

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №41

ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

К.т.н., доц.

Е.Л. Турнецкая

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

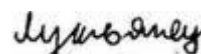
Анализ связей между признаками двумерного набора данных

по курсу: Методология и технология проектирования информационных
систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

М320М



П. Е. Лукьянец

подпись, дата

инициалы, фамилия

Санкт-Петербург 2023

Цель работы: изучение связи между признаками двумерного набора данных.

Ход выполнения работы

Для работы был выбран шестой датасет из списка под названием «6 games». В данном датасете представлена информация о видеоиграх: названии, платформе, годе выпуска, жанре, продажах в разных регионах и оценках. Чтобы начать работу, импортируем библиотеку, а затем считываем CSV файл. Выведем первые 20 строк с помощью метода head. Часть данных этого датасета представлена на рисунке 1.

Рисунок 1 – используемый датасет

Для работы с ним использовалась библиотека Pandas.

Работа была выполнена при помощи Visual Studio Code, а также Jupyter Notebook.

Ссылка на GitHub репозиторий с файлами:
<https://github.com/NinjaCaratist/MTPIS>

Чтобы начать работу, импортируем библиотеку, а затем считываем CSV файл.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

df = pd.read_csv("6games.csv")
```

Рисунок 2 – скриншот кода

Выведем первые 20 строк с помощью метода head. На рисунке 3 показан код, а на рисунке 4 – результат его работы.

```
print(df.head(20))
```

Рисунок 3 – скриншот кода

	Name	Platform	Year_of_Release
0	Wii Sports	Wii	2006.0
1	Super Mario Bros.	NES	1985.0
2	Mario Kart Wii	Wii	2008.0
3	Wii Sports Resort	Wii	2009.0
4	Pokemon Red/Pokemon Blue	GB	1996.0
5	Tetris	GB	1989.0
6	New Super Mario Bros.	DS	2006.0
7	Wii Play	Wii	2006.0
8	New Super Mario Bros. Wii	Wii	2009.0
9	Duck Hunt	NES	1984.0
10	Nintendogs	DS	2005.0
11	Mario Kart DS	DS	2005.0
12	Pokemon Gold/Pokemon Silver	GB	1999.0
13	Wii Fit	Wii	2007.0
14	Kinect Adventures!	X360	2010.0
15	Wii Fit Plus	Wii	2009.0
16	Grand Theft Auto V	PS3	2013.0
17	Grand Theft Auto: San Andreas	PS2	2004.0
18	Super Mario World	SNES	1990.0
19	Brain Age: Train Your Brain in Minutes a Day	DS	2005.0

	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	\
0	Sports	41.36	28.96	3.77	8.45	76.0	
1	Platform	29.08	3.58	6.81	0.77	NaN	
2	Racing	15.68	12.76	3.79	3.29	82.0	
3	Sports	15.61	10.93	3.28	2.95	80.0	
4	Role-Playing	11.27	8.89	10.22	1.00	NaN	
5	Puzzle	23.20	2.26	4.22	0.58	NaN	
6	Platform	11.28	9.14	6.50	2.88	89.0	
7	Misc	13.96	9.18	2.93	2.84	58.0	
8	Platform	14.44	6.94	4.70	2.24	87.0	
9	Shooter	26.93	0.63	0.28	0.47	NaN	
10	Simulation	9.05	10.95	1.93	2.74	NaN	
11	Racing	9.71	7.47	4.13	1.90	91.0	
12	Role-Playing	9.00	6.18	7.20	0.71	NaN	
13	Sports	8.92	8.03	3.60	2.15	80.0	
14	Misc	15.00	4.89	0.24	1.69	61.0	
15	Sports	9.01	8.49	2.53	1.77	80.0	
16	Action	7.02	9.09	0.98	3.96	97.0	
17	Action	9.43	0.40	0.41	10.57	95.0	
18	Platform	12.78	3.75	3.54	0.55	NaN	
19	Misc	4.74	9.20	4.16	2.04	77.0	

	User_Score	Rating
0	8	E
1	NaN	NaN
2	8.3	E
3	8	E
4	NaN	NaN
5	NaN	NaN
6	8.5	E
7	6.6	E
8	8.4	E
9	NaN	NaN
10	NaN	NaN
11	8.6	E
12	NaN	NaN
13	7.7	E
14	6.3	E
15	7.4	E
16	8.2	M
17	9	M
18	NaN	NaN
19	7.9	E

Рисунок 4 – результат вывода

Как уже было сказано, данная таблица содержит информацию о продаваемых автомобилях. Предметная область – автомобили и продажи. Опишем колонки подробнее:

Name – Название

Platform - Платформа

Year – год выпуска

Genre - Жанр

NA_Sales, EU_sales, JP_Sales, Other_Sales Продажи в разных регионах

Critic_Store - Оценка

Теперь с помощью метода «.info» оценим данные. Этот метод возвращает название столбцов, типы данных, количество ненулевых объектов каждом столбце. Результат работы метода представлен на рисунке 5.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                   16713 non-null  object
1   Platform               16715 non-null  object
2   Year_of_Release        16446 non-null  float64
3   Genre                  16713 non-null  object
4   NA_sales               16715 non-null  float64
5   EU_sales               16715 non-null  float64
6   JP_sales               16715 non-null  float64
7   Other_sales            16715 non-null  float64
8   Critic_Score           8137 non-null   float64
9   User_Score             10014 non-null  object
10  Rating                 9949 non-null   object
dtypes: float64(6), object(5)
memory usage: 1.4+ MB

```

Рисунок 5 – результат вывода

Теперь выведем на экран названия столбцов с помощью `df.columns`.
Названия всех колонок приемлимы.

```
print(df.columns)
```

Рисунок 6 – скриншот кода

```

Index(['Name', 'Platform', 'Year_of_Release', 'Genre', 'NA_sales', 'EU_sales',
       'JP_sales', 'Other_sales', 'Critic_Score', 'User_Score', 'Rating'],
      dtype='object')

```

Рисунок 7 – результат вывода

Найдём пропуски и устраним их. При помощи метода «`isna`» найдём все пропуски в таблице, а также при помощи `sum` выведем количество пропусков в каждом столбце. Как мы можем увидеть присутствуют пропуски в столбцах названия, жанра, года выпуска, оценок и рейтинга. Так как важным столбцом является лишь название, строки без информации в нём не имеют смысла, поэтому их стоит удалить при помощи метода «`dropna`». Однако можно не удалять строки без данных в других столбцах. Поэтому при помощи метода `fillna` заполним пустые значения оценок нулями, жанра и рейтинга строкой

'none', а года выпуска 1950 годом, в котором ещё не выпускались игры. Проверяем пропуски еще раз и их больше нет. Код представлен на рисунке 8. Проверяем пропуски еще раз и их больше нет.

```
print(df.isna())
print(df.isna().sum())
df = df.dropna(subset=['Name'])
df['Genre'] = df['Genre'].fillna('none')
df['Year_of_Release'] = df['Year_of_Release'].fillna(1950)
df['Rating'] = df['Rating'].fillna('none')
tmp_df = df
df['Critic_Score'] = df['Critic_Score'].fillna(0)
df['User_Score'] = df['User_Score'].fillna(0)

print(df.isna().sum())
```

Рисунок 8 – скриншот кода

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	\
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	
16710	False	False	False	False	False	False	False	
16711	False	False	False	False	False	False	False	
16712	False	False	False	False	False	False	False	
16713	False	False	False	False	False	False	False	
16714	False	False	False	False	False	False	False	

	Other_sales	Critic_Score	User_Score	Rating
0	False	False	False	False
1	False	True	True	True
2	False	False	False	False
3	False	False	False	False
4	False	True	True	True
...
16710	False	True	True	True
16711	False	True	True	True
16712	False	True	True	True
16713	False	True	True	True
16714	False	True	True	True

[16715 rows x 11 columns]	
Name	2
Platform	0
Year_of_Release	269
Genre	2
NA_sales	0
EU_sales	0
JP_sales	0
Other_sales	0
Critic_Score	8578
User_Score	6701
Rating	6766
dtype: int64	
Name	0
Platform	0
Year_of_Release	0
Genre	0
NA_sales	0
EU_sales	0
JP_sales	0
Other_sales	0
Critic_Score	0
User_Score	0
Rating	0

Рисунок 9 – результат вывода

Проверим данные на наличие дубликатов – рисунок 10. Полностью повторяющихся строк нет, результат представлен на рисунке 11. Уникальным значением в данном наборе является название, а точнее название, платформа и год выпуска, только в таком сочетании строка уникальна. Получается составной первичный ключ. Уникальных значений по имени 11559.

```
print("Количество дубликатов: " + str(df.duplicated().sum()))  
df.info()
```

Рисунок 10 – скриншот кода

```
Количество дубликатов: 0
```

Рисунок 11 – результат вывода

При помощи метода `drop_duplicates` удаляем дубликаты по столбцам Названия, платформы и года – рисунок 12. После переиндексации и вывода информации видим, что на одну запись стало меньше - рисунок 13. В записях остальных колонок ошибок нет.

```
print(df['Name'].nunique(), "\n")  
df = df.drop_duplicates(subset=['Name', 'Platform', 'Year_of_Release']).reset_index()  
df.info()  
print( "\n")  
print(df['Year_of_Release'].unique())  
print(df['Platform'].unique())  
print(df['Genre'].unique())  
print(df['Rating'].unique())
```

Рисунок 12 – скриншот кода


```

---
0  Name          16713 non-null object
1  Platform      16713 non-null object
2  Year_of_Release 16713 non-null float64
3  Genre         16713 non-null object
4  NA_sales      16713 non-null float64
5  EU_sales      16713 non-null float64
6  JP_sales      16713 non-null float64
7  Other_sales   16713 non-null float64
8  Critic_Score  16713 non-null float64
9  User_Score    16713 non-null object
10 Rating        16713 non-null object
dtypes: float64(6), object(5)
memory usage: 1.5+ MB
11559

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16712 entries, 0 to 16711
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   index           16712 non-null  int64
1   Name            16712 non-null  object
2   Platform        16712 non-null  object
3   Year_of_Release 16712 non-null  float64
4   Genre           16712 non-null  object
5   NA_sales        16712 non-null  float64
6   EU_sales        16712 non-null  float64
7   JP_sales        16712 non-null  float64
8   Other_sales     16712 non-null  float64
9   Critic_Score    16712 non-null  float64
10  User_Score      16712 non-null  object
11  Rating          16712 non-null  object
dtypes: float64(6), int64(1), object(5)
memory usage: 1.5+ MB

[2006. 1985. 2008. 2009. 1996. 1989. 1984. 2005. 1999. 2007. 2010. 2013.
 2004. 1990. 1988. 2002. 2001. 2011. 1998. 2015. 2012. 2014. 1992. 1997.
 1993. 1994. 1982. 2016. 2003. 1986. 2000. 1950. 1995. 1991. 1981. 1987.
 1980. 1983.]
['Wii' 'NES' 'GB' 'DS' 'X360' 'PS3' 'PS2' 'SNES' 'GBA' 'PS4' '3DS' 'N64'
 'PS' 'XB' 'PC' '2600' 'PSP' 'XOne' 'WiiU' 'GC' 'GEN' 'DC' 'PSV' 'SAT'
 'SCD' 'WS' 'NG' 'TG16' '3DO' 'GG' 'PCFX']
['Sports' 'Platform' 'Racing' 'Role-Playing' 'Puzzle' 'Misc' 'Shooter'
 'Simulation' 'Action' 'Fighting' 'Adventure' 'Strategy']
['E' 'none' 'M' 'T' 'E10+' 'K-A' 'AO' 'EC' 'RP']

```

Рисунок 13 – результат вывода

Проверим все ли типы данных соответствуют действительности. Все столбцы, кроме года выпуска и оценки пользователей соответствуют своему типу. Поэтому при помощи метода «to_datetime» изменяем тип на временной. А при помощи метода "to numeric" приводим оценки к формату с плавающей точкой, при этом прописывая условия о не превращающихся значениях в NaN

(см. рис. 14). Затем эти значения приравниваем к нулю и выводим информацию (см. рис. 15).

```

• df['Year_of_Release'] = pd.to_datetime(df['Year_of_Release'], format='%Y')
  df['User_Score'] = pd.to_numeric(df['User_Score'], errors='coerce')
  df['User_Score'] = df['User_Score'].fillna(0)
  print(df['Year_of_Release'].unique())
  print(df['User_Score'].unique())
  df.info()

```

Рисунок 14 – скриншот кода

```

['2006-01-01T00:00:00.000000000' '1985-01-01T00:00:00.000000000'
 '2008-01-01T00:00:00.000000000' '2009-01-01T00:00:00.000000000'
 '1996-01-01T00:00:00.000000000' '1989-01-01T00:00:00.000000000'
 '1984-01-01T00:00:00.000000000' '2005-01-01T00:00:00.000000000'
 '1999-01-01T00:00:00.000000000' '2007-01-01T00:00:00.000000000'
 '2010-01-01T00:00:00.000000000' '2013-01-01T00:00:00.000000000'
 '2004-01-01T00:00:00.000000000' '1990-01-01T00:00:00.000000000'
 '1988-01-01T00:00:00.000000000' '2002-01-01T00:00:00.000000000'
 '2001-01-01T00:00:00.000000000' '2011-01-01T00:00:00.000000000'
 '1998-01-01T00:00:00.000000000' '2015-01-01T00:00:00.000000000'
 '2012-01-01T00:00:00.000000000' '2014-01-01T00:00:00.000000000'
 '1992-01-01T00:00:00.000000000' '1997-01-01T00:00:00.000000000'
 '1993-01-01T00:00:00.000000000' '1994-01-01T00:00:00.000000000'
 '1982-01-01T00:00:00.000000000' '2016-01-01T00:00:00.000000000'
 '2003-01-01T00:00:00.000000000' '1986-01-01T00:00:00.000000000'
 '2000-01-01T00:00:00.000000000' '1950-01-01T00:00:00.000000000'
 '1995-01-01T00:00:00.000000000' '1991-01-01T00:00:00.000000000'
 '1981-01-01T00:00:00.000000000' '1987-01-01T00:00:00.000000000'
 '1980-01-01T00:00:00.000000000' '1983-01-01T00:00:00.000000000']
[8.  0.  8.3 8.5 6.6 8.4 8.6 7.7 6.3 7.4 8.2 9.  7.9 8.1 8.7 7.1 3.4 5.3
 4.8 3.2 8.9 6.4 7.8 7.5 2.6 7.2 9.2 7.  7.3 4.3 7.6 5.7 5.  9.1 6.5 8.8
 6.9 9.4 6.8 6.1 6.7 5.4 4.  4.9 4.5 9.3 6.2 4.2 6.  3.7 4.1 5.8 5.6 5.5
 4.4 4.6 5.9 3.9 3.1 2.9 5.2 3.3 4.7 5.1 3.5 2.5 1.9 3.  2.7 2.2 2.  9.5
 2.1 3.6 2.8 1.8 3.8 1.6 9.6 2.4 1.7 1.1 0.3 1.5 0.7 1.2 2.3 0.5 1.3 0.2
 0.6 1.4 0.9 1.  9.7]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16712 entries, 0 to 16711
Data columns (total 12 columns):
#   Column              Non-Null Count  Dtype
---  -
0   index                16712 non-null  int64
1   Name                 16712 non-null  object
2   Platform             16712 non-null  object
3   Year_of_Release      16712 non-null  datetime64[ns]
4   Genre                16712 non-null  object
5   NA_sales             16712 non-null  float64
6   EU_sales             16712 non-null  float64
7   JP_sales             16712 non-null  float64
8   Other_sales          16712 non-null  float64
9   Critic_Score         16712 non-null  float64
10  User_Score           16712 non-null  float64
11  Rating               16712 non-null  object
dtypes: datetime64[ns](1), float64(6), int64(1), object(4)
memory usage: 1.5+ MB

```

Рисунок 15 – результат вывода

Создадим сводную таблицу при помощи метода «data_pivot», код представлен на рисунке 16. Индексацию возьмём по платформам, а колонки по жанрам игр. Подсчёт будет по сумме продаж в Северной Америке. Таким образом, получится таблица, показывающая, на какой жанр и с какой консолью приходится больше всего сумма продаж. Это шутеры на xbox360, что ожидаемо. Это можно увидеть на рисунке 17.

```
data_pivot = df.pivot_table(index=['Platform'], columns='Genre', values='NA_sales', aggfunc='sum')
print(data_pivot)
```

Рисунок 16 – скриншот кода

TG16	NaN	0.00	NaN	NaN	NaN	NaN	NaN
WS	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wii	68.07	10.86	12.83	118.82	47.23	9.10	30.95
WiiU	9.69	0.07	3.28	5.50	9.94	0.62	3.47
X360	141.66	8.44	24.61	63.66	6.09	0.62	33.10
XB	34.34	2.22	10.01	7.19	6.58	0.32	21.48
XOne	21.37	1.27	1.65	4.77	0.51	NaN	4.36
Genre	Role-Playing	Shooter	Simulation	Sports	Strategy		
Platform							
2600	NaN	24.68	0.42	3.22	NaN		
3DO	NaN	NaN	0.00	NaN	NaN		
3DS	23.76	0.66	7.58	1.66	0.78		
DC	0.00	0.00	0.00	2.14	NaN		
DS	46.14	6.46	67.42	15.48	8.67		
GB	28.71	0.40	0.00	3.64	1.49		
GBA	28.29	2.35	3.61	10.30	4.21		
GC	7.91	10.01	5.37	18.37	2.23		
GEN	0.00	0.00	NaN	2.70	0.00		
GG	NaN	NaN	NaN	NaN	NaN		
N64	1.34	13.96	5.77	22.07	4.64		
NES	1.14	29.03	NaN	7.00	NaN		
NG	NaN	NaN	NaN	0.00	NaN		
PC	17.35	18.81	20.08	0.38	19.63		
PCFX	0.00	NaN	NaN	NaN	NaN		
PS	20.73	21.16	6.79	64.02	6.65		
PS2	32.85	57.53	19.37	134.12	3.57		
PS3	29.93	82.03	4.30	60.91	1.88		
PS4	9.34	32.66	0.21	19.78	0.17		
PSP	11.46	10.26	1.90	17.54	2.67		
PSV	2.24	1.48	0.01	1.15	0.03		
SAT	0.00	0.00	0.00	0.00	0.00		
SCD	0.00	NaN	NaN	NaN	0.00		
SNES	2.29	2.67	1.39	2.10	0.30		
TG16	NaN	0.00	NaN	NaN	NaN		
WS	0.00	NaN	NaN	NaN	0.00		
Wii	5.48	18.22	23.30	149.72	2.32		
WiiU	1.06	2.34	0.14	1.50	0.49		
X360	44.75	174.51	8.51	90.02	6.50		
XB	9.89	46.16	5.43	41.02	2.05		
XOne	6.15	36.86	0.32	15.59	0.27		

Рисунок 17 – результат вывода

Теперь при помощи метода `describe` выведем описание статистики по всем атрибутам. После этого построим несколько графиков: по количеству значений различных атрибутов, по количеству продаж в Америке в различные года, а также по зависимости переменных друг от друга.

```
print(df.describe(include='all',datetime_is_numeric=True))
df.hist()
plt.show()

df.plot(x='Year_of_Release', y='NA_sales', kind='scatter')
plt.show()

pd.plotting.scatter_matrix(df)
plt.show()
```

Рисунок 18 – скриншот кода

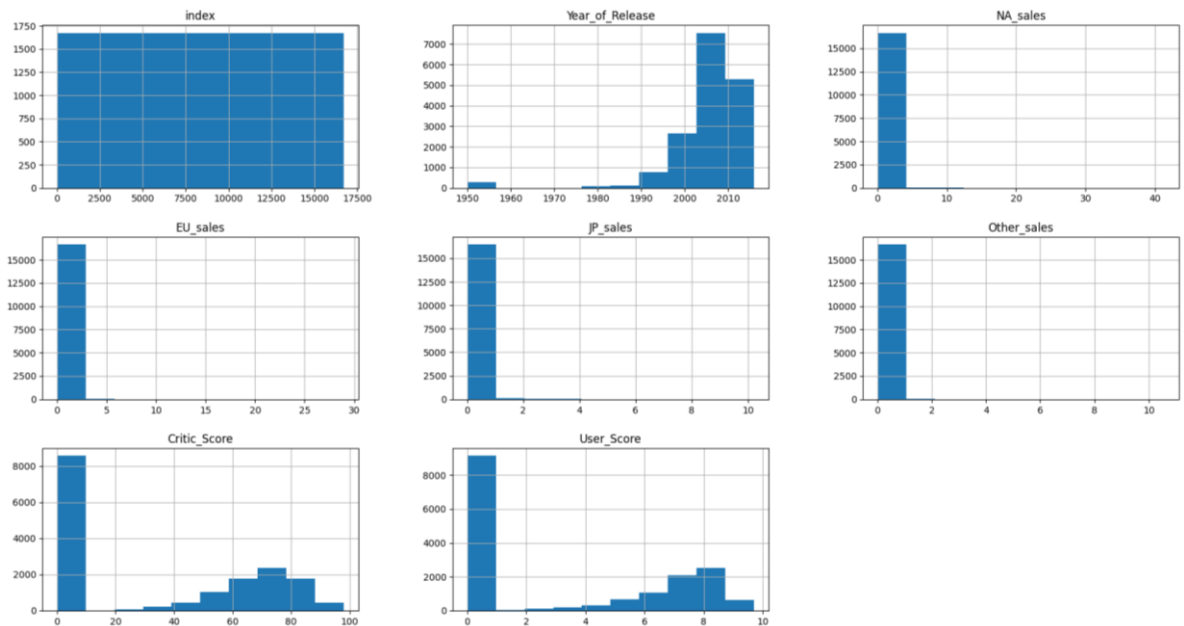


Рисунок 19 – построенные гистограммы

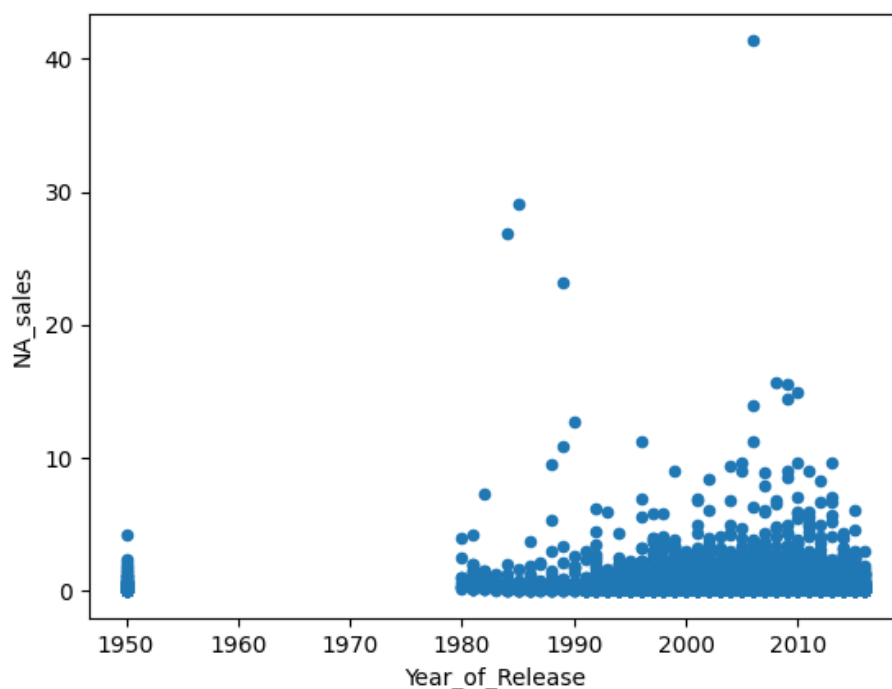


Рисунок 20 – построенный график

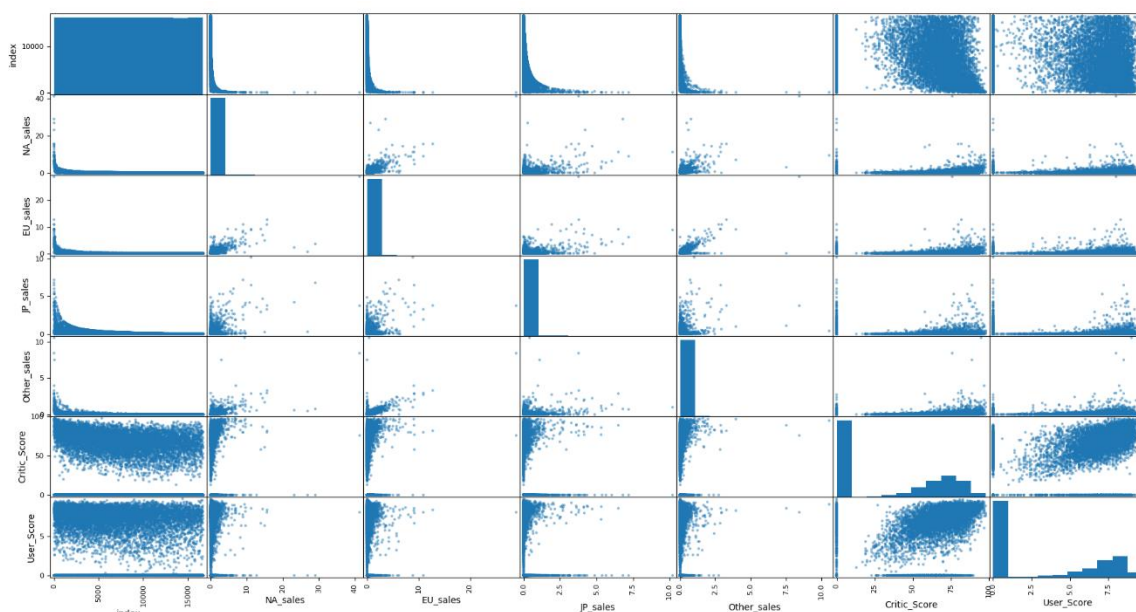


Рисунок 21 – построенная матрица зависимостей

Теперь исследуем взаимосвязь между переменными с помощью оценки коэффициента корреляции и ковариации, а также построим тепловую карту по этим значениям. В созданном заранее временном дата фрейме, удалим строки с отсутствующими значениями оценок для лучшей оценки корреляции и

ковариации. Интерпретируем результаты. По результатам ковариации можно сказать, что большинство числовых параметров изменяются в одном направлении, кроме продаж в Японии от оценок критиков и пользователей. По результатам корреляции, что самая близкая к линейной является зависимость оценок пользователей от оценок критиков. Также немалый коэффициент зависимости между собой имеют продажи в СА, Европе и остальном мире. У Японцев какой-то свой отдельный мир.

Один столбец с целевым признаком выделить сложно, скорее можно выделить группу атрибутов, связанных с продажами, так как главным для любого продукта является его окупаемость. Продажи, как показывает корреляция, не сильно зависят от оценок. Поэтому следует предположить, что влияют другие показатели, такие как: маркетинг, популярность серии или выпускающей корпорации, выпуск на разных платформах и т.д.

```
tmp_df = tmp_df.dropna(subset=['Critic_Score'])
tmp_df['Year_of_Release'] = pd.to_datetime(tmp_df['Year_of_Release'], format='%Y')
tmp_df['User_Score'] = pd.to_numeric(tmp_df['User_Score'], errors='coerce')
tmp_df = tmp_df.dropna(subset=['User_Score'])

corr = tmp_df.corr(method = 'pearson')
print(corr)
sb.heatmap(corr, cmap="Blues", annot=True)
plt.show()

cov = tmp_df.cov()
print(cov)
sb.heatmap(cov, cmap="Blues", annot=True)
plt.show()
```

Рисунок 22 – скриншот кода

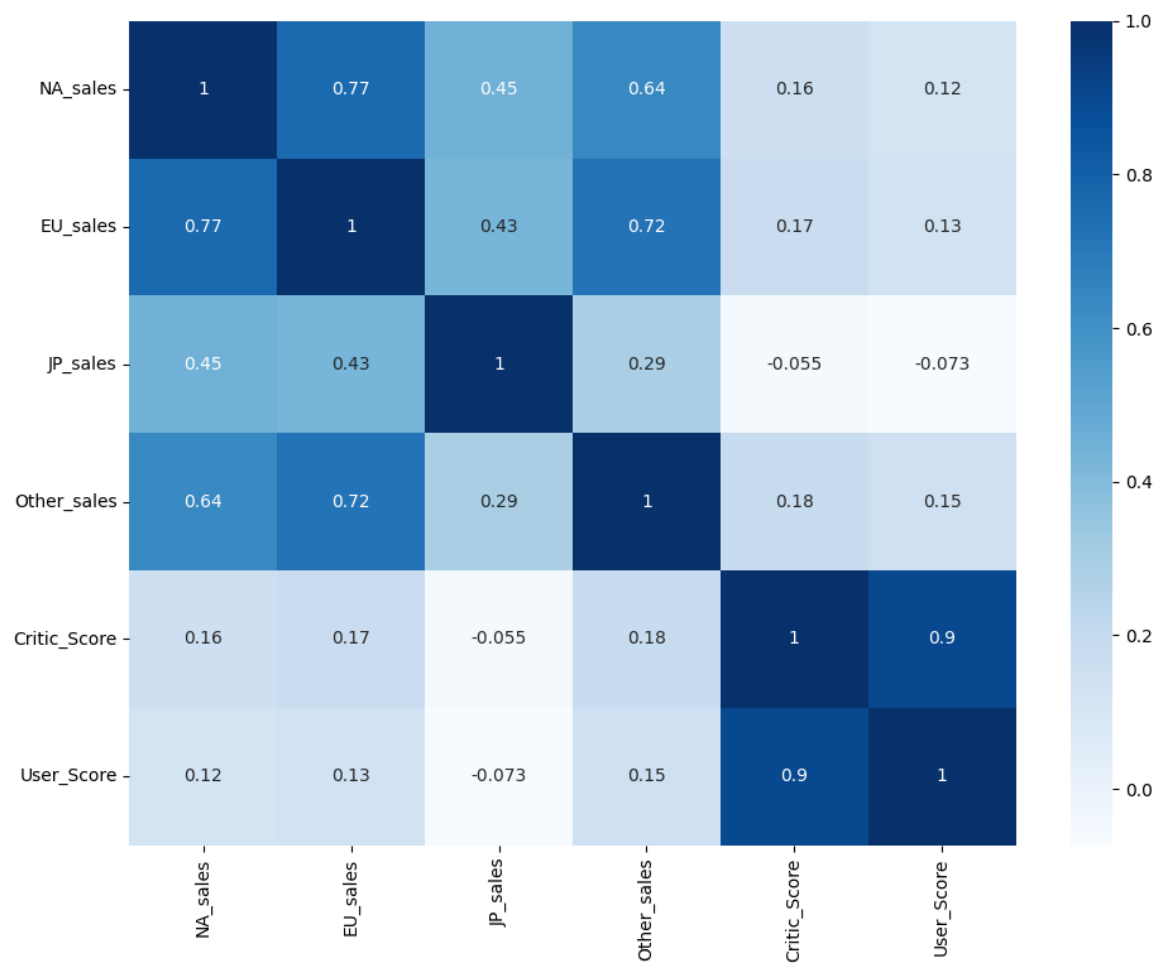


Рисунок 23 – построенная тепловая карта корреляции

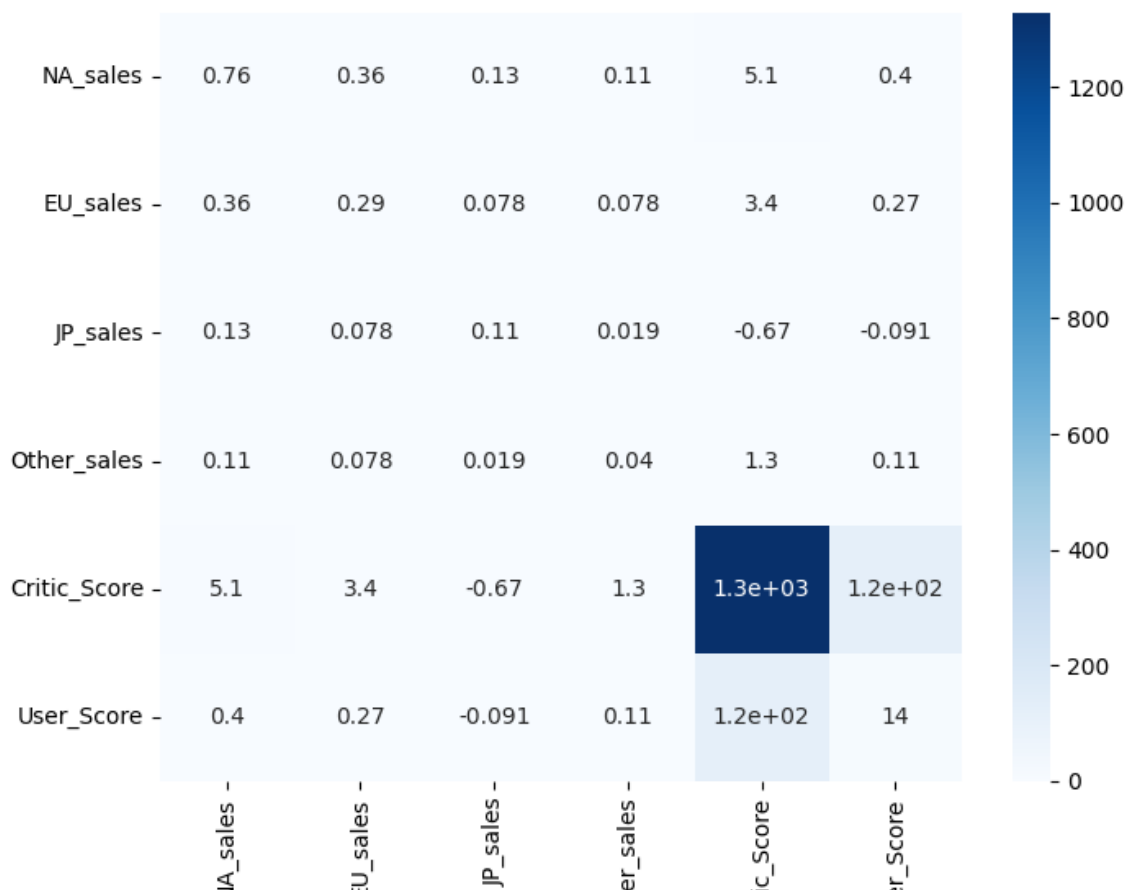


Рисунок 24 – построенная тепловая карта ковариации

Вывод: Вывод: Таким образом, в ходе выполнения лабораторной работы был выбран и описан выбранный датасет про продажи и оценки консольных игр, изучен интерфейс и возможности Jupyter Notebook, изучены базовые функции библиотеки Pandas и разработана программа, которая считывает данные, выводит о них информацию, удаляет дубликаты, пропуски, изменяет тип данных и создаёт сводную таблицу и графики, а также высчитывает коэффициенты корреляции и ковариации и строит тепловые карты по ним.