НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

(назва дисципліни)

на тему:_____Аналіз продажів комп'ютерних ігор_____

_____

Студента (ки) 2 курсу _ІТ-01_ групи

Спеціальності 121

_____Дмитрієвої І.І._____

(прізвище та ініціали)

«ПРИЙНЯВ» з оцінкою

_____

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

_____

Підпис Дата

Київ - 2022 рік

ЗМІСТ

ВСТУП

Відеоігри — це електронні ігри, які передбачають взаємодію з користувацьким інтерфейсом або пристроєм введення, наприклад джойстиком, контролером, клавіатурою або пристроєм для визначення руху, для створення візуального зворотного зв'язку. Сама гра зазвичай виводиться на пристрої відображення відео, наприклад на телевізорі, моніторі, сенсорному екрані або гарнітурі віртуальної реальності. Залежно від способу виводу та інструментів, за допомогою яких іде управління, відеоігри можуть поділятися на комп'ютерні, консольні, мобільні, браузерні тощо.

Відеоігри пройшли довгий шлях з моменту появи перших ігор у 1970-х роках. За 40 років існування з'явилося безліч нових жанрів, видів графіки, способів та технік управління грою. Відеоігри тісно пов'язувалися з культурою того часу, тому часто випускалися по сюжетам бестселерних книг, популярних фільмів, робилися ремейки з більш сучасною графікою та новими деталями в сюжеті, аби увага гравців-користувачів не згасала ні до фільмів чи книг, ні до ігор. По багатьом відеоіграм навіть робили так званий "мерч" (товари-сувеніри по всесвіту даної реальності).

Відеоігри активно завойовують собі місце у сучасній культурі. як наприклад у 2021 році мультсеріал по популярній грі League of Legends побив рекорди по переглядам, а сама гра - по кількості нових гравців. Сучасні відеоігри пропонують фотореалістичну графіку та імітують реальність до ступеня, яка в багатьох випадках вражає, що й зрівнятися не може з тими аналогами, що були випущені якісь 10 років тому.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1) Огляд предметної області

Предметна область даної роботи - продажі комп’ютерних ігор у різних країнах за 1970 - 2017 роки та залежність успішності продажів від різних ознак.

Відеоігри стають дедалі популярнішими у сучасному світі, люди залежно від разних факторів частіше й частіше знаходять щось для себе з великого розмаїття ігор. Популярність різних ігор може залежати від абсолютно протилежних по змісту тематик та рейтингів, а розважальна “фермерська” гра для будь-якого віку може стояти в топ-рейтингу по продажам поруч з високорейтинговим шутером з переліком попереджень про можливий зміст, невідповідний для чутливих людей (дітей, вагітних, тощо).

The Entertainment Software Rating Board (ESRB) — це саморегулівна організація, яка призначає вікові та вмістні рейтинги споживчих відеоігор у Сполучених Штатах і Канаді. ESRB була заснована в 1994 році Асоціацією розважального програмного забезпечення (ESA, раніше Interactive Digital Software Association (IDSA)), у відповідь на критику суперечливих відеоігор із надмірно насильницьким або сексуальним вмістом, особливо після слухань у Конгресі 1993 року після випуску Mortal Kombat і Night Trap для домашніх консолей і Doom для домашніх комп'ютерів. Хоча компанія раніше працювала лише в США та Канаді, зараз системою ESRB користуються по всьому світу. Категорії поділяються відповідно до змісту, наявності попереджень щодо контенту (вживання алкоголю чи наркотичних речовин, наявність кровавих чи постільних сцен, тощо) та мінімального віку споживачів, на яких розрахований даний контент.

Незважаючи на те, що можна легко припустити, що чим менший ESRB-рейтинг - тим більша авдиторія покупців, насправді на кожен вид товару знаходиться своя кількість споживачів. Передбачення успішності й кількості продажів ігор залежить від низки факторів. Наприклад, популярність першої частини гри може змотивувати гравця купити наступну частину гри або схожу від того самого виробника, або вподобаний жанр однієї гри може підштовхнути споживача на купівлю гри зі схожим сюжетом, але зовсім з іншим рейтингом та від іншого виробника. Частіше за все гравці купують ігри на основі власних уподобань або за рекомендацією в соціальних мережах або від знайомих.

Найпоширенішим критерієм вибору гри є жанр. Людині, що любить спорт або бойовики, навряд будуть цікаві рольові ігри з довгим сюжетом. На мій погляд, прогнозування популярного жанру ігор з найвищими продажами може бути корисним розробникам аби розуміти, як розподіляти бюджет, на чому зосередити рекламу, яких спеціалістів долучати та які розробки будуть актуальними для ринку. На основі даних результатів роблять висновки про найбільший інтерес споживачів до певних категорій ігор, а отже найвищі продажі.

2) Огляд доступних джерел даних

Використані мною набори даних взяті з kaggle.com. Це два датасети, що містять різні види даних по одним і тим самим іграм.

Перший датасет (“score_count_df”) - це список ігор з даними по виробникам, розробникам, платформам, рокам випуску, рейтингу, оцінкам критиків та звичайних користувачів, а також представлені дані у мільйонах по продажам у Північній Америці, Європі, Японії та іншим країнам.

Другий датасет (“warnings_df”) містить у собі вибірку ігор з більш детальними даними про ESRB-рейтинг. Він включає в себе булеві значення, чи є дане попередження про контент у грі (1 - є, 0 - немає), а також чи продається дана гра в консолі.

За допомогою ETL-засобів завантажуємо датасет для подальшого огляду й аналізу, створюємо відповіний датафрейм. Перед аналізом потрібно передбачити

наявність неузгоджених, неінформативних та відсутніх даних у наборі. Для усунення помилок при подальшому поєднанні двох датафреймів буде виконана підготовка даних: будуть взяті “префікси” ігор у нижньому регістрі. Це робиться для того, аби не створювати дублікати записів одних і тих самих ігор, які незначною мірою відрізняються: закінченням, регістром тощо. Також рейтинг та попередження однієї гри не залежать від платформи чи продажів, отже можна поєднувати лише по префіксам.

Дані по продажам відеоігор

```
score_count_df = pd.read_csv("datasets/vgsales-4.csv")
score_count_df['prefix_l'] = score_count_df['Name'].str.slice(0, 12).str.lower()
score_count_df
```

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61
4	Pokemon Red/Pokemon Blue	G	1996.0	Role-Playing	Nintendo	11.27
...	...	...	...	...	...	...
17411	Nancy Drew: The Deadly Secret of Olde W...	DS	2007.0	Adventure	Majesco Entertainment	0.00
17412	Fashion Designer: Style Icon	DS	2007.0	Simulation	505 Games	0.00
17413	Ashita no Joe 2: The Anime Super Remix	PS2	2002.0	Fighting	Capcom	0.00
17414	NadePro!! Kisama no Seiyuu Yatte Miro!	PS2	2009.0	Adventure	GungHo	0.00
17415	Brian Lara 2007 Pressure Play	PSP	2007.0	Sports	Codemasters	0.00

JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Rating	prefix_l
3.77	8.45	82.54	76.0	51.0	8.0	324.0	E	wii sports
6.81	0.77	40.24	NaN	NaN	NaN	NaN	NaN	super mario
3.79	3.29	35.57	82.0	73.0	8.3	712.0	E	mario kart w
3.28	2.95	32.78	80.0	73.0	8.0	193.0	E	wii sports r
10.22	1.00	31.37	NaN	NaN	NaN	NaN	NaN	pokemon red/
...	...	...	...	...	...	...	...	...
0.00	0.00	0.01	64.0	7.0	NaN	NaN	E	nancy drew:
0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN	fashion desi
0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN	ashita no jo
0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN	nadepro!! ki
0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN	brian lara 2

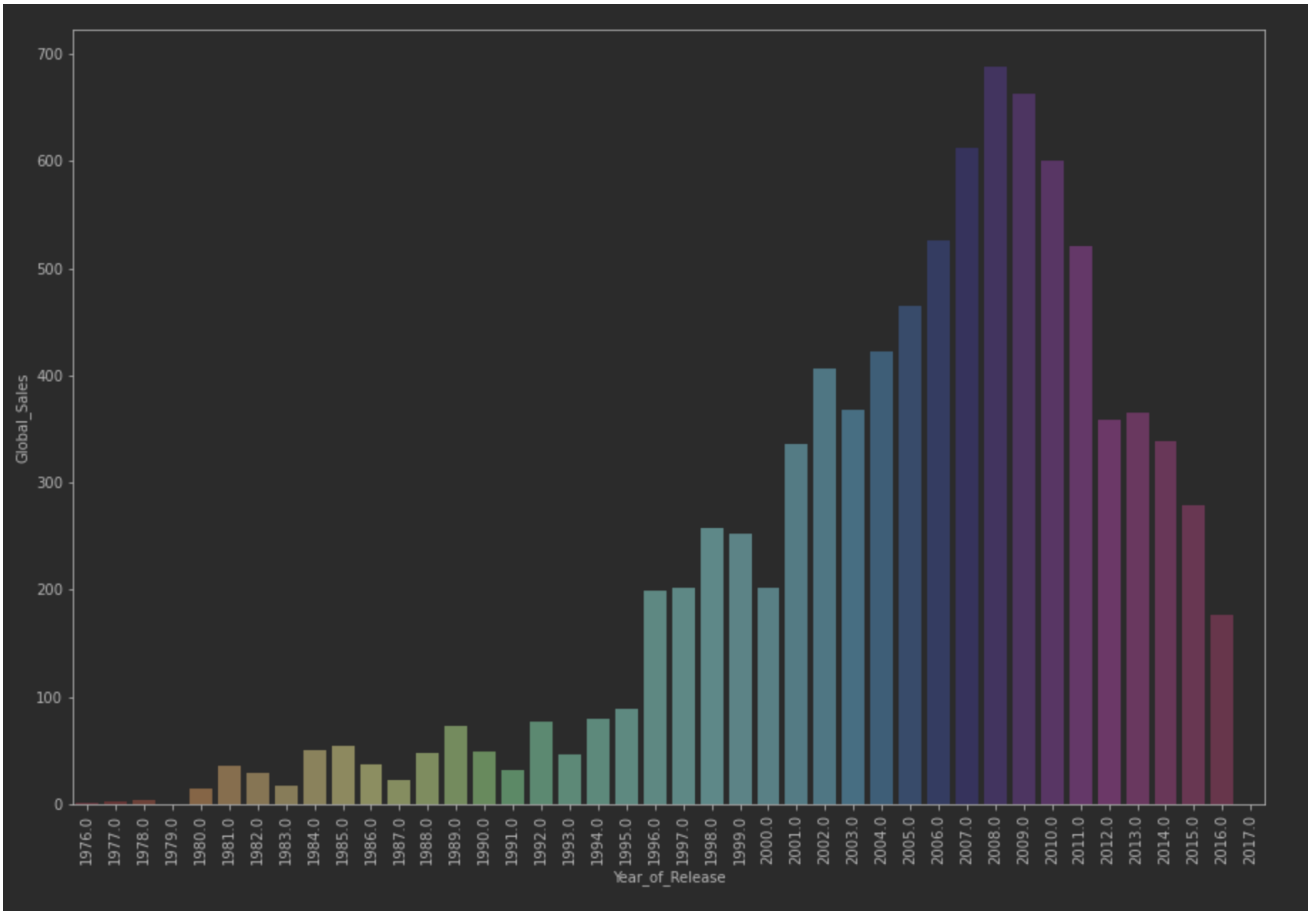
Для подальшого аналізу виведемо список часток відсутніх записів для кожної ознаки у відсотках.

```
def percent_missing_func(df):
    percent_missing = df.isnull().sum() * 100 / len(df)
    missing_value_df = pd.DataFrame({'percent_missing': percent_missing})
    return missing_value_df
```

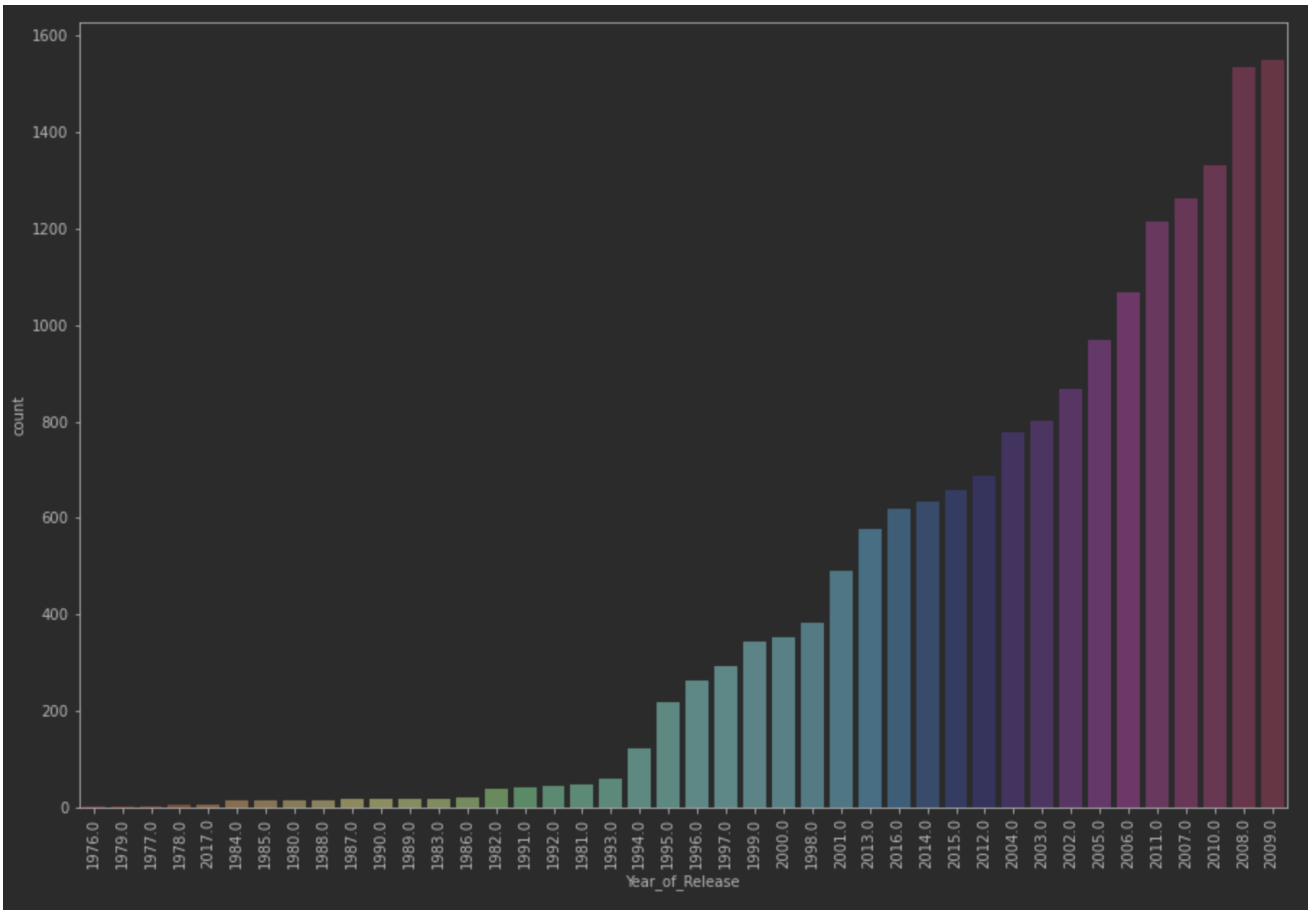
По результатам виводу можна побачити, що даних достатньо для подальшого аналізу.

	percent_missing
Name	0.000000
Platform	0.000000
Year_of_Release	0.045935
Genre	0.000000
Publisher	0.005742
NA_Sales	0.000000
EU_Sales	0.000000
JP_Sales	0.000000
Other_Sales	0.000000
Global_Sales	0.000000
Critic_Score	52.135967
Critic_Count	52.135967
User_Score	55.225080
User_Count	55.225080
Rating	41.134589
prefix_l	0.000000

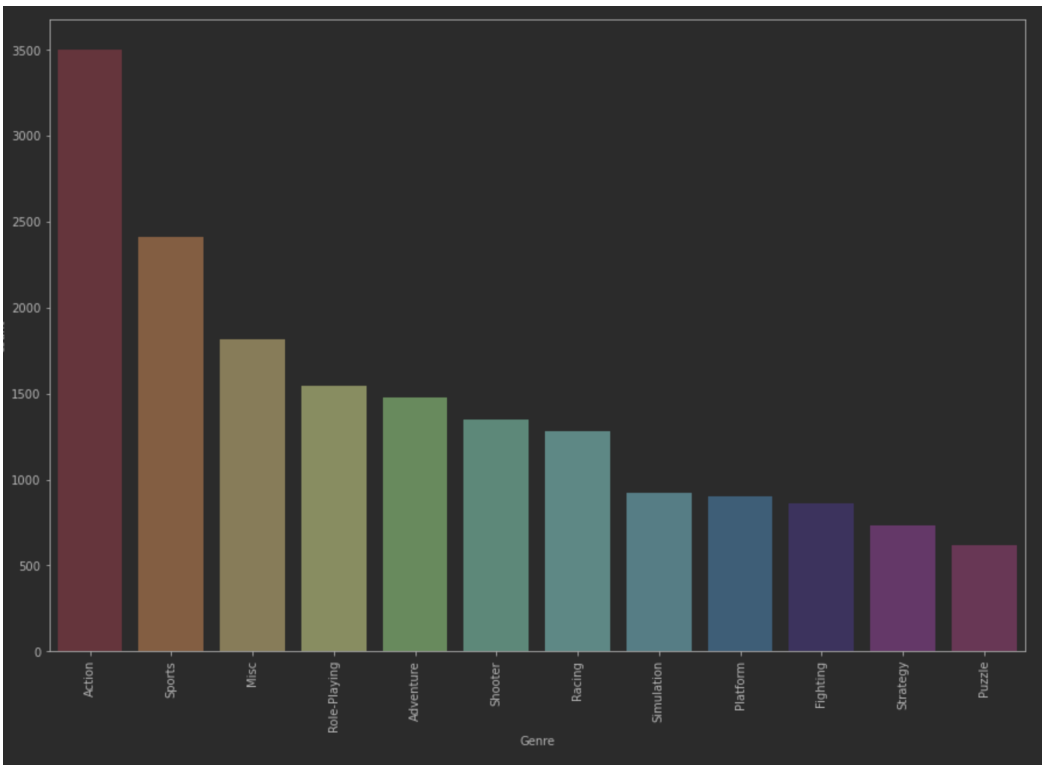
Розглянемо статистику продажів, випуску та популярності тих чи інших жанрів по 1970-2017 роках.



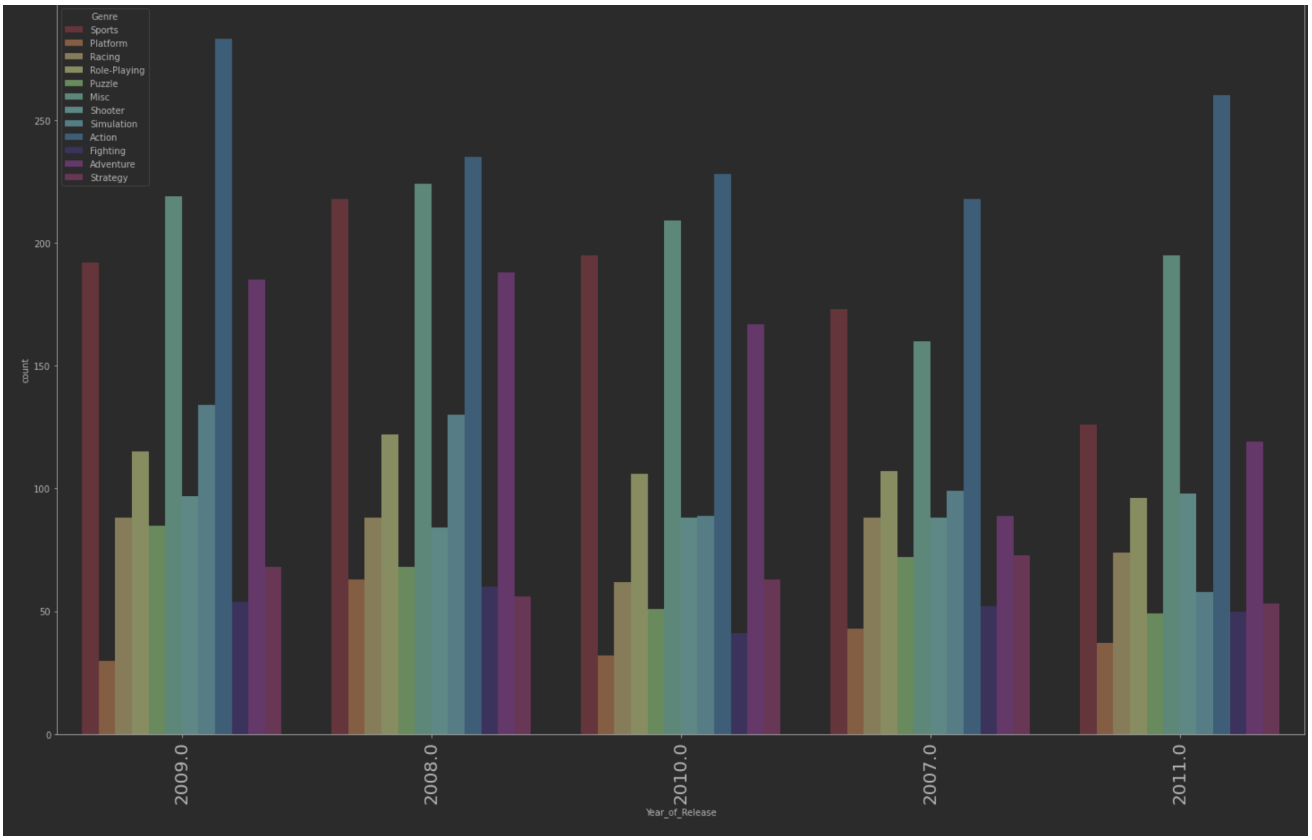
Найбільші продажі ігор по рокам



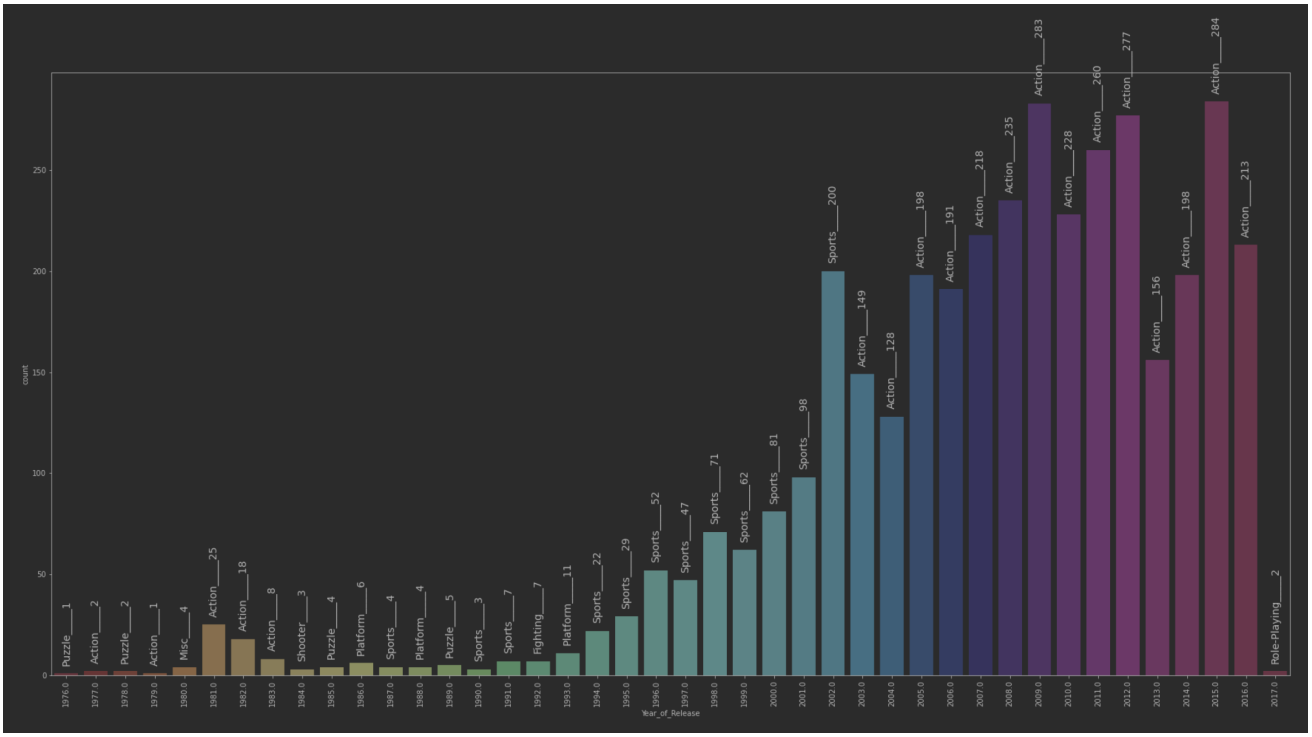
Кількість випущених ігор по роках



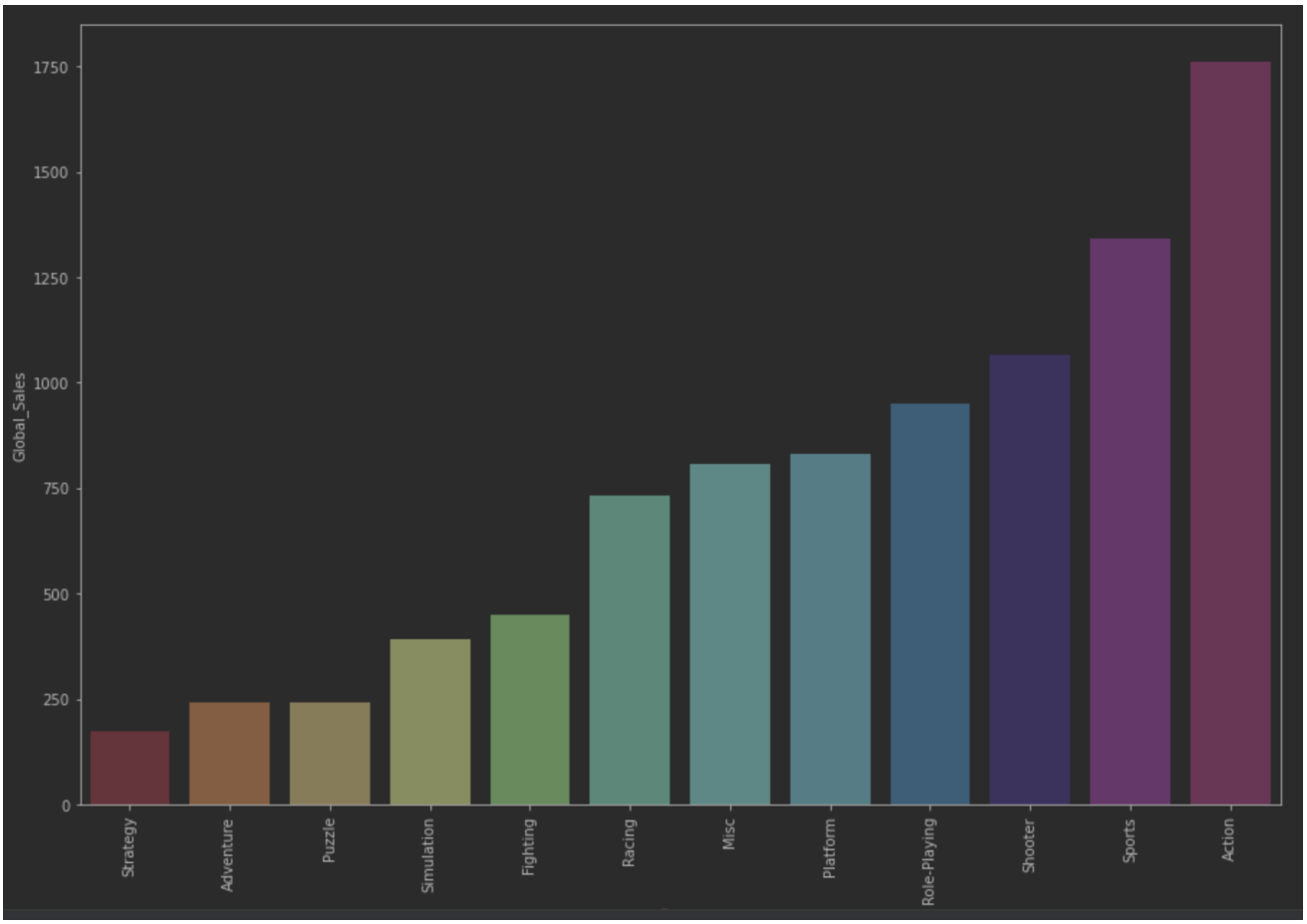
Найпопулярніший жанр серед випущених ігор



Жанри топ-5 років по кількісті по кількості випущених ігор



Найпопулярніші жанри по рокам



Жанри ігор з найвищими продажами

Також серед ознак, що нас цікавлять при аналізі, є ESRB-рейтинг. Задля оцінки залежності жанрів від рейтингу та від попереджень про контент, а для також доповнення даних про ESRB-рейтинг, поєднуємо обидва датафрейми в один. У даному випадку, як і вище, формуємо стовпчик префіксів, по яким поєднуватимемо датафрейми.

title	console	alcohol_reference	animated_blood	blood	blood_and_gore
Monster Jam Steel Titans 2	1	0	0	0	0
Subnautica: Below Zero	1	0	1	0	0
NIER REPLICANT VER.1.22474487139…	1	0	0	1	0
Jamestown+	0	0	0	0	0
Neptunia Virtual Stars	0	0	0	0	0
Monster Energy Supercross - The Officia...	1	0	0	0	0
Monochrome Order	0	0	0	1	0
Blightbound	1	0	0	0	1
Maquette	0	1	0	0	0
FATAL FURY™ BATTLE ARCHIVES VOL.2	0	0	0	0	0

Перевіряємо датафрейм на відсутність записів для кожної ознаки. Можемо зробити висновок, що пропущених значень немає.

	percent_missing
title	0.0
console	0.0
alcohol_reference	0.0
animated_blood	0.0
blood	0.0
blood_and_gore	0.0
cartoon_violence	0.0
crude_humor	0.0
drug_reference	0.0
fantasy_violence	0.0
intense_violence	0.0
language	0.0
lyrics	0.0
mature_humor	0.0
mild_blood	0.0
mild_cartoon_violence	0.0
mild_fantasy_violence	0.0
mild_language	0.0
mild_lyrics	0.0
mild_suggestive_themes	0.0
mild_violence	0.0
no_descriptors	0.0
nudity	0.0
partial_nudity	0.0
sexual_content	0.0
sexual_themes	0.0
simulated_gambling	0.0
strong_janguage	0.0
strong_sexual_content	0.0
suggestive_themes	0.0
use_of_alcohol	0.0
use_of_drugs_and_al...	0.0

Поєднуємо датафрейми в один за префіксами, виводимо 10 перших даних для перевірки та перелік стовпчиків для перевірки.

```
fact_df = pd.merge(score_count_df, warnings_df, left_on="prefix_l", right_on="prefix_r")
fact_df.head(10)
```

Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales
Grand Theft Auto V	PS3	2013.0	Action	Take-Two Interactive	7.02	9.14
Grand Theft Auto: San Andreas	PS2	2004.0	Action	Take-Two Interactive	9.43	0.40
Grand Theft Auto V	X360	2013.0	Action	Take-Two Interactive	9.66	5.17
Grand Theft Auto: Vice City	PS2	2002.0	Action	Take-Two Interactive	8.41	5.49
Grand Theft Auto V	PS4	2014.0	Action	Take-Two Interactive	4.12	6.77
Grand Theft Auto III	PS2	2001.0	Action	Take-Two Interactive	6.99	4.51
Grand Theft Auto IV	X360	2008.0	Action	Take-Two Interactive	6.77	3.07
Grand Theft Auto IV	PS3	2008.0	Action	Take-Two Interactive	4.77	3.70
Grand Theft Auto: Liberty City Stories	PSP	2005.0	Action	Take-Two Interactive	2.90	2.81
Grand Theft Auto V	XOne	2014.0	Action	Take-Two Interactive	2.93	2.32

title	console	alcohol_reference	animated_blood	blood	blood_and_gore	cartoon_violence	crude_humor
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0
Grand Theft Auto V	1	0	0	0	1	0	0

```
fact_df.shape
```

```
(3888, 51)
```

```
fact_df.columns
```

```
Index(['Name', 'Platform', 'Year_of_Release', 'Genre', 'Publisher', 'NA_Sales'
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales', 'Critic_Score',
       'Critic_Count', 'User_Score', 'User_Count', 'Rating', 'prefix_l',
       'title', 'console', 'alcohol_reference', 'animated_blood', 'blood',
       'blood_and_gore', 'cartoon_violence', 'crude_humor', 'drug_reference',
       'fantasy_violence', 'intense_violence', 'language', 'lyrics',
       'mature_humor', 'mild_blood', 'mild_cartoon_violence',
       'mild_fantasy_violence', 'mild_language', 'mild_lyrics',
       'mild_suggestive_themes', 'mild_violence', 'no_descriptors', 'nudity',
       'partial_nudity', 'sexual_content', 'sexual_themes',
       'simulated_gambling', 'strong_janguage', 'strong_sexual_content',
       'suggestive_themes', 'use_of_alcohol', 'use_of_drugs_and_alcohol',
       'violence', 'esrb_rating', 'prefix_r'],
      dtype='object')
```

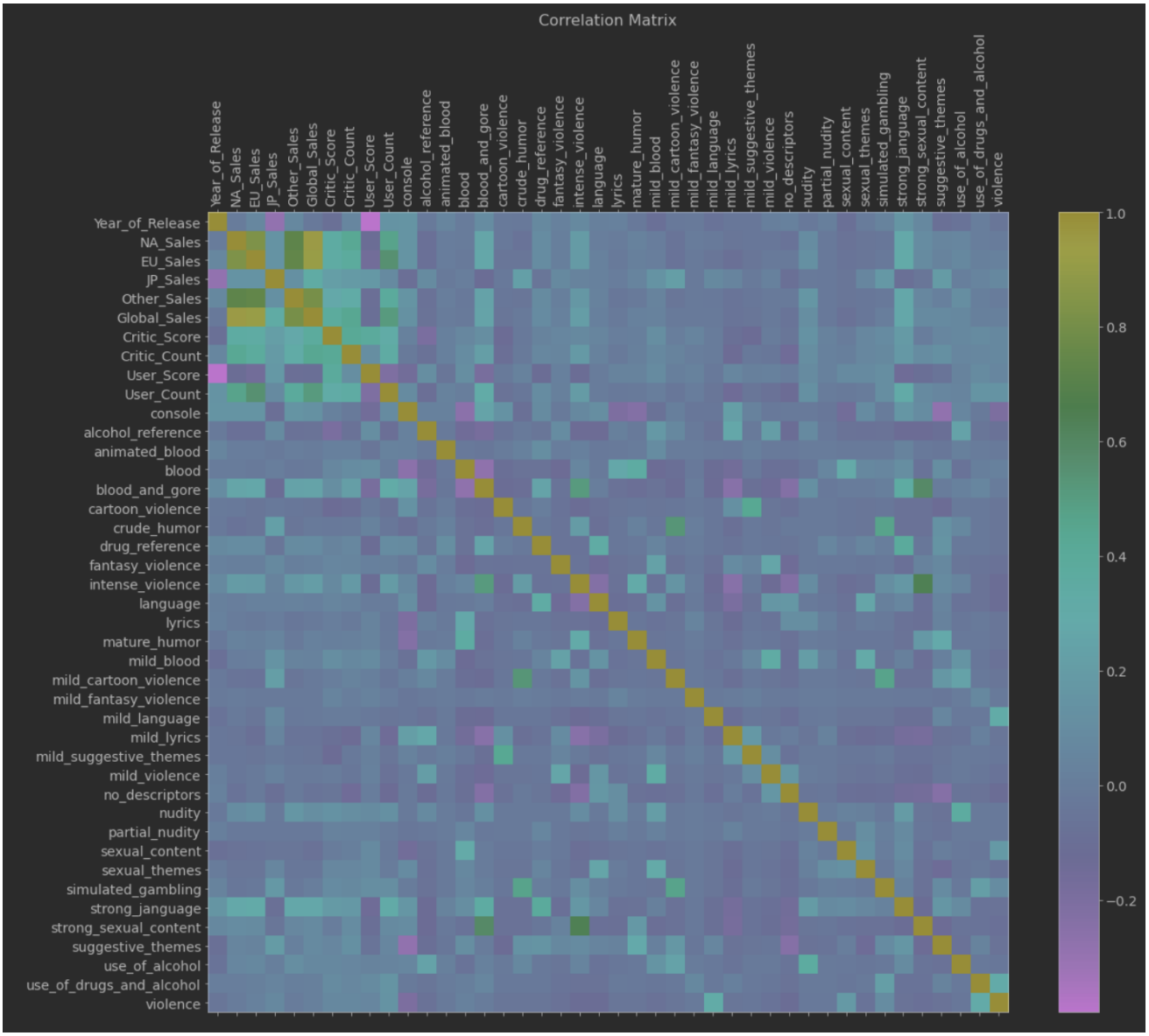
Отже, ми створили датафрейм з достатньою кількістю ознак для аналізу та створення моделей. Отже, допоміжні стовпчики з префіксами можна видалити.

За допомогою describe() виведемо описову статистику. Описова статистика включає статистику, яка підсумовує центральну тенденцію, дисперсію та форму розподілу набору даних, за винятком значень NaN.

	Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score
count	3888.000000	3888.000000	3888.000000	3888.000000	3888.000000	3888.000000	2334.000000
mean	2008.000772	0.562371	0.336975	0.168933	0.120262	1.188444	74.336761
std	6.103536	1.184550	0.778044	0.443382	0.334468	2.254293	12.842768
min	1978.000000	0.000000	0.000000	0.000000	0.000000	0.010000	19.000000
25%	2004.000000	0.010000	0.010000	0.000000	0.000000	0.120000	67.000000
50%	2009.000000	0.140000	0.060000	0.010000	0.020000	0.380000	76.000000
75%	2013.000000	0.530000	0.280000	0.130000	0.080000	1.160000	83.000000
max	2016.000000	9.730000	9.140000	4.870000	10.570000	21.120000	98.000000

count – кількість даних в кожному стовпчику
mean – середнє значення
std – стандартне відхилення
min – мінімальне значення стовпчика
25% - нижній процентиль
50% - середній процентиль, який збігається з медіаною
75% - верхній процентиль
max – максимальне значення стовпчика

Побудуємо таблицю кореляції для всіх стовпчиків.



На таблиці добре видно, що найбільше корелюють продажі по всьому світу (Global_Sales) з продажами по Європі (EU_Sales), США (NA_Sales) та іншим країнам, крім Японії (Other_Sales). Це пояснюється тим, що продажі по всьому

світу по суті є сумою всіх інших продажів, а отже може трапитися overfitting моделі. Отже, при подальшому прогнозуванні варто виключити продажі по всьому світу з ознак.

3) Постановка задачі

Мета дослідження – передбачення жанру гри по популярності, її рейтингу та продажам у різних країнах.

Проведений аналіз повинен дати на наступні питання відповідь:

1. На скільки точно можна передбачити жанр гри по розрізненим ознакам? Чи потрібно визначати окремі ознаки, за якими жанр може бути передбачений?

2. Яким способом жанр буде передбачений з найвищою точністю?

ОБГРУНТУВАННЯ ВИБОРУ МЕТОДІВ ТА МОДЕЛЕЙ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Класифікація - це алгоритм в контрольованому машинному навчанні, який навчається визначити категорії і представити, до яких категорій вони відносяться, для нових значень.

Лише частина ознак, за якими буде відбуватися прогнозування жанру у даній роботі - кількісні (продажі, оцінка та кількість критиків та користувачів). Серед ознак також є якісні (платформа, виробник, рейтинг) і булеві (попередження щодо контенту). Саме тому найкращим вибором є навчання класифікаційної моделі за допомогою класифікатору K-найближчих сусідів (KNN) та Random forest (RF).

K-найближчих сусідів (KNN) — це метод навчання з підкріпленням (з вчителем), що використовується для класифікації та регресії. В обох випадках вхідні дані складаються з k найближчих навчальних прикладів у наборі даних. Результатом є належність до певного класу. Об’єкт класифікується за допомогою множини голосів його сусідів, причому об’єкт відноситься до класу, найбільш поширеного серед його k найближчих сусідів (k — ціле додатне число, зазвичай невелике). KNN спочатку ідентифікує K точок у навчальних даних, найближчих до даного об’єкта, дано, представлених N_0. Потім він оцінює умовну ймовірність для класу j як частку точок у N_0, значення відповіді для яких дорівнює j:

$$P\left(Y = j|X = x_0\right) = \frac{1}{K} \sum_{i \in N_0} I\left(y_i = j\right)$$

Нарешті, KNN застосовує правило Байєса і класифікує тестове спостереження x_0 до класу з найбільшою ймовірністю. Якщо k = 1, то об’єкт просто приписується до класу цього єдиного найближчого сусіда.

KNN – це тип класифікації, де функція локально лише апроксимується, а всі обчислення відкладаються до оцінки функції. Оскільки цей алгоритм покладається на відстань для класифікації, якщо ознаки представляють різні фізичні одиниці або мають дуже різні масштаби, то нормалізація навчальних даних може значно підвищити їх точність.

Random forest вважається високоточним та надійним методом, оскільки у процесі прогнозування бере участь безліч дерев рішень. Random forest маловірогідно матиме проблему перенавчання, так як використовує середнє значення всіх прогнозів, що також сприяє усуненню зміщення. Random forest може використовуватися як у класифікації, так і в регресії). Random forest також розраховує відносну важливість показників, що допомагає у виборі найбільш значимих ознак класифікатора, та дуже актуально у даній роботі.

Через використання безлічі дерев RF досить повільний: кожному дереву в лісі передаються ті самі вхідні дані, на підставі яких воно має повернути своє передбачення, після чого ще й обирається найбільш підходящий варіант.

ЗАСТОСУВАННЯ ТА ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ТА МОДЕЛЕЙ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

1. Визначення моделей та методів, що можуть бути використані

Після використання KNN та RF також будуть протестовані інші класифікатори та, як результат, оцінена ефективність розроблених моделей й обрані методи шляхом порівняння з іншими класифікаторами, такими як метод опорних векторів (англ. SVM, support vector machine), Decision Tree Learning, Adaptive Boosting та інші.

2. Вибір ознак, що будуть використані для аналізу

Перед початком роботи додаємо ще стовпчики за допомогою OneHotEncoder та LabelEncoder для платформи, виробника, жанрів та рейтингу. Це робиться тому, що усі ці дані - категоріальні, й аби не виникало непередбачуваних помилок вони замінюються на відповідне їм числове значення.

```
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
# creating instance of one-hot-encoder
enc = OneHotEncoder(handle_unknown='ignore')
# passing bridge-types-cat column (label encoded values of bridge_types)
enc_df = pd.DataFrame(enc.fit_transform(fact_df[['Platform']]).toarray())
# merge with main df bridge_df on key values
f_df = fact_df.join(enc_df)
f_df = f_df.drop(columns=["Platform", "Rating"])
f = f_df.fillna(0)

f['Genre'] = LabelEncoder().fit_transform(f['Genre'])
f['Publisher'] = LabelEncoder().fit_transform(f['Publisher'])
f['esrb_rating'] = LabelEncoder().fit_transform(f['esrb_rating'].astype('str'))
```

3. Підготовка даних для навчання та верифікації моделей

Результатом передбачень будуть жанри, саме тому їх записуємо в y, а всі інші ознаки - в X.

```
from sklearn.preprocessing import StandardScaler

y=f["Genre"].values

X = f.drop(columns=["Genre", "Name", "title"])
X = StandardScaler().fit_transform(X)
```

Розділяємо усі значення на навчальні й тестові, у співвідношенні 4:1.

```
X_train,X_test, y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=4)
```

Можна побачити, що жанри у y_train дійсно перекодувалися в числові значення.

y_train

	0
0	8
1	2
2	8
3	7
4	0
5	0
6	0
7	7
8	9
9	1
10	4
11	7
12	0
13	7
14	8
15	8
16	8

4. **Формування моделей. Вибір оптимального класу складності моделей. Верифікація моделей**

У циклі перебирається різна кількість сусідів для об’єкта, модель одразу ж тестується на тестових даних, а в масив записується точність моделі при кожному варіанті.

```
from sklearn.neighbors import KNeighborsClassifier
#import metrics model to check the accuracy
from sklearn import metrics

#Try running from R=1 through 25 and record testing accuracy
k_range = range (1,26)
scores = {}
scores_list = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred=knn.predict(X_test)
    scores[k] = metrics.accuracy_score(y_test,y_pred)
    scores_list.append(metrics.accuracy_score(y_test,y_pred))
```

Із результатів можна зробити висновок, що найточніша модель за 1-3, 5 сусідах.

```
scores_list
```

```
[0.7622107969151671,
 0.7095115681233933,
 0.7069408740359897,
 0.6838046272493573,
 0.7030848329048843,
 0.6786632390745502,
 0.6773778920308483,
 0.6670951156812339,
 0.6658097686375322,
 0.6658097686375322,
 0.6555269922879178,
 0.6491002570694088,
 0.6465295629820051,
 0.6439588688946015,
 0.6388174807197944,
 0.6555269922879178,
 0.6529562982005142,
 0.6491002570694088,
 0.6426735218508998,
```

З Random Forest навчається друга модель на навчальних даних, одразу ж тестується на тестових і знаходиться її точність.

```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
# Create a Random forest Classifier
clf = RandomForestClassifier(n_estimators = 100)

# Train the model using the training sets
clf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
y_pred = clf.predict(X_test)
accuracy = clf.score(X_test, y_test)
accuracy
```

```
0.877892030848329
```

5. Висновки щодо якості побудованих моделей

Як результат, можна зазначити, що точність моделі Random Forest вища за модель з KNN. Якщо порівняти цю ж модель з результатами тестів з іншими класифікаторами, можна визначити, що є класифікатори, що дають вищу точність (Multi-layer Perceptron та GaussianProcess), проте Random Forest все ще має найвищу точність прогнозування.

```
Nearest Neighbors - 0.7075835475578406
Linear SVM - 0.7506426735218509
RBF SVM - 0.4588688946015424
Gaussian Process - 0.8084832904884319
Decision Tree - 0.5880462724935732
```

```
Neural Net - 0.8116966580976864
AdaBoost - 0.4138817480719794
Naive Bayes - 0.27892030848329047
QDA - 0.570694087403599
```

6. Результати аналізу

Отже, як виявилося, можна спрогнозувати жанр гри, базуючись на даних продажах чи рейтингу, й з доволі високою точністю. Дана задача була виконана за допомогою класифікації. Початкові дані були відібрані відповідно до потреб подальшої роботи, була виконана підготовка даних експертним шляхом, вирішена проблема недостачі декотрих даних.

При розділі даних на навчальний та тестувальний набір було враховано, що дані з початкового датафрейму - категоріальні, та були перекодовані в чисельні.

Було визначено, що в жаному впадку з двох проаналізованих моделей Random Forest краще прогнозує точність прогнозування, ніж KNN.

ПЕРЕЛІК ПОСИЛАНЬ

1) https://www.kaggle.com/datasets/imohtn/video-games-rating-by-esrb

2) https://www.kaggle.com/datasets/kendallgillies/video-game-sales-and-ratings?resource=download

3) https://stackoverflow.com/questions/29177498/python-pandas-replace-nan-in-one-column-with-value-from-corresponding-row-of-sec

4) https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html
https://pythonguides.com/matplotlib-multiple-bar-chart/

5) https://scikit-learn.org/dev/modules/svm.html#classification

6) https://stackoverflow.com/questions/51070985/find-out-the-percentage-of-missing-values-in-each-column-in-the-given-dataset

7) https://pythonru.com/uroki/sklearn-random-forest

8) https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

9) https://runebook.dev/ru/docs/scikit_learn/modules/preprocessing

10) https://www.kaggle.com/code/snanilim/video-games-sales-analysis-and-visualization

11) лекції курсу «Аналіз даних в інформаційних системах»

ДОДАТКИ

Підготовка даних за допомогою LabelEncoder, OneHotEncoder та StandardScaler:

```
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
# creating instance of one-hot-encoder
enc = OneHotEncoder(handle_unknown='ignore')
# passing bridge-types-cat column (label encoded values of bridge_types)
enc_df = pd.DataFrame(enc.fit_transform(fact_df[['Platform']]).toarray())
# merge with main df bridge_df on key values
f_df = fact_df.join(enc_df)
f_df = f_df.drop(columns=["Platform", "Rating"])
f = f_df.fillna(0)

f['Genre'] = LabelEncoder().fit_transform(f['Genre'])
f['Publisher'] = LabelEncoder().fit_transform(f['Publisher'])
f['esrb_rating'] = LabelEncoder().fit_transform(f['esrb_rating'].astype('str'))

from sklearn.preprocessing import StandardScaler

y=f["Genre"].values

X = f.drop(columns=["Genre", "Name", "title"])
X = StandardScaler().fit_transform(X)
```

Навчання моделі з KNN:

```
from sklearn.neighbors import KNeighborsClassifier
#import metrics model to check the accuracy
from sklearn import metrics

#Try running from R=1 through 25 and record testing accuracy
k_range = range (1,26)
scores = {}
scores_list = []
for k in k_range:
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  y_pred=knn.predict(X_test)
  scores[k] = metrics.accuracy_score(y_test,y_pred)
  scores_list.append(metrics.accuracy_score(y_test,y_pred))
```

Навчання моделі з Random Forest:

```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
# Create a Random forest Classifier
clf = RandomForestClassifier(n_estimators = 100)

# Train the model using the training sets
clf.fit(X_train, y_train)
```

Оцінка по іншим класифікаторам:

```
from sklearn.gaussian_process.kernels import RBF
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.svm import SVC

classifiers = [
  SVC(kernel="linear", C=0.025),
  SVC(gamma=2, C=1),
  GaussianProcessClassifier(1.0 * RBF(1.0)),
  DecisionTreeClassifier(max_depth=5),
  MLPClassifier(alpha=1, max_iter=1000),
  AdaBoostClassifier(),
  GaussianNB(),
  QuadraticDiscriminantAnalysis(),
]
linearly_separable = (X, y)

datasets = [
  linearly_separable,
]
names = [
  "Linear SVM",
  "RBF SVM",
```

```
    "Gaussian Process",
    "Decision Tree",
    "Neural Net",
    "AdaBoost",
    "Naive Bayes",
    "QDA"
]
i = 1
# iterate over datasets
for ds_cnt, ds in enumerate(datasets):
    # preprocess dataset, split into training and test part
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.4, random_state=42
    )
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

    i += 1

    # iterate over classifiers
    for name, clf in zip(names, classifiers):
        clf.fit(X_train, y_train)
        score = clf.score(X_test, y_test)
        print(f"{name} - {score}")

        i += 1
```