Як кажуть: «Грати з друзями веселіше», і результати візуалізації це підтверджують. Значна частина (приблизно 50-60%) випущених ігор різними студіями є багатокористувацькими. Розглядаючи продажі за жанрами вони посідають друге і тертє міцся відповідно.

Що до сінглплеєр ігор, беззаперечним лідером продажів та випуску є “action” (20% від усіх ігор). Зазвичай це крупні ААА ігри (мінімум 15 годин проходження), з якісною графікою та сюжетом. Розробляються великими студіями та потребують значних витрат. Найгірший результат, в свою чергу, показують «Стратегії», жанр, що пережив свої найкращі роки та не зміг принести прибуток більше 50 мільйонів ні на одному із ринків.

Не дивно, що найуспішнішими компаніями є крупні видавці, які активно працюють та випускають багато ігор в рік (від 25 до 200). Nintendo, Electronic Arts, Activision Blizzard, Ubisoft і Sony Computer Entertaiment. Перше місце займають Nintendo хоча вони і випустили всього 6% від загальної кількості ігор. Вони вдало обирають час для випуску ігор в потрібному жанрі, були найактивнішою компанією до 2010 року та правильно встановлюють ціни на свою продукцію. Хоча останніми роками їх не можна назвати найуспішнішою

команією. Electronic Arts та Ubisoft вийшли вперед за рахунок своїх франшиз та швидкості роботи.

Що до платформ, як було згадано раніше, більшість з них – застарілі. На сьогоднішній день популярною платформою є ПК, оскільки він є більш доступним з точки зору ціни. На ПК існують спеціальні підписки та хмарні сервіси, які навіть не вимагають придбання гарнітури. Ціни на ігри для ПК нижчі, і вони пропонують багато знижок у порівнянні з консолями останнього покоління. В обраному ж датасеті найпопулярнішою є консоль DS від Nintendo, 2004 року випуску. Важливо зазначити, що розробники дуже тісно пов'язані з платформами, на яких вони випускають свої ігри. Так само жанри також мають великий вплив на продажі.

Також можна сказати, що кожен ринок унікальний і має свої вподобання. Особливо яскраво це виражено у японському ринку. Хоча найбільший вплив має Америка, Японія також досить великий видавець і споживач у ігровій індустрії. Географічне положення країни та менталітет сильно вплинули на багато сфер їх життя. Ігрова індустрія не стала вийнятком. У той час коли в світі поплуярний жанр “action”, вони віддають перевагу рольовим іграм. Також японці більше підтримують власних виробників. І на жаль, тільки японський ринок має негативну кореляцію прибутку з роками.

Аналізуючи результати регресійних моделей, виявлено, що випадковий ліс може прогнозувати до 54% глобальних продажів ігор, використовуючи інформацію про видавця, рік випуску, платформу та жанр. Це означає, що ці фактори дійсно мають значний вплив на успішність продажів. В свою чергу, за допомогою моделі, що використовує дані про глобальні продажі, можна з точністю до 93% визначити очікувані продажі на ринках Європи та Америки. Це дозволяє з високою достовірністю передбачити, які регіони будуть найбільш сприйнятливими для конкретної гри. Ці результати підкреслюють важливість аналізу ринкових факторів та споживацьких тенденцій для розробників та видавців ігор.

На підставі отриманих результатів можна зробити наступні рекомендації для покращення стратегій маркетингу, розвитку і управління в ігровій індустрії:

- Досліджувати популярні жанри та зв'язки між ними. Враховуючи, що жанри багатокористувацьких ігор та "action" є основними на ринку, акцентувати увагу на розробці та маркетингу ігор саме з цими характеристиками.
- Враховувати вартість розробки ігор та потенційні продажі на різних ринках (за допомогою регресійних моделей). Ретельно аналізувати цінові пропозиції конкурентів та реагувати на зміни в ринковій кон'юнктурі. Вводити знижки, спеціальні пропозиції та сезонні акції для заохочення споживачів.
- Розглядати партнерство із крупними видавцями, для залучення широкої аудиторії та забезпечення необхідними ресурсами для розвитку та маркетингу ігор.
- Основуючись на географічних особливостях різних ринків, розробити специфічні стратегії для кожного регіону. Наприклад, залучення японських розробників у свій проект для збільшення успішності продажів в цій країні.
- Зосередитися на розвитку ігор саме для сучасних платформ, таких як ПК, консолі та мобільні пристрої.

Висновки до третього розділу

У третьому розділі було продемонстровано усі отримані діаграми, графіки, розрахунки. Проведено аналіз значень зображених на них. Отримано повну візуалізацію зв’язків, тенденцій та даних наявних у датасеті. Та як результат підведено підсумки із обґрунтованими рекомендаціями для покращення стратегій маркетингу, розвитку і управління.

ВИСНОВКИ

В результаті виконання курсової роботи було проведено аналіз датасету «Video Game Sales». Було розглянуто тип та розподіл даних з набору, залежності між ними та візуалізовано це на графіках і діаграмах. Після чого проаналізовані результати та підведені підсумки.

Отриманий результат відповідає темі курсової роботи, а саме «Візуалізація продажу відеоігор».

В першому розділі курсової роботи було розглянуто основні аспекти візуального аналізу та обрано відповідний інструментарій для роботи із ними. Також були описані формули та алгоритми інтелектуального аналізу даних, враховано їх переваги та недоліки.

В другому розділі курсової роботи було побудовано модель аналізу та реалізовано її на практиці з використанням обраного програмного забезпечення. Представлено і описано основні реалізовані методи, що відповідають за роботу програми.

В третьому розділі курсової роботи було продемонстровано усі отримані діаграми, графіки, розрахунки. Проведено аналіз значень зображених на них. Отримано повну візуалізацію зв’язків, тенденцій і даних наявних у датасеті, та як результат підведено підсумки із обґрунтованими рекомендаціями для покращення стратегій маркетингу, розвитку і управління.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Video Game Sales visualization [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.kaggle.com/code/kostyabahshetsyan/video-game-sales-visualization/notebook.
2. API reference - kde [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.kde.html.
3. Regression in Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/python_ml_linear_regression.asp#:~:text=The%20term%20regression%20is%20used,the%20outcome%20of%20future%20events.
4. RandomForestRegressor [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/random-forest-regression-in-python/.
5. Data Visualization [Електронний ресурс] – Режим доступу до ресурсу: https://builtin.com/data-science/data-visualization-tutorial.
6. Association Analysis in Python [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/analytics-vidhya/association-analysis-in-python-2b955d0180c.
7. KMeans Clustering [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/how-to-perform-kmeans-clustering-using-python-7cc296cec092.
8. Regression Analysis [Електронний ресурс] – Режим доступу до ресурсу: https://www.tibco.com/reference-center/what-is-regression-analysis#:~:text=Regression%20analysis%20is%20a%20statistical,dependent%20variable%20against%20independent%20variables.
9. Sales Vizualization [Електронний ресурс] – Режим доступу до ресурсу: https://about.crunchbase.com/blog/sales-data-visualization/.
10. TSNE [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

ДОДАТКИ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import io
import datetime
%matplotlib inline
from google.colab import files
uploaded = files.upload()
vgs = pd.read_csv(io.StringIO(uploaded['vgsales.csv'].decode('utf-
8')))
vgs.info()

vgs.head()
# Перевірка на наявність пропущених значень
print(vgs.isnull().sum())
# Перевірка наявності дублікатів
print("Dublicated:")
print(vgs.duplicated().sum())
# Перевірка статистичних показників для числових стовпців
print(vgs.describe())
# Заміна пропущених значень у стовпці "Year" на середнє значення
mean_year = vgs['Year'].mean()
vgs['Year'].fillna(mean_year, inplace=True)
# Видалення рядків з пропущеними значеннями у стовпці "Publisher"
vgs.dropna(subset=['Publisher'], inplace=True)

sns.kdeplot(vgs['Global_Sales'])
plt.xlabel('Global Sales')
plt.ylabel('Density')
plt.title('Density Plot of Global Sales')
# Встановлення меж осі x
plt.xlim(-5, 5)
# Розподілення позначок рівномірно
ticks = np.linspace(-5, 5, 9)
```

```
plt.xticks(ticks)
plt.show()


data = vgs['Global_Sales']
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = data[(data < lower_bound) | (data > upper_bound)]
print("Outliers:")
print(outliers)


median_sales = vgs['Global_Sales'].median()
mean_sales = vgs['Global_Sales'].mean()


print("Median Global Sales:", median_sales)
print("Mean Global Sales:", mean_sales)


from matplotlib import cm
sales_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales', 'Global_Sales']
fig, ax = plt.subplots(figsize=(10, 6))
bp = ax.boxplot(vgs[sales_columns], patch_artist=True, notch=True,
vert=False, showfliers=True)
colors = [plt.colormaps['plasma'](i / len(sales_columns)) for i in
range(len(sales_columns))]
for patch, color in zip(bp['boxes'], colors):
    patch.set(facecolor=color)
outlier_colors = [plt.colormaps['plasma']((i+1) /
(len(sales_columns)+1)) for i in range(len(sales_columns))]
for i, flier in enumerate(bp['fliers']):
    flier.set(markerfacecolor=outlier_colors[i], marker='o',
alpha=0.5)
ax.set_yticklabels(sales_columns)
ax.set_xlabel('Sales')
ax.set_title('Box Plot of Sales Types')
```

```
    ax.grid(axis='x', linestyle='--', linewidth=0.5)
    plt.tight_layout()
    plt.show()


    correlation_matrix = vgs.corr(numeric_only=True)
    mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5, mask=mask)
    plt.title('Correlation Matrix')
    plt.show()
    print('')
    print(correlation_matrix)



    import matplotlib.pyplot as plt
    sales_by_year = vgs.groupby('Year')['Global_Sales'].sum()
    plt.plot(sales_by_year.index, sales_by_year.values, linestyle='-',
color='c')
    plt.fill_between(sales_by_year.index, sales_by_year.values,
color='c', alpha=0.3)
    plt.xlabel('Year')
    plt.ylabel('Global Sales')
    plt.title('Global Sales by Year')
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.grid(color='lightgray', linestyle='--')
    plt.xticks(rotation=45)
    plt.show()


    import matplotlib.pyplot as plt
    from matplotlib.cm import get_cmap
    from matplotlib.lines import Line2D
    platform_sales =
vgs.groupby('Platform')['Year'].value_counts().unstack(fill_value=0)
    platform_sales = platform_sales.loc[platform_sales.sum(axis=1) >=
100]
```

```
    num_platforms = len(platform_sales)
    num_subplots = int(np.ceil(num_platforms / 5))
    fig, axes = plt.subplots(num_subplots, 1, figsize=(10,
6*num_subplots))
    for i, ax in enumerate(axes):
        start = i * 5
        end = start + 5
        platform_subset = platform_sales.iloc[start:end]
        colors = plt.colormaps['plasma'](np.linspace(0, 1,
len(platform_subset.index)))
        for j, (platform, color) in
enumerate(zip(platform_subset.index, colors)):
            ax.plot(platform_subset.columns,
platform_subset.loc[platform], linestyle='-', color=color)
            ax.fill_between(platform_subset.columns,
platform_subset.loc[platform], color=color, alpha=0.3)
        ax.set_xlabel('Year')
        ax.set_ylabel('Number of Games')
        ax.set_title(f'Number of Games Released by Platform and Year
(Part {i+1})')
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)
        ax.grid(color='lightgray', linestyle='--')
        ax.set_xticks(platform_subset.columns)
        ax.set_xticklabels(platform_subset.columns, rotation=45)
        lines = [Line2D([0], [0], color=c, label=l) for c, l in
zip(colors, platform_subset.index)]
        ax.legend(handles=lines, loc='upper left')
        ax.set_ylim(10, ax.get_ylim()[1])
    plt.tight_layout()
    plt.show()



    table_sales =
pd.pivot_table(vgs,values=['Global_Sales'],index=['Year'],columns=['Gen
re'],aggfunc='max',margins=False)
    plt.figure(figsize=(19,16))
    sns.heatmap(table_sales['Global_Sales'],linewidths=.5,annot=True,v
min=0.01,cmap='PuBu')
```

```
plt.title('Global_Sales of games')

table_count =
pd.pivot_table(vgs,values=['Global_Sales'],index=['Year'],columns=['Gen
re'],aggfunc='count',margins=False)

plt.figure(figsize=(19,16))

sns.heatmap(table_count['Global_Sales'],linewidths=.5,annot=True,f
mt='2.0f',vmin=0,cmap='YlGn')

plt.title('Count of games')

table_sales = pd.pivot_table(vgs, values=['Global_Sales'],
index=['Year'], columns=['Publisher'], aggfunc='sum', margins=False)

top_10_publishers =
table_sales['Global_Sales'].sum().sort_values(ascending=False)[:10]  #
Вибираємо топ-10 студій за продажами

table_top_publishers =
table_sales['Global_Sales'][top_10_publishers.index]



plt.figure(figsize=(19, 16))

sns.heatmap(table_top_publishers, linewidths=0.5, annot=True,
fmt='2.0f', vmin=0, cmap='Oranges')

plt.title('Top 10 Publishers by Sales')

plt.show()

import matplotlib.ticker as ticker

regions = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales',
'Global_Sales']

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))

axes = axes.flatten()

for i, region in enumerate(regions):

    genre_sales = vgs.groupby('Genre')[region].sum().reset_index()

    sns.barplot(x='Genre', y=region, data=genre_sales, ax=axes[i],
palette='viridis')

    axes[i].set_xlabel('', fontsize=0)

    axes[i].set_ylabel('', fontsize=0)

    axes[i].set_title('{}'.format(region), fontsize=12)

    axes[i].set_xticklabels(genre_sales['Genre'], rotation=90,
fontsize=8)

    axes[i].yaxis.set_major_formatter(ticker.StrMethodFormatter('{
x:,.0f}'))

    plt.xticks(rotation=90, fontsize=8)

if len(regions) < len(axes):

    for j in range(len(regions), len(axes)):
```

```
            fig.delaxes(axes[j])
    fig.suptitle('Total Sales by Genre in Different Regions',
fontsize=14)
    plt.tight_layout()
    plt.show()


    regions = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales',
'Global_Sales']
    top_10_developers =
vgs.groupby('Publisher')['Global_Sales'].sum().nlargest(10).index.tolis
t()
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(12, 6))
    axes = axes.flatten()
    for i, region in enumerate(regions):
        developer_sales =
vgs[vgs['Publisher'].isin(top_10_developers)].groupby('Publisher')[regi
on].sum().reset_index()
        sns.barplot(x=region, y='Publisher', data=developer_sales,
ax=axes[i], palette='coolwarm')
        axes[i].set_xlabel('', fontsize=10)
        axes[i].set_ylabel('', fontsize=10)
        axes[i].set_title('({})'.format(region), fontsize=12)
        axes[i].set_yticklabels(developer_sales['Publisher'],
fontsize=8)
        axes[i].xaxis.set_major_formatter(ticker.StrMethodFormatter('{
x:,.0f}'))
        plt.xticks(fontsize=8)
    if len(regions) < len(axes):
        for j in range(len(regions), len(axes)):
            fig.delaxes(axes[j])
    fig.suptitle('Total Sales by Publisher in Different Regions',
fontsize=14
    plt.tight_layout()
    plt.show()


    plt.figure(figsize=(12, 4))
    genre_counts = vgs['Genre'].value_counts()
    colors = sns.color_palette('viridis', len(genre_counts))
    explode = [0.1] * len(genre_counts)
```

```
    wedgeprops = {'linewidth': 1, 'edgecolor': 'white'}
    patches, _ = plt.pie(genre_counts, colors=colors, explode=explode,
startangle=90, wedgeprops=wedgeprops)
    plt.axis('equal')
    for patch in patches:
        patch.set_label('')
    total = sum(genre_counts)
    percentages = [(count / total) * 100 for count in genre_counts]
    labels = ['{0} - {1:1.1f}%'.format(label, percentage) for label,
percentage in zip(genre_counts.index, percentages)]
    plt.legend(patches, labels, loc='center left', bbox_to_anchor=(0,
0.5), fontsize=10)  # Помещение легенды слева
    plt.title('Genre Distribution')


    plt.subplots_adjust(left=0.3)
    plt.show()



    import seaborn as sns
    plt.figure(figsize=(12, 4))
    platform_counts = vgs['Platform'].value_counts()
    platform_counts = platform_counts[platform_counts /
platform_counts.sum() >= 0.01]
    colors = sns.color_palette('plasma', len(platform_counts))
    explode = [0.1] * len(platform_counts)
    wedgeprops = {'linewidth': 1, 'edgecolor': 'white'}
    patches, _ = plt.pie(platform_counts, colors=colors,
explode=explode, startangle=90, wedgeprops=wedgeprops)
    plt.axis('equal')
    for patch in patches:
        patch.set_label('')
    total = sum(platform_counts)
    percentages = [(count / total) * 100 for count in platform_counts]
    labels = ['{0} - {1:1.1f}%'.format(label, percentage) for label,
percentage in zip(platform_counts.index, percentages) if percentage >=
1]
    plt.legend(patches, labels, loc='center left', bbox_to_anchor=(0,
0.5), fontsize=10)
    plt.title('Platform Distribution')
```

```
plt.subplots_adjust(left=0.3)
plt.show()



plt.figure(figsize=(18, 4))
publisher_counts = vgs['Publisher'].value_counts()
publisher_counts = publisher_counts[publisher_counts / publisher_counts.sum() >= 0.01]
colors = sns.color_palette('coolwarm', len(publisher_counts))
explode = [0.1] * len(publisher_counts)
wedgeprops = {'linewidth': 1, 'edgecolor': 'white'}


patches, _ = plt.pie(publisher_counts, colors=colors, explode=explode, startangle=90, wedgeprops=wedgeprops)
plt.axis('equal')
for patch in patches:
    patch.set_label('')
total = sum(publisher_counts)
percentages = [(count / total) * 100 for count in publisher_counts]
labels = ['{0} - {1:1.1f}%'.format(label, percentage) for label, percentage in zip(publisher_counts.index, percentages) if percentage >= 1]
plt.legend(patches, labels, loc='center left', bbox_to_anchor=(0, 0.5), fontsize=10)
plt.title('Publisher Distribution')
plt.subplots_adjust(left=0.3)
plt.show()



n_colors = 32
palette = sns.color_palette('coolwarm', n_colors=n_colors)
plt.figure(figsize=(12, 6))
sns.scatterplot(data=vgs, x='Year', y='Platform', hue='Platform', palette=palette, s=100, alpha=0.7)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Platform', fontsize=12)
plt.title('Scatter Plot of Platform - Year', fontsize=14)
```

```
plt.grid(False)
plt.yticks(fontsize=10)
plt.legend(title='Platform', title_fontsize=10, loc='center left', bbox_to_anchor=(1, 0.5), fontsize=10)
plt.show()



top_10_developers = vgs.groupby('Publisher')['Global_Sales'].sum().nlargest(10).index.tolist()
filtered_data = vgs[vgs['Publisher'].isin(top_10_developers)]


sales_by_publisher_genre = filtered_data.groupby(['Publisher', 'Genre'])['Global_Sales'].sum().reset_index()
sales_by_publisher_genre['Percentage'] = (sales_by_publisher_genre['Global_Sales'] / sales_by_publisher_genre.groupby('Publisher')['Global_Sales'].transform('sum'))
heatmap_data = pd.pivot_table(sales_by_publisher_genre, values='Percentage', index='Publisher', columns='Genre', fill_value=0)
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, cmap='YlGnBu', annot=True, fmt='.1%', cbar=False)
plt.xlabel('Genre')
plt.ylabel('Publisher')
plt.title('Percentage of Sales by Genre for Top 10 Publishers')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()



import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
vgs[categorical_features] = vgs[categorical_features].astype(str)
transactions = vgs[categorical_features].values.tolist()
```

```
    te = TransactionEncoder()
    te_ary = te.fit_transform(transactions)
    df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = apriori(df_encoded, min_support=0.05,
use_colnames=True)
    frequent_itemsets = frequent_itemsets.sort_values(by='support',
ascending=False)
    frequent_itemsets['itemsets'] =
frequent_itemsets['itemsets'].apply(lambda x: ', '.join(list(x)))
    colors = sns.color_palette('Blues_r', len(frequent_itemsets))
    fig, ax = plt.subplots(figsize=(10, 6))
    bars = ax.barh(range(len(frequent_itemsets)),
frequent_itemsets['support'], color=colors)
    ax.set_xlabel('Support', fontsize=12)
    ax.set_ylabel('Itemsets', fontsize=12)
    ax.set_title('Frequent Items', fontsize=14)
    ax.set_yticks(range(len(frequent_itemsets)))
    ax.set_yticklabels(frequent_itemsets['itemsets'], fontsize=10)
    ax.set_xlim(0, 1)
    for i, bar in enumerate(bars):
        ax.text(bar.get_width() + 0.01, bar.get_y() + bar.get_height()
/ 2,
                f'{frequent_itemsets.iloc[i]["support"]:.2f}',
va='center')
    plt.tight_layout(pad=2)
    plt.savefig('frequent_itemsets_plot.png')
    plt.show()
     print(frequent_itemsets)



    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    import matplotlib.pyplot as plt
    import warnings
    warnings.filterwarnings("ignore", category=DeprecationWarning)
    frequent_itemsets2 = frequent_itemsets.sort_values(by='support',
ascending=False).head(20)
```

```
      plt.figure(figsize=(10, 6))
      plt.barh(range(len(frequent_itemsets2)),
frequent_itemsets2['support'], tick_label=[', '.join(itemset) for
itemset in frequent_itemsets['itemsets']], color=colors)
      plt.xlabel('Support')
      plt.ylabel('Itemsets')
      plt.title('Most frequent Itemsets')
      plt.tight_layout()
      plt.show()



      df_filtered = vgs.copy()
      #threshold = 100
      #threshold = 0
      #categorical_columns = ['Genre', 'Publisher', 'Platform']
      #for column in categorical_columns:
       #   value_counts = df_filtered[column].value_counts()
       #   outliers = value_counts[value_counts < threshold].index
       #   df_filtered =
df_filtered[~df_filtered[column].isin(outliers)]
      #print(df_filtered)



      categorical_features = ['Genre', 'Publisher',
'Platform','Global_Sales']
      X_categorical = df_filtered[categorical_features]
      encoder = OneHotEncoder()
      X_encoded = encoder.fit_transform(X_categorical)
      k = 3
      kmeans = KMeans(n_clusters=k, n_init='auto', random_state=42)
      kmeans.fit(X_encoded)
      labels = kmeans.predict(X_encoded)
      df_filtered['Cluster'] = labels
      labels = kmeans.labels_
      print(labels)
```

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_encoded.toarray())
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=labels, cmap='viridis')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.title('Cluster Visualization')
plt.colorbar()
plt.show()



import scipy.stats
pd.options.display.float_format = '{:.3f}'.format
grouped_stats = df_filtered.groupby('Cluster')[['Other_Sales',
'Global_Sales']].agg(['mean', 'median'])
print(grouped_stats)
import matplotlib.pyplot as plt
import numpy as np
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10),
sharex=True)
clusters = grouped_stats.index
features = ['Other_Sales', 'Global_Sales']
colors = ['blue', 'orange']
x_pos = np.arange(len(clusters))
bar_width = 0.1
opacity = 0.7
for i, feature in enumerate(features):
    values_mean = grouped_stats[(feature,
'mean')].values.flatten()
    ax1.bar(x_pos + i * bar_width, values_mean, color=colors[i],
width=bar_width, label=feature, alpha=opacity)
for i, feature in enumerate(features):
        values_median = grouped_stats[(feature,
'median')].values.flatten()
        ax2.bar(x_pos + i * bar_width, values_median,
color=colors[i], width=bar_width, label=feature, alpha=opacity)
ax1.set_xticks(x_pos + (len(features) - 1) * bar_width / 2)
ax1.set_xticklabels(clusters)
```

```
    ax2.set_xticks(x_pos + (len(features) - 1) * bar_width / 2)
    ax2.set_xticklabels(clusters)
    ax1.legend(loc='upper right')
    ax1.set_ylabel('Mean')
    ax1.set_title('Grouped Sales Statistics (Mean) by Cluster')
    ax2.legend(loc='upper right')
    ax2.set_ylabel('Median')
    ax2.set_xlabel('Cluster')
    ax2.set_title('Grouped Sales Statistics (Median) by Cluster')

    plt.tight_layout()
    plt.show()



    import pandas as pd
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    sales_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales','Global_Sales','Rank']
    features = [col for col in df_filtered.columns if col not in
sales_columns]
    target = 'Global_Sales'
    X = df_filtered[features]
    y = df_filtered[target]
    label_encoder = LabelEncoder()
    X_encoded = X.copy()
    for feature in features:
        X_encoded[feature] = label_encoder.fit_transform(X[feature])
    X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.1, random_state=1000)
    model = RandomForestRegressor(n_estimators=200, random_state=300)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = model.score(X_test, y_test)
    print(f'Accuracy: {accuracy}')
    plt.figure(figsize=(10, 6))
```

```
     plt.scatter(range(len(y_test)), y_test, color='blue',
label='Actual', alpha=0.8)

     plt.scatter(range(len(y_test)), y_pred, color='red',
label='Predicted', alpha=0.4)

     plt.xlabel('Data Points')

     plt.ylabel('Global Sales')

     plt.title('Actual vs Predicted Global Sales')

     plt.legend()

     plt.grid(True, linestyle='--', alpha=0.7)

     plt.tight_layout()

     plt.show()

     genre_data = vgs[vgs['Genre'] == 'Action']

     text = ' '.join(genre_data['Name'])


     wc = WordCloud(background_color="white", max_words=2000,
stopwords=stopwords,

                    max_font_size=40, random_state=42)

     wc.generate(text)

     plt.imshow(wc, interpolation="bilinear")

     plt.axis("off")

     plt.show()

     #importance = pd.DataFrame({'Feature': features, 'Importance':
model.feature_importances_})

     #print(importance)
```