ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc182695312)

[Актуальность темы 3](#_Toc182695313)

[Цель работы 3](#_Toc182695314)

[Задачи работы 3](#_Toc182695315)

[Объект исследования 4](#_Toc182695316)

[Предмет исследования 4](#_Toc182695317)

[ГЛАВА 1. ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc182695318)

[Понятие визуализации данных и ее важность 5](#_Toc182695319)

[Цели и задачи визуализации данных 5](#_Toc182695320)

[Роль визуализации данных в анализе данных и принятии решений 5](#_Toc182695321)

[Обзор существующих библиотек для визуализации данных 6](#_Toc182695322)

[ГЛАВА 2. МЕТОДИКА ИССЛЕДОВАНИЯ 9](#_Toc182695323)

[Выбор критериев сравнения 9](#_Toc182695324)

[ГЛАВА 3. ИССЛЕДОВАНИЕ 10](#_Toc182695325)

[Подготовка данных 10](#_Toc182695326)

[Создание визуализаций matplotlib 11](#_Toc182695327)

[Создание визуализаций seaborn 13](#_Toc182695328)

[Создание визуализаций plotly 17](#_Toc182695329)

[ЗАКЛЮЧЕНИЕ 22](#_Toc182695330)

[Функциональность 22](#_Toc182695331)

[*Создание различных типов графиков* 22](#_Toc182695332)

[*Настройка внешнего вида графиков* 23](#_Toc182695333)

[*Добавление интерактивности* 23](#_Toc182695334)

[Удобство использования 24](#_Toc182695335)

[*Простота и интуитивность* 24](#_Toc182695336)

[*Документация и примеры* 24](#_Toc182695337)

[*Поддержка сообщества* 25](#_Toc182695338)

[Производительность 25](#_Toc182695339)

[*Скорость работы* 25](#_Toc182695340)

[*Потребление ресурсов* 26](#_Toc182695341)

[СПИСОК ЛИТЕРАТУРЫ 27](#_Toc182695342)

[ПРИЛОЖЕНИЕ 29](#_Toc182695343)

ВВЕДЕНИЕ

Актуальность темы

В современном мире данные играют ключевую роль в принятии решений во многих сферах, начиная от бизнеса и заканчивая научными исследованиями. Визуализация данных является неотъемлемой частью анализа и интерпретации информации, позволяя наглядно представить сложные данные и выявить скрытые закономерности. Выбор подходящей библиотеки для визуализации данных имеет решающее значение для эффективности и качества работы.

Цель работы

Целью данной работы является сравнение функциональности, удобства использования и производительности библиотек Matplotlib, Seaborn и Plotly для визуализации данных.

Задачи работы

1. Провести обзор и анализ литературы по теме.
   1. Изучить существующие исследования и статьи, посвященные визуализации данных и использованию библиотек Matplotlib, Seaborn и Plotly.
   2. Определить основные подходы и методы визуализации данных.
2. Описать основные возможности и особенности библиотек Matplotlib, Seaborn и Plotly.
   1. Проанализировать функциональные возможности каждой библиотеки.
   2. Выявить преимущества и недостатки каждой из библиотек.
3. Создать набор визуализаций с использованием каждой из библиотек.
   1. Разработать примеры визуализаций, демонстрирующие возможности каждой библиотеки.
   2. Сравнить внешний вид и функциональность полученных графиков.
4. Сравнить библиотеки по критериям функциональности, удобства использования и производительности.
   1. Оценить возможности библиотек по созданию различных типов графиков.
   2. Проанализировать удобство использования библиотек, включая наличие документации и поддержку сообщества.
   3. Измерить и сравнить производительность библиотек при создании визуализаций.
5. Сформулировать выводы и рекомендации по выбору библиотеки для конкретных задач.
   1. Обобщить результаты сравнения.
   2. Предложить рекомендации по выбору библиотеки в зависимости от типа задач и требований.

Объект исследования

Объектом исследования являются библиотеки для визуализации данных Matplotlib, Seaborn и Plotly.

Предмет исследования

Предметом исследования являются функциональность, удобство использования и производительность библиотек для визуализации данных.

ГЛАВА 1. ОБЗОР ЛИТЕРАТУРЫ

Понятие визуализации данных и ее важность

Определение визуализации данных

Визуализация данных — это процесс представления информации в графическом формате, который позволяет пользователям быстро и эффективно воспринимать и интерпретировать данные. Визуализация данных использует графические элементы, такие как диаграммы, графики, карты и другие визуальные инструменты, для передачи сложных данных в понятной и доступной форме.

Цели и задачи визуализации данных

Основные цели визуализации данных включают:

1. Повышение понимания данных: Визуализация помогает пользователям быстрее и легче воспринимать сложные данные, выявлять закономерности, аномалии и тренды.
2. Поддержка принятия решений: Визуальные представления данных позволяют принимать более обоснованные и информированные решения.
3. Общение результатов: Визуализация данных является эффективным инструментом для представления результатов анализа аудитории, включая нетехнических пользователей.

Роль визуализации данных в анализе данных и принятии решений

Визуализация данных играет ключевую роль в анализе данных и принятии решений по нескольким причинам:

1. Визуальная интуиция: Люди лучше воспринимают и интерпретируют информацию, представленную визуально, чем в текстовом или числовом формате.
2. Выявление закономерностей: Визуализация помогает выявлять скрытые закономерности, которые могут быть незаметны при анализе данных в табличной форме.
3. Повышение эффективности анализа: Визуальные инструменты позволяют быстро и эффективно анализировать большие объемы данных.
4. Поддержка коммуникации: Визуализация данных помогает эффективно доносить результаты анализа до аудитории, включая нетехнических пользователей.

***Обзор существующих библиотек для визуализации данных***

Существует множество библиотек для визуализации данных на Python, каждая из которых имеет свои преимущества и недостатки. Некоторые из наиболее популярных библиотек включают:

Таблица 1 - Сравнение преимуществ и недостатков

|  |  |
| --- | --- |
| Matplotlib  Это одна из самых популярных и мощных библиотек для визуализации данных на Python. Она предоставляет широкие возможности для создания статических, анимированных и интерактивных графиков. | |
| Преимущества | Недостатки |
| Широкие возможности настройки. | Сложный синтаксис. |
| Богатая документация и поддержка сообщества. | Низкий уровень абстракции. |
| Поддержка различных типов графиков и диаграмм. | Не всегда интуитивно понятный интерфейс. |
| Seaborn  Это библиотека для статистической визуализации данных на Python, основанная на Matplotlib. Она предоставляет высокоуровневый интерфейс для создания привлекательных и информативных статистических графиков. | |
| Высокий уровень абстракции. | Меньше возможностей настройки по сравнению с Matplotlib. |
| Простой и интуитивно понятный интерфейс. | Зависимость от Matplotlib. |
| Встроенная поддержка статистических графиков. | Меньшее разнообразие типов графиков. |
| Plotly  Это библиотека для создания интерактивных графиков и визуализаций данных на Python. Она предоставляет широкий выбор типов графиков и возможностей для создания интерактивных веб-приложений. | |
| Интерактивность. | Более высокие требования к производительности. |
| Широкий выбор типов графиков. | Сложнее в освоении по сравнению с Matplotlib и Seaborn. |
| Поддержка создания интерактивных веб-приложений. | Зависимость от JavaScript для некоторых функций. |

Matplotlib

Основные возможности:

1. Поддержка различных типов графиков, включая линейные графики, гистограммы, диаграммы рассеяния, графики распределения, графики временных рядов и многое другое.
2. Широкие возможности настройки внешнего вида графиков, включая цвета, стили линий, метки осей, заголовки и легенды.
3. Возможность создания многооконных графиков и подграфиков.
4. Поддержка сохранения графиков в различных форматах, таких как PNG, JPEG, SVG и другие.

Seaborn

Основные возможности:

1. Встроенная поддержка статистических графиков, таких как диаграммы рассеяния с регрессионными линиями, графики плотности, графики распределения, графики корреляции и многое другое.
2. Встроенные темы для улучшения внешнего вида графиков.
3. Возможность создания многооконных графиков и подграфиков.
4. Интеграция с Pandas DataFrame.

Plotly

Основные возможности:

1. Поддержка различных типов графиков, включая линейные графики, гистограммы, диаграммы рассеяния, тепловые карты, картограммы и многое другое.
2. Интерактивность, включая масштабирование, панорамирование, наведение курсора и выделение данных.
3. Возможность создания интерактивных веб-приложений с использованием Plotly Dash.
4. Поддержка сохранения графиков в различных форматах, таких как HTML, PNG, JPEG и другие.

ГЛАВА 2. МЕТОДИКА ИССЛЕДОВАНИЯ

Выбор критериев сравнения

Для сравнения библиотек Matplotlib, Seaborn и Plotly для визуализации данных были выбраны следующие критерии:

1. Функциональность:
   1. Типы графиков: Поддержка различных типов графиков, таких как линейные графики, гистограммы, диаграммы рассеяния, тепловые карты, графики распределения, графики временных рядов и другие.
   2. Настройка внешнего вида: Возможности настройки внешнего вида графиков, включая цвета, стили линий, метки осей, заголовки, легенды и другие элементы.
   3. Интерактивность: Поддержка интерактивных элементов, таких как масштабирование, панорамирование, наведение курсора, выделение данных и другие.
2. Удобство использования:
   1. Интуитивность интерфейса: Простота и интуитивность использования библиотеки, включая синтаксис, документацию и примеры.
   2. Встроенные функции: Наличие встроенных функций для создания популярных типов графиков и статистических визуализаций.
   3. Поддержка сообщества: Активность и поддержка сообщества, включая форумы, чаты, документацию и примеры.

ГЛАВА 3. ИССЛЕДОВАНИЕ

Подготовка данных

 Для примеров визуализаций я выбрал данные о продажах и оценках видео-игр с Kaggle Datasets. Данные об оценках игр есть не для всех строк, поэтому сразу оставим только те записи, по которым есть полные данные.

import os  
import pandas as pd  
  
# Получаем абсолютный путь к файлу  
file\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv')  
  
# Загрузка данных из CSV файла и их обработка  
df = pd.read\_csv(file\_path).dropna()  
  
df['User\_Score'] = df.User\_Score.astype('float64')  
df['Year\_of\_Release'] = df.Year\_of\_Release.astype('int64')  
df['User\_Count'] = df.User\_Count.astype('int64')  
df['Critic\_Count'] = df.Critic\_Count.astype('int64')

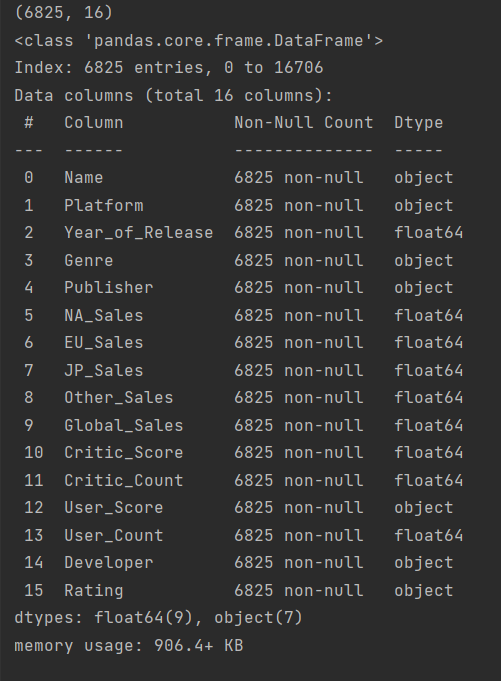


Рисунок 1 - Результат выполнения частичной подготовки данных

Всего в таблице 6825 объектов и 16 признаков для них. Посмотрим на несколько первых записей c помощью метода head, чтобы убедиться, что все распарсилось правильно. Для удобства я оставил только те признаки, которые мы будем в дальнейшем использовать.

useful\_cols = ['Name', 'Platform', 'Year\_of\_Release', 'Genre',  
 'Global\_Sales', 'Critic\_Score', 'Critic\_Count',  
 'User\_Score', 'User\_Count', 'Rating'  
 ]  
print(df[useful\_cols].head())

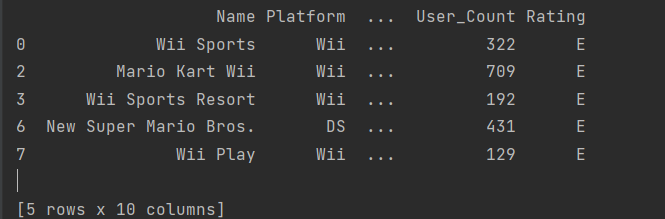


Рисунок 2 - Вид данных после подготовки

Создание визуализаций matplotlib

Прежде чем мы перейдем к рассмотрению методов библиотек seaborn и plotly, обсудим самый простой и зачастую удобный способ визуализировать данные из pandas dataframe — это воспользоваться функцией plot. Для примера построим график продаж видео игр в различных странах в зависимости от года.

Для начала отфильтруем только нужные нам столбцы, затем посчитаем суммарные продажи по годам и у получившегося dataframe вызовем функцию plot без параметров.

Реализация функции plot в pandas основана на библиотеке matplotlib.

# Создание визуализаций matplotlib  
graf1 = df[[x for x in df.columns if 'Sales' in x] +  
 ['Year\_of\_Release']].groupby('Year\_of\_Release').sum().plot()  
plt.show()

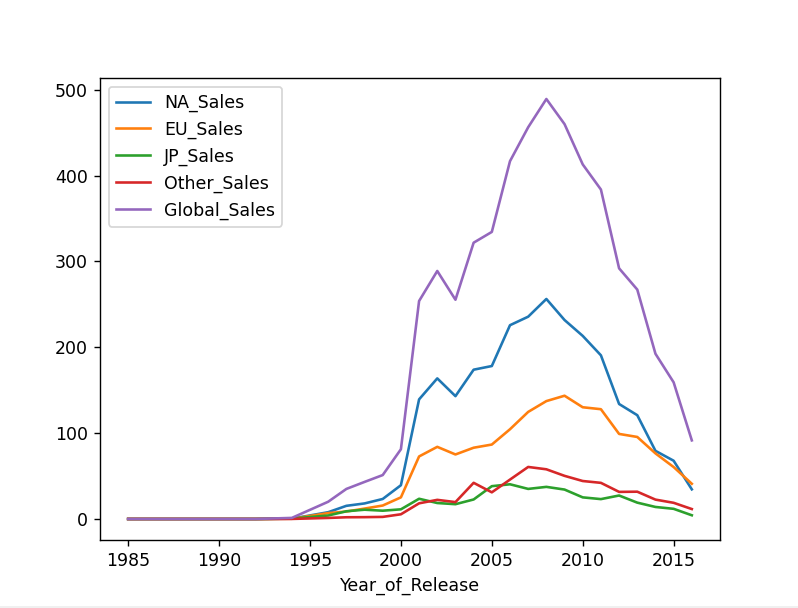


Рисунок 3 - График зависимости суммы от года реализации

C помощью параметра kind можно изменить тип графика, например, на bar chart.

Matplotlib позволяет очень гибко настраивать графики. На графике можно изменить почти все, что угодно, но потребуется порыться в документации и найти нужные параметры. Например, параметра rot отвечает за угол наклона подписей к оси x.

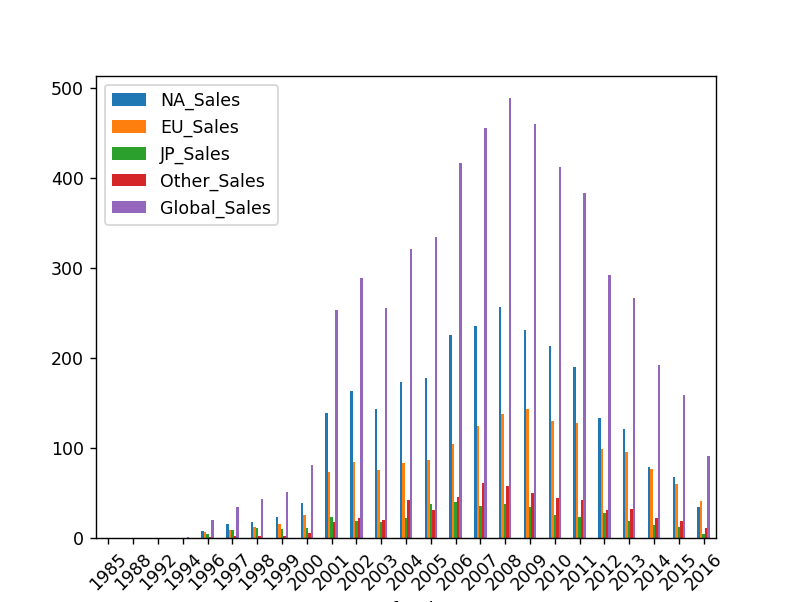


Рисунок 4 - График зависимости суммы от года реализации

Временные ряды с пиками и впадинами позволяют визуализировать динамику продаж игр по годам с выделением пиковых и минимальных значений.

# Группировка данных по годам и суммирование продаж  
yearly\_sales = df.groupby('Year\_of\_Release')['Global\_Sales'].sum()  
  
# Определение пиков и впадин  
peaks = yearly\_sales[yearly\_sales > yearly\_sales.quantile(0.75)]  
valleys = yearly\_sales[yearly\_sales < yearly\_sales.quantile(0.25)]  
  
# Создание временного ряда с пиками и впадинами  
plt.figure(figsize=(12, 6))  
plt.plot(yearly\_sales.index, yearly\_sales.values, label='Продажи по годам')  
plt.scatter(peaks.index, peaks.values, color='red', label='Пики')  
plt.scatter(valleys.index, valleys.values, color='green', label='Впадины')  
plt.title('Динамика продаж игр по годам с выделением пиков и впадин')  
plt.xlabel('Год выпуска')  
plt.ylabel('Общие продажи (в млн. копий)')  
plt.legend()  
plt.grid(True)

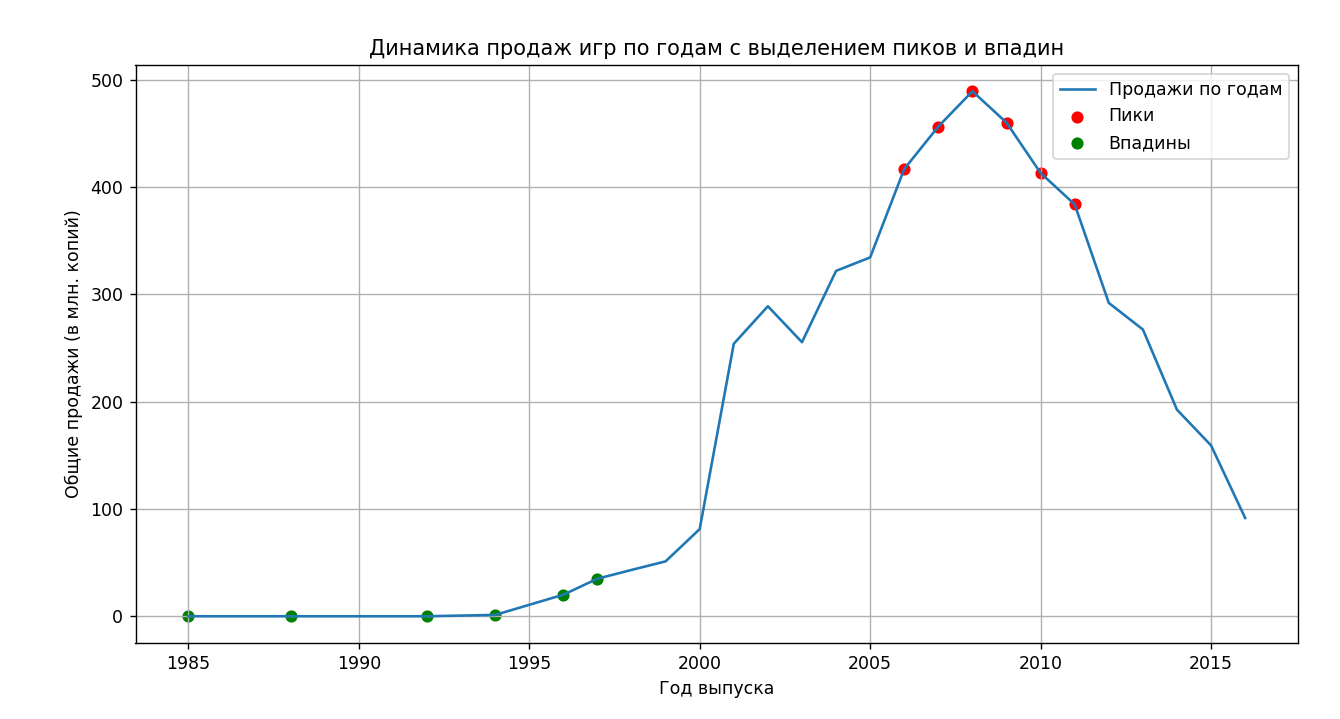


Рисунок 5 - Временные ряды с пиками

Стоит сказать, что библиотека matplotlib предоставляет широкие возможности для создания статических, анимированных и интерактивных графиков. Выше мы рассмотрели «базовые» графики, дальше будем рассматривать некоторые из ее возможностей на примере данных о продажах и оценках видеоигр.

Диаграмма площади Unstacked позволяет визуализировать соотношение продаж игр по жанрам на протяжении лет. Это помогает выявить изменения в популярности жанров с течением времени.

from main import df  
import matplotlib.pyplot as plt  
  
# Группировка данных по годам и жанрам, суммирование продаж  
yearly\_genre\_sales = df.groupby(['Year\_of\_Release', 'Genre'])['Global\_Sales'].sum().unstack()  
  
# Создание диаграммы площади Unstacked  
plt.figure(figsize=(12, 8))  
plt.stackplot(yearly\_genre\_sales.index, yearly\_genre\_sales.values.T, labels=yearly\_genre\_sales.columns)  
plt.title('Соотношение продаж игр по жанрам на протяжении лет')  
plt.xlabel('Год выпуска')  
plt.ylabel('Общие продажи (в млн. копий)')  
plt.legend(loc='upper left')  
plt.grid(True)  
plt.show()

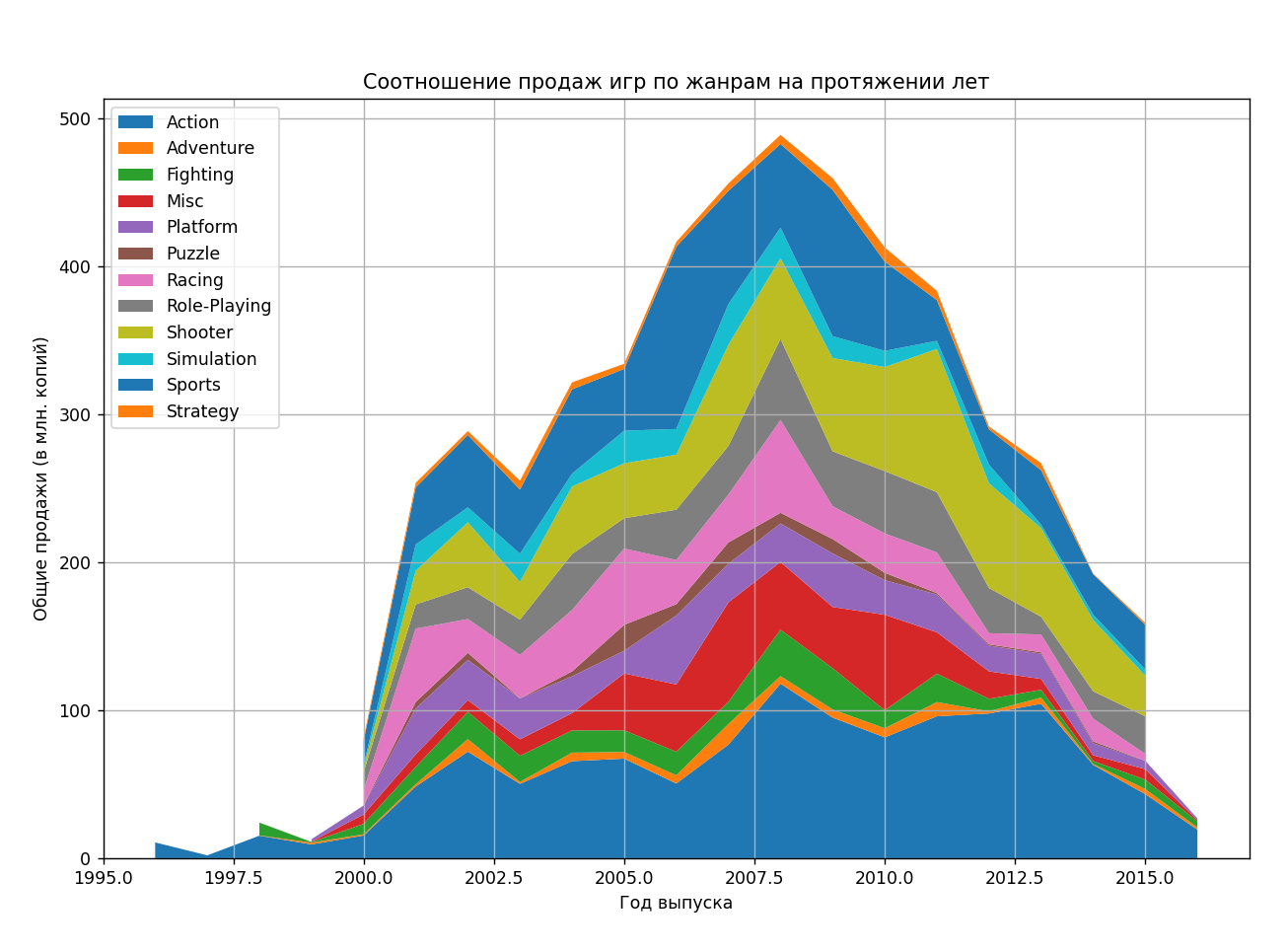


Рисунок 6 – Диаграмма площади Unstacked

График «скрипками» позволяет визуализировать распределение оценок критиков и пользователей по жанрам.

from main import df  
import matplotlib.pyplot as plt  
  
# Фильтрация данных для игр с известными оценками  
filtered\_df = df[(df['Critic\_Score'].notna()) & (df['User\_Score'].notna())]  
  
# Создание графика «скрипками»  
plt.figure(figsize=(12, 6))  
plt.violinplot([filtered\_df[filtered\_df['Genre'] == genre]['Critic\_Score'] for genre in filtered\_df['Genre'].unique()], showmedians=True)  
plt.title('График «скрипками» оценок критиков по жанрам')  
plt.xlabel('Жанр')  
plt.ylabel('Оценка')  
plt.xticks(range(1, len(filtered\_df['Genre'].unique()) + 1), filtered\_df['Genre'].unique(), rotation=45)  
plt.grid(True)  
plt.show()

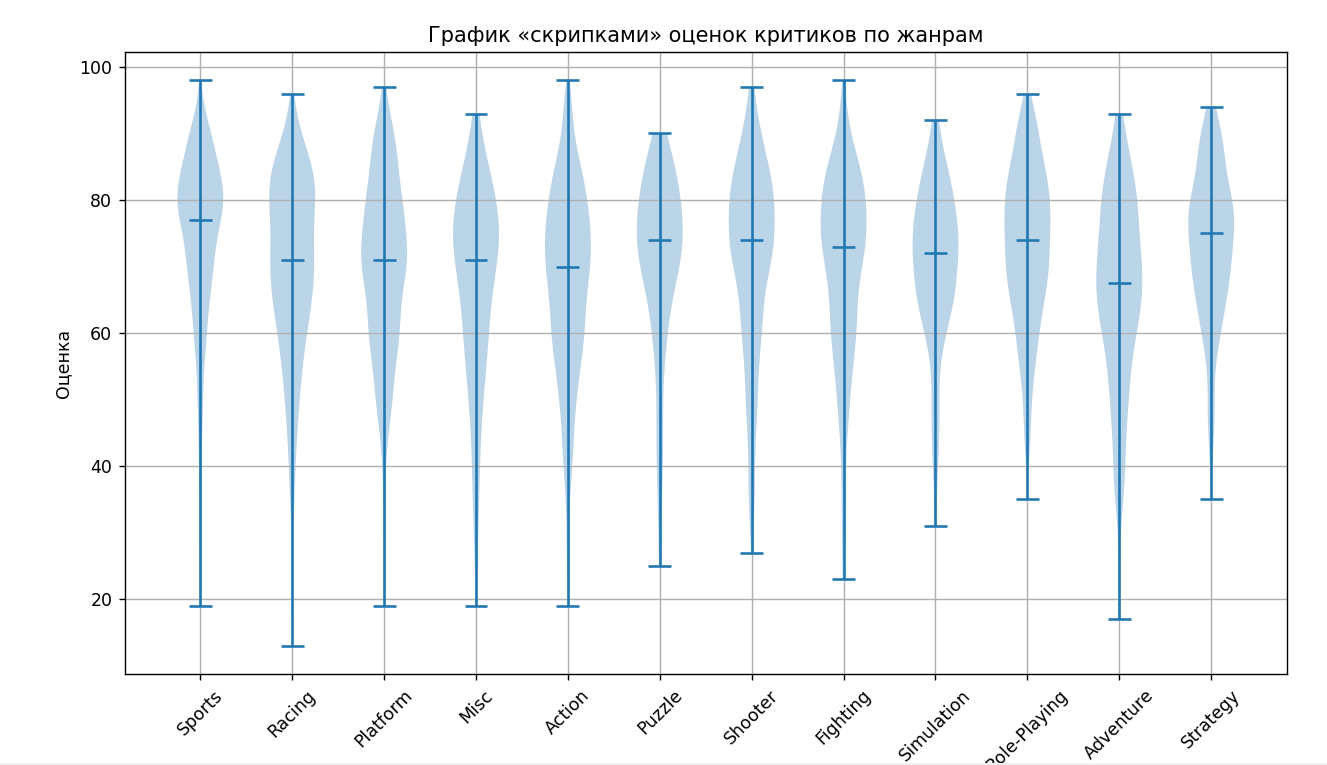


Рисунок 7 - График "Скрипками"

Создание визуализаций seaborn

Теперь давайте перейдем к библиотеке seaborn. Seaborn — это по сути более высокоуровневое API на базе библиотеки matplotlib. Seaborn содержит более адекватные дефолтные настройки оформления графиков. Если просто добавить в код import seaborn, то картинки станут гораздо симпатичнее. Также в библиотеке есть достаточно сложные типы визуализации, которые в matplotlib потребовали бы большого количество кода.

Познакомимся с первым таким "сложным" типом графиков pair plot (scatter plot matrix). Эта визуализация поможет нам посмотреть на одной картинке, как связаны между собой различные признаки.

sns.pairplot(df[['Global\_Sales', 'Critic\_Score', 'Critic\_Count',  
 'User\_Score', 'User\_Count']])  
plt.show()

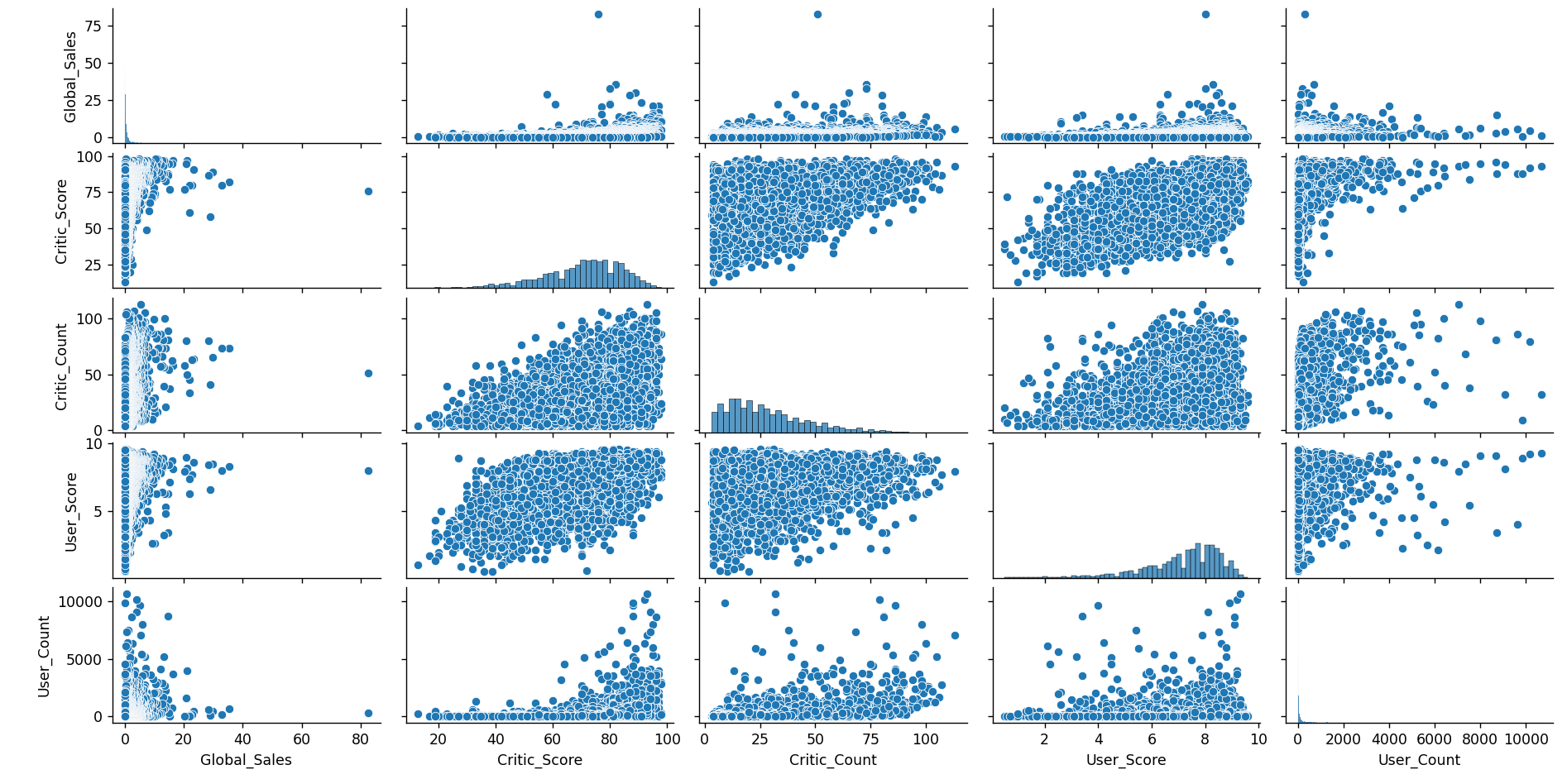


Рисунок 8 - Связь различных признаков

Также с помощью seaborn можно построить распределение, для примера посмотрим на распределение оценок критиков Critic\_Score. Для этого построим displot. По умолчанию на графике отображается гистограмма.

sns.distplot(df.Critic\_Score)

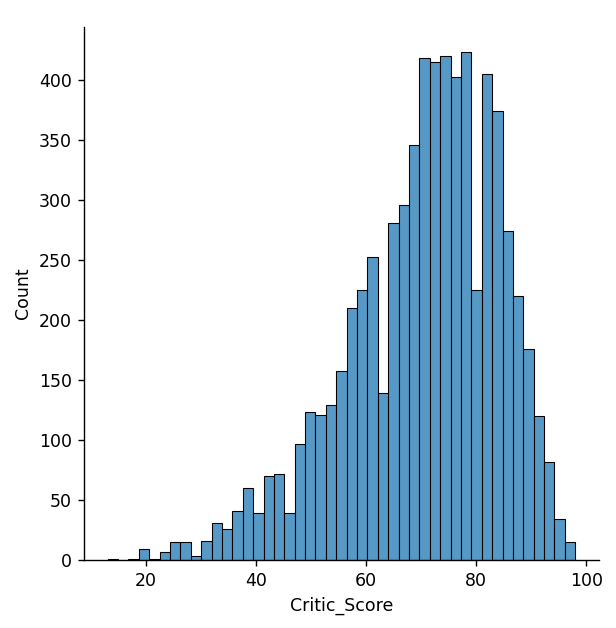


Рисунок 9 - Распределение оценок Critic\_Score

Для того чтобы подробнее посмотреть на взаимосвязь двух численных признаков, есть еще и  joint\_plot – это гибрид scatter plot и histogram (отображаются также гистограммы распределений признаков). Посмотрим на то, как связаны между собой оценка критиков Critic\_Score и оценка пользователя User\_Score.

sns.jointplot(x='Critic\_Score', y='User\_Score',

data=df, kind='scatter')

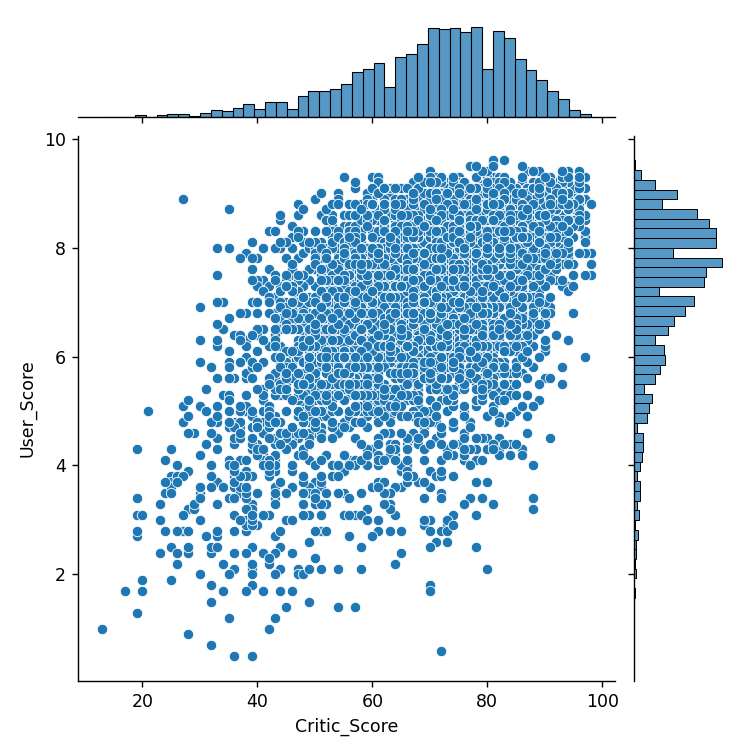


Рисунок 10 - Распределение Critic\_Score и User\_Score

Еще один полезный тип графиков – это box plot. Давайте сравним пользовательские оценки игр для топ-5 крупнейших игровых платформ.

from pair\_disp\_joint import df, sns, plt  
  
# Ящик  
top\_platforms = df.Platform.value\_counts().sort\_values(ascending = False).head(5).index.values  
sns.boxplot(y="Platform", x="Critic\_Score",  
 data=df[df.Platform.isin(top\_platforms)], orient="h")  
  
plt.show()

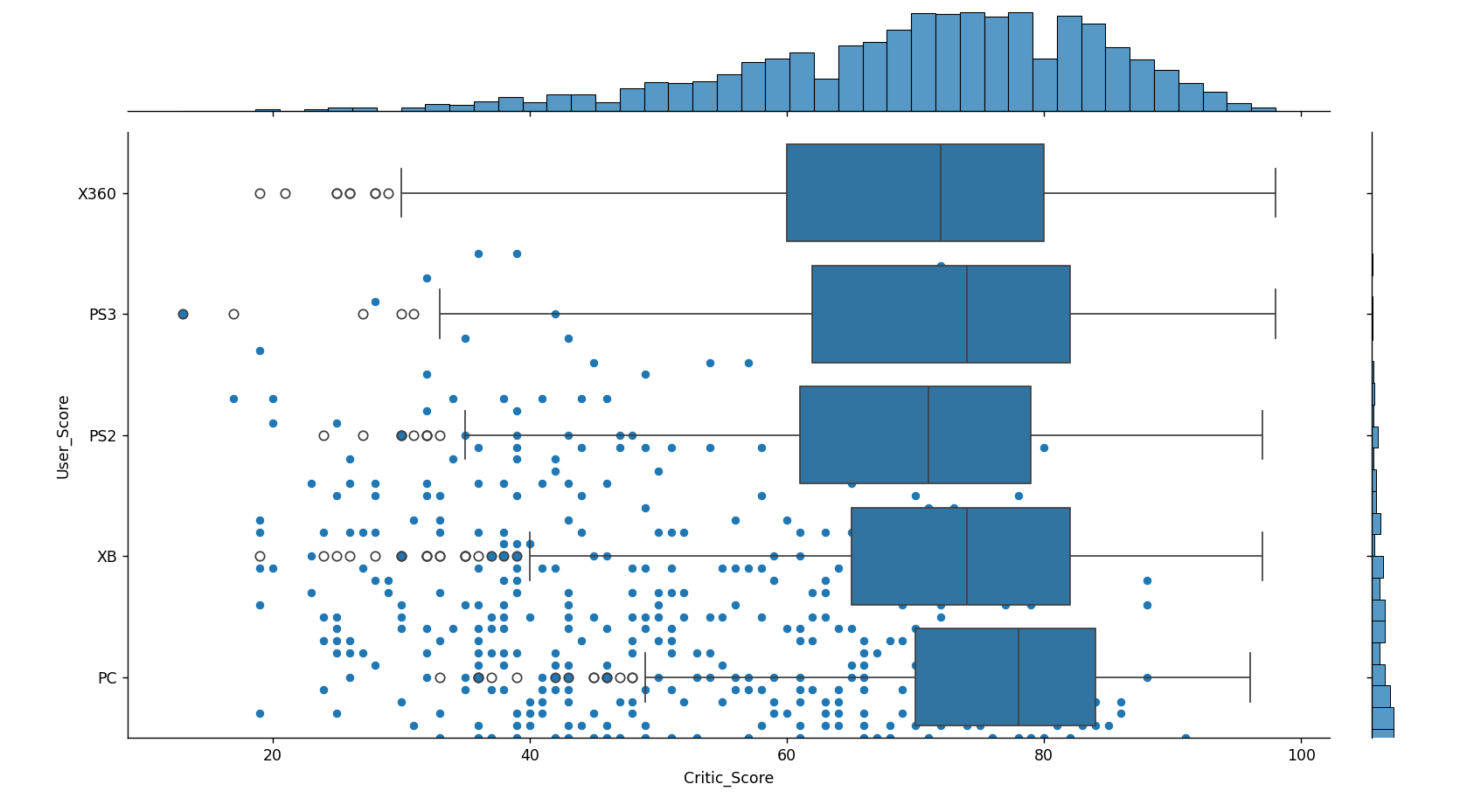


Рисунок 11 - Сравнение пользовательских оценок по игровым платформам

Думаю, стоит обсудить немного подробнее, как же понимать box plot. Box plot состоит из коробки (поэтому он и называется box plot), усиков и точек. Коробка показывает интерквантильный размах распределения, то есть соответственно 25% (Q1) и 75% (Q3) процентили.

Черта внутри коробки обозначает медиану распределения. С коробкой разобрались, перейдем к усам. Усы отображают весь разброс точек кроме выбросов, то есть минимальные и максимальные значения, которые попадают в промежуток (Q1 - 1.5\*IQR, Q3 + 1.5\*IQR), где IQR = Q3 - Q1 - интерквантильный размах. Точками на графике обозначаются выбросы (outliers) - те значения, которые не вписываются в промежуток значений, заданный усами графика.

И еще один тип графиков (последний из тех, которые мы рассмотрим в этой статье) - это heat map. Heat map позволяет посмотреть на распределение какого-то численного признака по двум категориальным. Визуализируем суммарные продажи игр по жанрам и игровым платформам.

from pair\_disp\_joint import df, sns, plt  
  
# Тепловая карта  
platform\_genre\_sales = df.pivot\_table(  
 index='Platform',  
 columns='Genre',  
 values='Global\_Sales',  
 aggfunc="sum").fillna(0).applymap(float)  
sns.heatmap(platform\_genre\_sales, annot=True, fmt=".1f", linewidths=.5)  
  
plt.show()

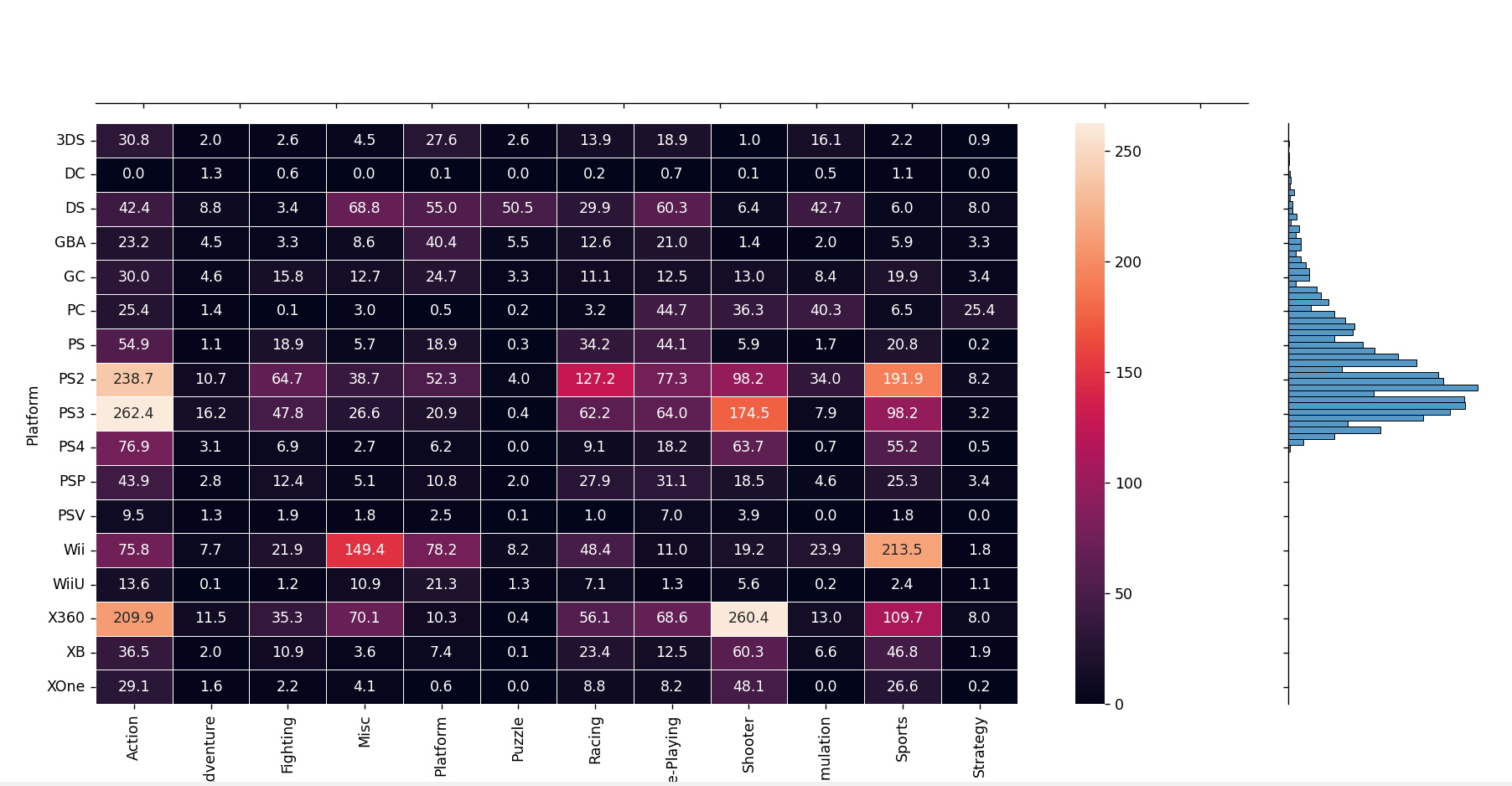


Рисунок 12 - Распределение численного признака по двум категориальным

Этот график поможет нам визуально анализировать распределение продаж видеоигр по жанрам и платформам, выявляя потенциальные закономерности и аномалии в данных.

from pair\_disp\_joint import df, sns, plt  
  
# Создаем catplot для отображения распределения продаж по жанрам и платформам  
g = sns.catplot(data=df, x='Platform', y='Global\_Sales', col='Genre', kind='box',  
 col\_wrap=3, height=4, aspect=1.5, sharey=False, palette='viridis')  
g.set\_axis\_labels('Платформа', 'Глобальные продажи (млн)')  
g.set\_titles(col\_template="{col\_name}")  
g.fig.suptitle('Распределение глобальных продаж по жанрам и платформам', y=1.02)  
  
# Поворачиваем метки оси x для лучшей читаемости  
for ax in g.axes.flat:  
 ax.tick\_params(axis='x', rotation=45)  
  
plt.tight\_layout()  
plt.show()

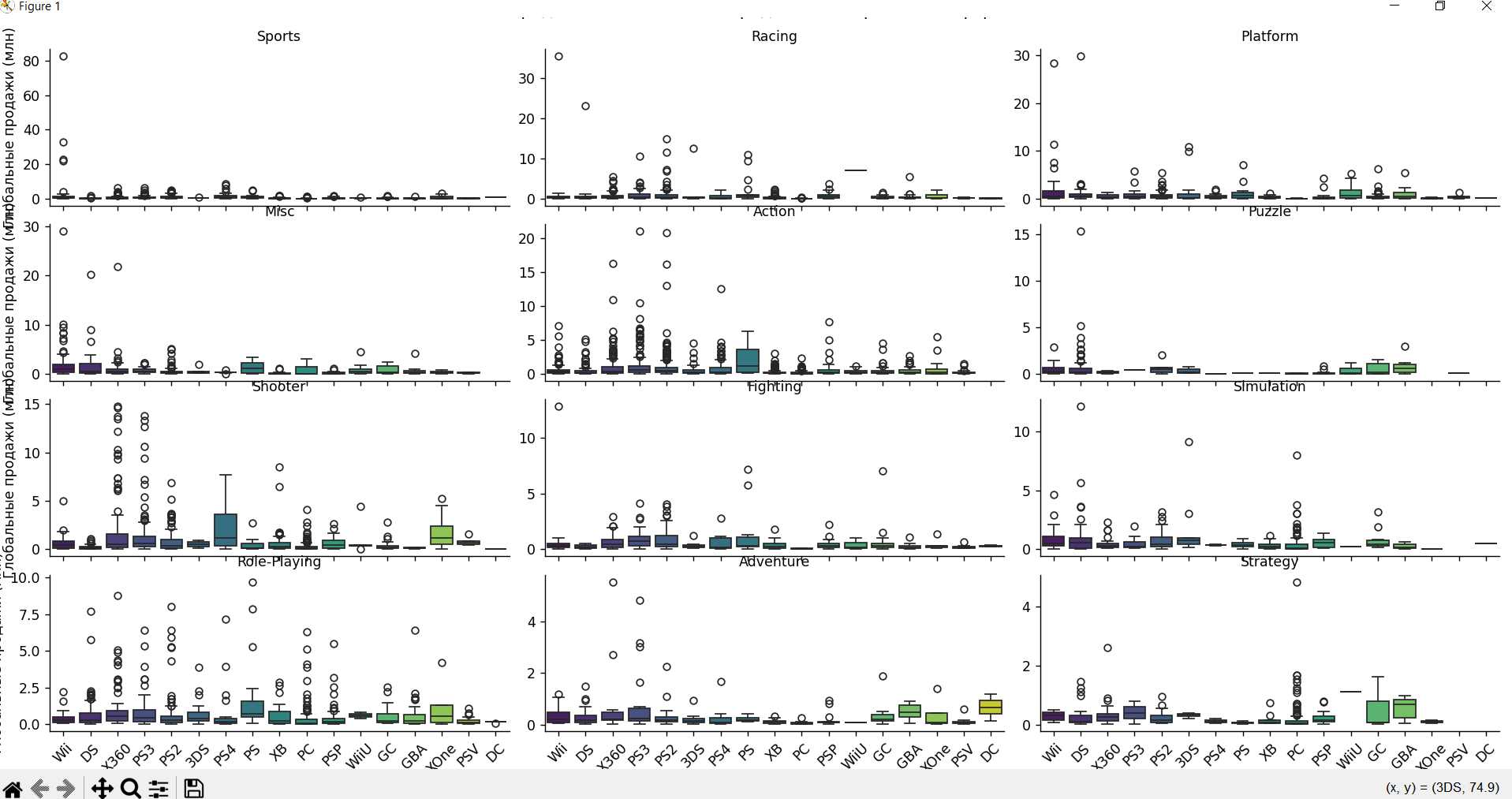


Рисунок 13 - График catplot

Создание визуализаций plotly

Мы рассмотрели визуализации на базе библиотеки matplotlib. Однако, это не единственная опция для построения графиков на языке python. Познакомимся также с библиотекой plotly. Plotly - это open-source библиотека, которая позволяет строить интерактивные графики без необходимости зарываться в javascript код.

Прелесть интерактивных графиков заключается в том, что можно посмотреть точное численное значение при наведении мыши, скрыть неинтересные ряды в визуализации, приблизить определенный участок графика и т.д.

Перед началом работы импортируем все необходимые модули в файл plotl.py и так же тут будем производить определенные действия с данными.

import plotly.express as px  
from main import df

Для начала построим line plot с динамикой числа вышедших игр и их продаж по годам.

# Группировка данных по годам и суммирование продаж, подсчет количества игр  
years\_df = df.groupby('Year\_of\_Release')[['Global\_Sales']].sum().join(  
 df.groupby('Year\_of\_Release')[['Name']].count())  
years\_df.columns = ['Global\_Sales', 'Number\_of\_Games']

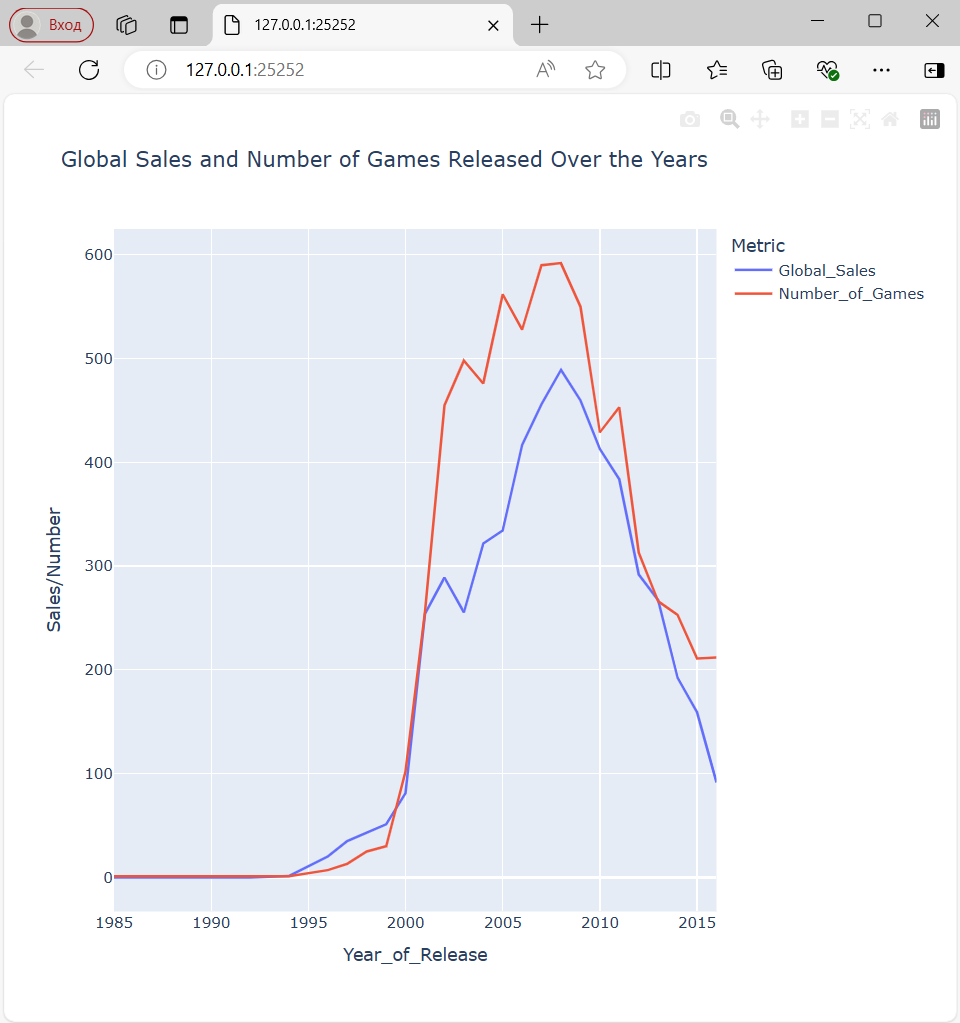


Рисунок 14 - Интерактивный график Plotly

from plotl import years\_df, px  
  
# Создание line plot с использованием Plotly  
fig = px.line(years\_df.reset\_index(), x='Year\_of\_Release', y=['Global\_Sales', 'Number\_of\_Games'],  
 title='Global Sales and Number of Games Released Over the Years',  
 labels={'value': 'Sales/Number', 'variable': 'Metric'})  
  
# Отображение графика  
fig.show()

Посмотрим также на рыночную долю игровых платформ, расчитанную по количеству выпущенных игр и по суммарной выручке. Для этого построим bar chart и изменим его внешний вид.

# Группировка данных по платформам и суммирование продаж, подсчет количества игр  
platforms\_df = df.groupby('Platform')[['Global\_Sales']].sum().join(  
 df.groupby('Platform')[['Name']].count()  
)  
platforms\_df.columns = ['Global\_Sales', 'Number\_of\_Games']  
  
# Сортировка данных по суммарным продажам в порядке убывания  
platforms\_df.sort\_values('Global\_Sales', ascending=False, inplace=True)

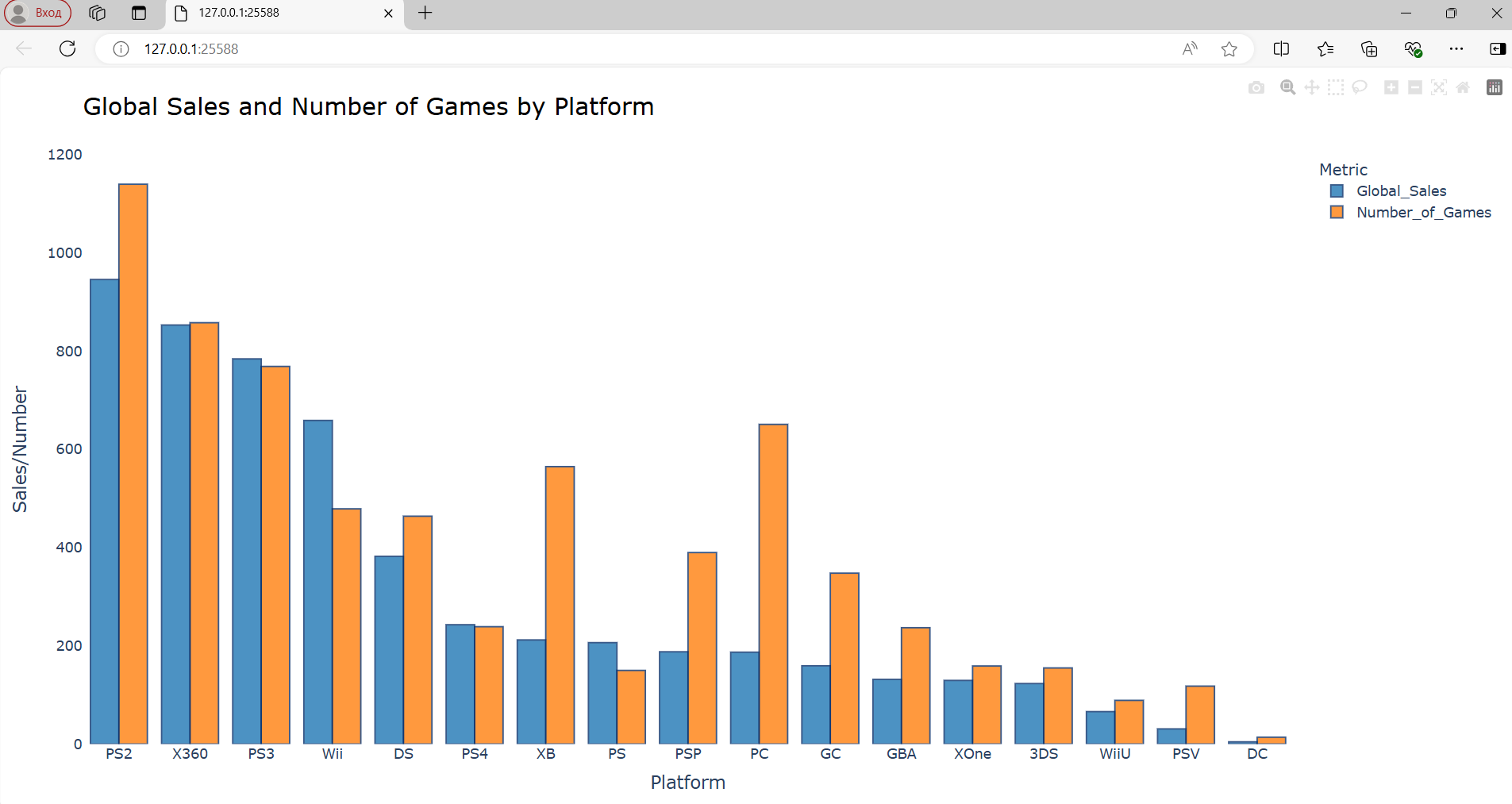


Рисунок 15 - bar chart на plotly

from plotl import platforms\_df, px  
  
# Создание bar chart с использованием Plotly  
fig = px.bar(platforms\_df.reset\_index(), x='Platform', y=['Global\_Sales', 'Number\_of\_Games'],  
 title='Global Sales and Number of Games by Platform',  
 labels={'value': 'Sales/Number', 'variable': 'Metric'},  
 barmode='group',  
 color\_discrete\_sequence=["#1f77b4", "#ff7f0e"]) # Установка цветов столбцов  
  
# Настройка внешнего вида графика  
fig.update\_layout(  
 title\_font\_size=24, # Размер шрифта заголовка  
 title\_font\_color="black", # Цвет шрифта заголовка  
 xaxis\_title\_font\_size=18, # Размер шрифта заголовка оси X  
 yaxis\_title\_font\_size=18, # Размер шрифта заголовка оси Y  
 xaxis\_tickfont\_size=14, # Размер шрифта меток оси X  
 yaxis\_tickfont\_size=14, # Размер шрифта меток оси Y  
 legend\_title\_font\_size=16, # Размер шрифта заголовка легенды  
 legend\_font\_size=14, # Размер шрифта текста легенды  
 plot\_bgcolor='white', # Цвет фона графика  
 paper\_bgcolor='white', # Цвет фона области вокруг графика  
 margin=dict(l=50, r=50, t=80, b=50) # Отступы вокруг графика  
)  
  
# Настройка столбцов  
fig.update\_traces(  
 marker\_line\_color='rgb(8,48,107)', # Цвет границ столбцов  
 marker\_line\_width=1.5, # Толщина границ столбцов  
 opacity=0.8 # Прозрачность столбцов  
)  
  
# Отображение графика  
fig.show()

В plotly можно построить и box plot. Рассмотрим различия оценок критиков в зависимости от жанра игры.

# Создаем список для хранения данных для каждого жанра  
data = []  
  
# Проходим по каждому уникальному жанру  
for genre in df.Genre.unique():  
 # Добавляем box plot для каждого жанра  
 data.append(  
 go.Box(y=df[df.Genre == genre].Critic\_Score, name=genre)  
 )

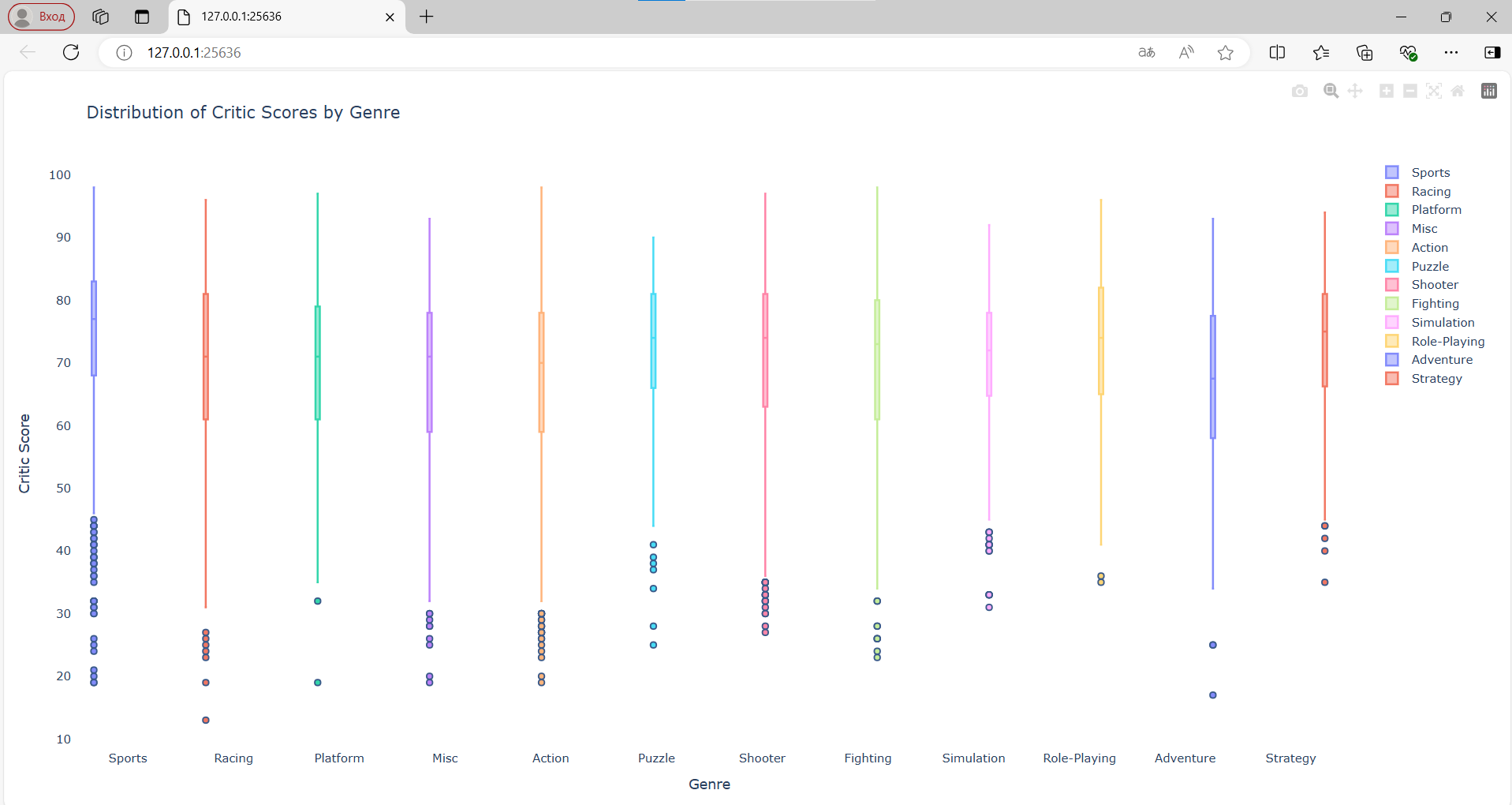


Рисунок 16 - Распределение оценок критиков в зависимости от жанра

from plotl import go, data  
  
# Создаем фигуру с box plot  
fig = go.Figure(data=data)  
  
# Настройка внешнего вида графика  
fig.update\_layout(  
 title='Distribution of Critic Scores by Genre',  
 yaxis=dict(title='Critic Score'),  
 xaxis=dict(title='Genre'),  
 boxmode='group', # Группировка box plot по жанрам  
 plot\_bgcolor='white', # Цвет фона графика  
 paper\_bgcolor='white', # Цвет фона области вокруг графика  
 margin=dict(l=50, r=50, t=80, b=50) # Отступы вокруг графика  
)  
  
# Настройка box plot  
fig.update\_traces(  
 marker\_line\_color='rgb(8,48,107)', # Цвет границ box plot  
 marker\_line\_width=1.5, # Толщина границ box plot  
 opacity=0.8 # Прозрачность box plot  
)  
  
# Отображение графика  
fig.show()

Sunburst Chart - иерархическое представление данных по жанрам и платформам. Это поможет увидеть, как продажи распределены по жанрам и платформам.

import plotly.express as px  
from main import df, useful\_cols  
  
df\_useful = df[useful\_cols]  
  
fig4 = px.sunburst(df\_useful, path=['Genre', 'Platform'], values='Global\_Sales', title='Иерархия жанров и платформ по продажам')  
fig4.show()

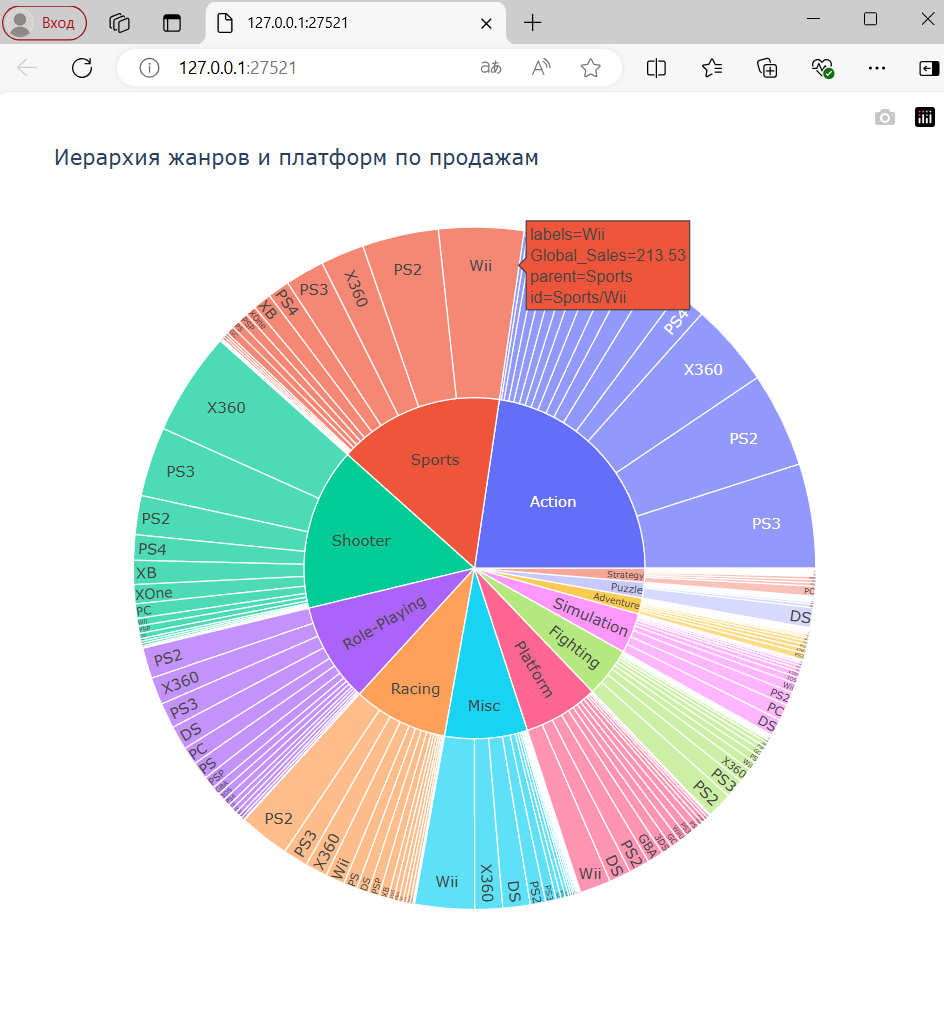
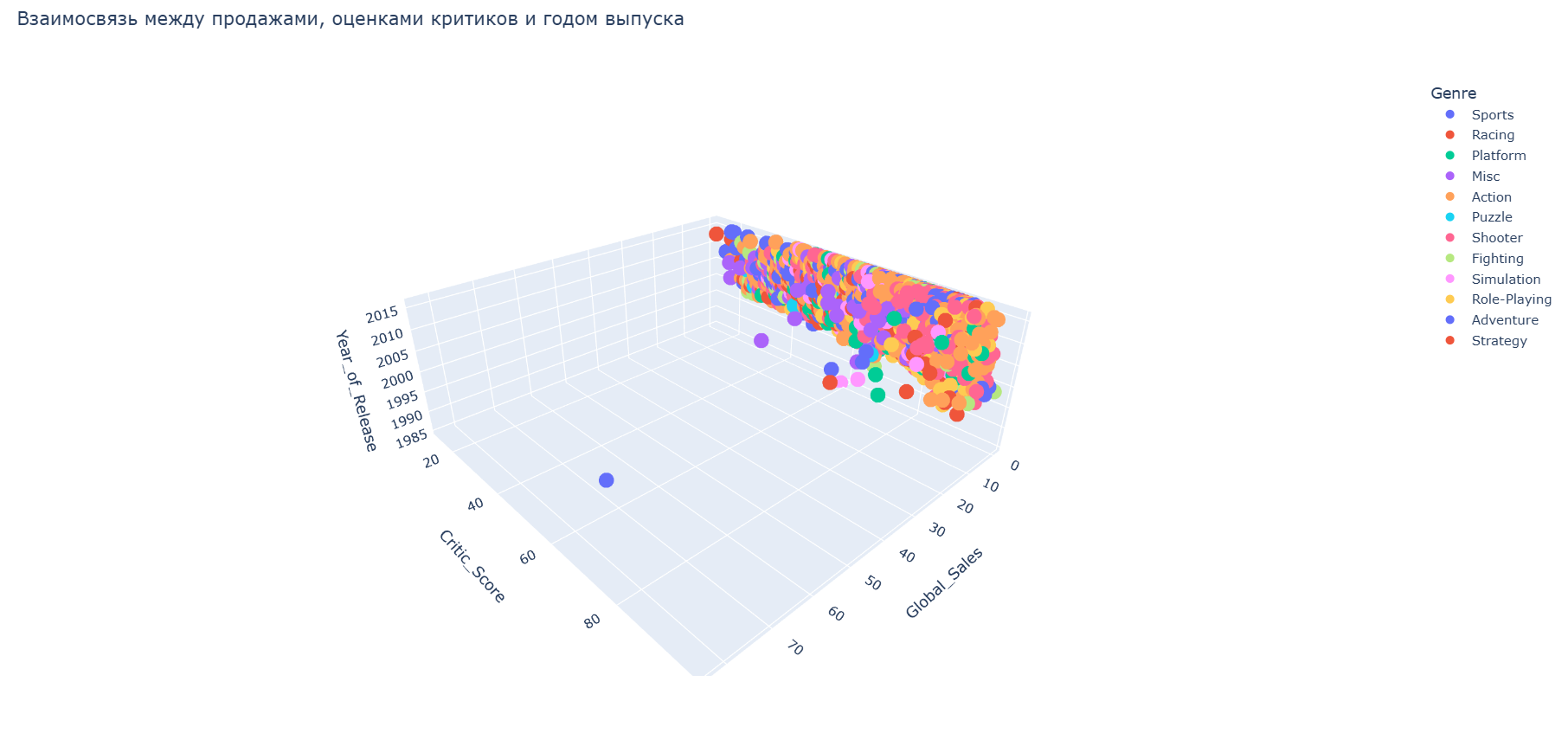


Рисунок 17 - Иерархия жанров

Так же с помощью данной библиотеки можно отобразить 3D график. Давайте посмотрим на примере обычной диаграммы рассеяния



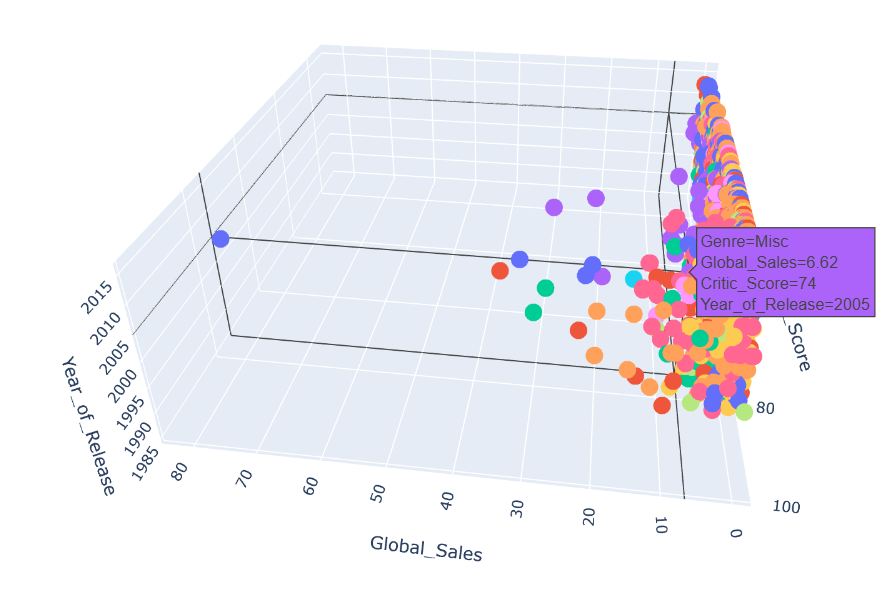


Рисунок 18 - Диаграмма рассеяния

ЗАКЛЮЧЕНИЕ

В ходе данного проекта было проведено сравнение трех популярных библиотек для визуализации данных в Python: Matplotlib, Seaborn и Plotly. Исследование было основано на анализе датасета, содержащего данные о продажах и оценках видеоигр, что позволило продемонстрировать возможности каждой библиотеки в реальных условиях. Проведя данное исследование можно сделать некоторые выводы и обозначить существенную разницу по критериям: Функциональность, удобство пользования, производительность.

Функциональность

### *Создание различных типов графиков*

Matplotlib является одной из самых старых и широко используемых библиотек для визуализации данных в Python. Она предоставляет широкий спектр типов графиков, включая линейные графики, гистограммы, диаграммы рассеяния, графики распределения, 3D графики и многое другое. Matplotlib также позволяет создавать сложные композиции графиков, комбинируя различные типы визуализаций на одном холсте.

Seaborn — это библиотека, построенная на основе Matplotlib, которая предоставляет более высокоуровневый интерфейс для создания статистических графиков. Seaborn упрощает создание сложных визуализаций, таких как тепловые карты, диаграммы рассеяния с регрессионными линиями, графики распределения, иерархические кластеризации и многое другое. Seaborn также предоставляет удобные функции для работы с данными, такими как pairplot, jointplot и heatplot

Plotly — это библиотека для создания интерактивных графиков и визуализаций. Она поддерживает широкий спектр типов графиков, включая линейные графики, гистограммы, диаграммы рассеяния, тепловые карты, 3D графики, картографические визуализации и многое другое. Plotly также позволяет создавать сложные визуализации с использованием Dash, фреймворка для создания веб-приложений с интерактивными графиками.

### *Настройка внешнего вида графиков*

Matplotlib предоставляет обширные возможности для настройки внешнего вида графиков. Пользователи могут изменять цвета, стили линий, метки осей, легенды, фон и многие другие аспекты графика. Однако, из-за своей низкоуровневой природы, настройка может быть сложной и требовать глубокого понимания API.

Seaborn упрощает настройку внешнего вида графиков, предоставляя набор тем и стилей по умолчанию, которые можно легко изменить. Seaborn также предоставляет функции для настройки цветовых палитр, меток осей и других элементов графика. Однако, более глубокая настройка может потребовать использования функций Matplotlib.

Plotly предоставляет мощные инструменты для настройки внешнего вида графиков. Пользователи могут изменять цвета, стили, метки осей, легенды, анимацию и другие аспекты графика. Plotly также поддерживает CSS-стили для настройки внешнего вида веб-интерфейса. Благодаря интерактивности, пользователи могут легко изменять параметры графика на лету.

### *Добавление интерактивности*

Matplotlib предоставляет базовые возможности для интерактивности, такие как масштабирование и панорамирование, но они ограничены и требуют дополнительных библиотек, таких как mpl\_interactions или ipywidgets.

Seaborn не предоставляет встроенных возможностей для интерактивности. Для добавления интерактивности к графикам Seaborn, пользователи должны использовать Matplotlib или другие библиотеки.

Plotly — это лидер в области интерактивной визуализации. Графики, созданные с использованием Plotly, могут быть интерактивными из коробки, с возможностью масштабирования, панорамирования, наведения курсора, выделения данных и многого другого. Plotly также поддерживает создание интерактивных веб-приложений с использованием Dash.

Удобство использования

### *Простота и интуитивность*

Matplotlib имеет относительно низкий порог вхождения, но его API может быть сложным для новичков. Для создания сложных графиков требуется глубокое понимание функций и методов библиотеки.

Seaborn предлагает более высокоуровневый интерфейс по сравнению с Matplotlib, что делает его более интуитивным и простым в использовании. Многие функции Seaborn автоматически настраивают внешний вид графиков, что упрощает создание сложных визуализаций.

Plotly предоставляет интуитивный интерфейс для создания интерактивных графиков. Благодаря поддержке синтаксиса, похожего на Seaborn, Plotly легко освоить. Однако, для создания сложных интерактивных приложений может потребоваться изучение дополнительных инструментов, таких как Dash.

### *Документация и примеры*

Matplotlib имеет обширную документацию, включающую множество примеров и руководств. Однако, из-за своей сложности, документация может быть перегружена техническими деталями.

Seaborn также имеет хорошо структурированную документацию с множеством примеров и руководств. Документация Seaborn более доступна для новичков, чем у Matplotlib, благодаря более высокоуровневому интерфейсу.

Plotly предоставляет подробную документацию с множеством примеров и руководств. Документация Plotly хорошо структурирована и включает разделы для начинающих и опытных пользователей. Plotly также предлагает множество примеров на своем сайте и в репозитории GitHub.

### *Поддержка сообщества*

Matplotlib имеет одно из самых больших сообществ среди библиотек для визуализации данных в Python. Он активно поддерживается и развивается, с множеством ресурсов, включая форумы, GitHub и Stack Overflow.

Seaborn также имеет активное сообщество, но оно меньше, чем у Matplotlib. Тем не менее, Seaborn активно поддерживается и развивается, с множеством ресурсов, включая документацию, форумы и GitHub.

Plotly имеет активное и растущее сообщество. Благодаря интерактивности и возможностям создания веб-приложений, Plotly привлекает множество пользователей. Plotly также активно поддерживается и развивается, с множеством ресурсов, включая документацию, форумы и GitHub.

Производительность

### *Скорость работы*

Matplotlib является одной из самых быстрых библиотек для визуализации данных в Python. Однако, создание сложных графиков может занимать значительное время.

Seaborn, будучи построенной на основе Matplotlib, наследует его производительность. Однако, из-за более высокоуровневого интерфейса, Seaborn может быть немного медленнее, чем Matplotlib, при создании сложных визуализаций.

Plotly, благодаря использованию JavaScript для рендеринга графиков, может быть медленнее, чем Matplotlib и Seaborn, особенно при работе с большими наборами данных. Однако, Plotly оптимизирован для создания интерактивных графиков, что компенсирует его относительно низкую скорость.

### *Потребление ресурсов*

Matplotlib потребляет относительно мало ресурсов, что делает его подходящим для работы на слабых устройствах или в средах с ограниченными ресурсами.

Seaborn, будучи надстройкой над Matplotlib, потребляет немного больше ресурсов, чем Matplotlib. Однако, это незначительно влияет на общую производительность.

Plotly, из-за использования JavaScript и создания интерактивных графиков, потребляет больше ресурсов, чем Matplotlib и Seaborn. Однако, это компенсируется возможностями создания сложных интерактивных визуализаций.

СПИСОК ЛИТЕРАТУРЫ

1. Марк Лутц. Изучаем Python / Марк Лутц.. – 5-е изд Изучаем Python, том1,5-еизд : Пер.сангл.—СПб. : ООО “Диалектика”, 2019.—832с. : ил.—Парад, тит.англ. – Режим доступа: – URL:https://studylib.ru/doc/6266928/izuchaem-python.-5-e-izd.-tom-1.-mark-lutc (дата обращения: 16.11.2024). – ISBN 978-5-907144-52-1. – Текст : электронный.
2. Мюллер, Джон Пол. Python для чайников, 2-е изд. : Пер. с англ. -СПб. : ООО "Диалектика", 2019. -416 с.: ил. -Парал. тит. англ. – Режим доступа: – URL:https://studylib.ru/doc/6296242/myuller-dzh-p-python-dlya-chajnikov-dlya-chajnikov-2019 (дата обращения: 16.11.2024). – ISBN 978-5-907144-26-2. – Текст : электронный.
3. Уэс Маккинни. Python и анализ данных: Первичная обработка данных с применением pandas, NumPy и Jupiter / пер. с англ. А. А. Слинкина. 3-е изд. – М.: МК Пресс, 2023. – 536 с.: ил. – Режим доступа: – URL:https://studylib.ru/doc/6399984/python-i-analiz-dannyh-tret.\_e-izdanie-2023-makkinni-ue-s (дата обращения: 16.11.2024). – ISBN 978-5-93700-174-0. – Текст : электронный.
4. Эрик Мэтиз. Изучаем Python / Эрик Мэтиз. – 3-е изд. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017. — 496 с.: ил. — (Серия «Библиотека программиста»). – Режим доступа: – URL: https://howdyho.net/static/uploads/files/E\_Metiz\_-\_Izuchaem\_Python\_Programmirovanie\_igr\_vizualizatsia\_dannykh\_veb-prilozhenia\_-\_2017.pdf (дата обращения 16.11.2024) –ISBN 978-5-496-02305-4. – Текст : электронный.
5. Плас Дж. Вандер. Python для сложных задач: наука о данных и машинное обучение. — СПб.: Пи-тер, 2018. — 576 с.: ил. — (Серия «Бестселлеры O’Reilly»). – Режим доступа: – URL: https://studylib.ru/doc/6344331/python-dlya-slozhnyh-zadach.-nauka-o-dannyh-i-mashinnoe-obuche
6. Абдрахманов М.И. Библиотека Matplotlib - devpractice.ru. 2019. - 100 с.: ил. – Режим доступа: – URL: <https://studylib.ru/doc/6386039/biblioteka-matplotlib-abdrahmanov-m.i>.
7. Статья: "Визуализация данных в Python с использованием Matplotlib и Seaborn". (2021). Хабр. [Электронный ресурс]. URL: https://habr.com/ru/articles/547158/ (дата обращения: 17.11.2024).
8. Статья: "Введение в Plotly: создание интерактивных графиков в Python". (2022). Хабр. [Электронный ресурс]. URL: https://habr.com/ru/articles/656085/ (дата обращения: 17.11.2024).

ПРИЛОЖЕНИЕ

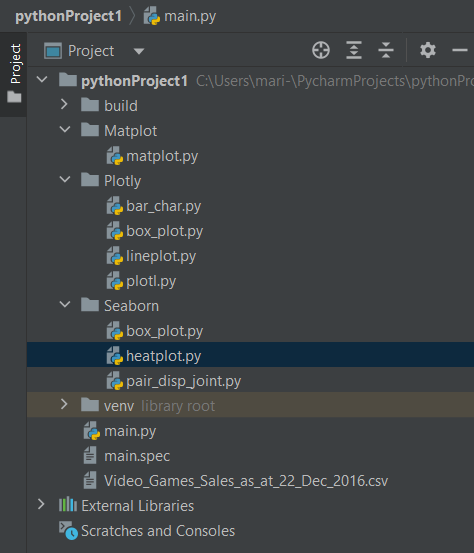


Рисунок 19 - Структура проекта