ДИПЛОМНАЯ РАБОТА

Сравнение различных библиотек для визуализации данных:
Matplotlib, Seaborn и Plotly

Автор:

Ганцева Анна Вячеславовна

2024 год

СОДЕРЖАНИЕ

1. **Введение**

Визуализация данных - это графическое представление данных. Она преобразует огромный набор данных в небольшие графики, тем самым помогая в анализе данных и прогнозах. Это незаменимый элемент науки о данных, который делает сложные данные более понятными и доступными.

Matplotlib, Seaborn и ***Plotly*** выступают в качестве основы визуализации данных с помощью Python.

2 **Обоснование выбора темы:**

Я считаю, что данная тема имеет не только практическое значение для многих людей, работающих со сбором аналитических данных (их группировкой, анализом, сортировкой, выборкой и т.п.), но и для людей, принимающих решения на основе полученных данных. Когда ты сам работаешь в сфере анализа, ты умеешь ценить время и своё, и своих коллег. Ты понимаешь, насколько важно в минимально отведенное время преподнести максимальный объем информации для принятия быть может кардинально отличающихся решений.

Кроме этого на мой выбор повлияли следующие факторы:

1. *Актуальность темы:*
   - В современном мире растет объем данных и возникает необходимость их визуализации для анализа и принятия решений.
   - Выбор сравнения популярных библиотек для визуализации данных актуален и востребован.
2. *Применимость к практике:*
   - Работа над такой темой поможет лучше понять возможности и ограничения популярных инструментов для работы с данными.
   - Знание сравнения библиотек будет полезно при выборе оптимального инструмента для конкретной задачи в будущем.
3. *Возможностью углубленного исследования:*
   - Каждая из библиотек имеет свои особенности и преимущества, что позволяет провести глубокий анализ.
   - Сравнение позволяет выявить сильные и слабые стороны каждого инструмента.
4. *Потенциальное применение результатов:*
   - Результаты исследования могут быть использованы при выборе инструментов для проектов в будущем.
   - Информация может быть полезна для других исследователей и разработчиков.

Всё это позволяет не только получить теоретические знания, но и приобрести практические навыки работы с популярными инструментами для визуализации данных.

3 Истории созданий не будут лишними…

Для начала немного истории. Она поспособствует пониманию предназначения этих библиотек, цели создания, их развития и текущего состояния, что является немаловажным фактором при проведении сравнения.

Matplotlib

Была создана в 2003 году. Автором и основателем проекта стал нейробиолог Джон Д. Хантер (John D. Hunter). Началом ее создания была попытка создать библиотеку, подобную MATLAB но только для Python. Целью Джона было сделать Matplotlib простыми, а сложные — возможными.

Но постепенно Matplotlib превратилась в полноценную библиотеку для визуализации данных.

Постепенно добавлялись новые возможности для создания графиков и диаграмм, улучшались интерфейсы и документация, производительность для больших объемов данных.

С 2012 года данная библиотека продолжает развиваться и поддерживаться обществом. И на текущий момент она остается основной библиотекой для создания статистических графиков.

Seaborn

Seaborn была разработана в 2010 году Майклом Бетенкуртом (Michael W. T. Betancourt) и основана на Matplotlib.

Она была разработана как расширение и улучшение возможности Matplotlib для создания визуализаций данных, для упрощения процесса создания красивых статистических графиков на основе мощностей Matplotlib.

Сейчас Seaborn действует как библиотека-оболочка для Matplotlib и является популярной библиотекой для визуализации данных в сообществе Python. Она часто используется вместе с другими библиотеками из экосистемы NumPy и SciPy.

По умолчанию он генерирует красивые графики и предлагает множество функций для визуализации статистических данных. Это помогает исследовать взаимосвязи в наборах данных и обнаруживать закономерности между различными переменными.

Plotly

В 2011 год: Джек Честер (Jack Chester), Крис Уилкокс (Chris Wilcox), Мэттью Шарп (Matthew Sharp) и Алекс Дэвис (Alex Davis) основали компанию Plotly.

И в этом же году начинается разработка открытой библиотеки для визуализации данных на Python с одноименным названием – **Plotly.py.**

Над проектом трудится поэт и программист Фил Эйверс. Plotly начался как проект для создания интерактивных графиков в Python.

Разработка первой версии продолжается 3 года (по 2013 год включительно) и уже в 2014 году Plotly.py становится одной из самых скачиваемых библиотек для визуализации данных.

Но компания не останавливается на достигнутом и в 2016 году выходит первое крупное обновление Plotly Dash 1.0.0.

А в 2017 году Plotly был открыт как бесплатная библиотека с открытым исходным кодом.

После этого компания начинает работать над расширением функциональности и выпускает различные версии и модификации **Plotly.py** рассчитанные на разные слои пользователей и их кошельки. Так появляются:

- Plotly.express – для упрощенного способа создания графиков
- Dash Enterprise - платной версии Dash с дополнительными функциями
- Plotly.js 1.55 с улучшенной поддержкой

И по настоящий момент Plotly постоянно развивается и расширяет свои возможности, оставаясь одной из ведущих библиотек для визуализации данных в Python, которая известна своей возможностью создания интерактивных графиков.

4 Определение цели и задач исследования:

Основной целью исследования сравнения трех популярных библиотек Matplotlib, Seaborn и Plotly является помощь пользователям сделать выбор необходимой библиотеки в той или иной ситуации.

Для достижения поставленной цели:

- ✓ мы выделим основные характеристики всех трех библиотек (ключевые особенности, преимущества и недостатки);
- ✓ сделаем анализ функциональности и возможностей каждой из трех библиотек на примере нескольких самых распространенных типов графиков;
- ✓ исследуем возможности интерактивности и встраиваемости в веб-приложения.
- ✓ дадим рекомендации по выбору библиотеки в зависимости от задачи.

5 Основные характеристики библиотек (их ключевые особенности)

Перечислять все особенности не имеет смысла, поскольку для понимания отличительных особенностей каждой из сравниваемых библиотек, достаточно выделить основные.

5.1 Ключевые особенности *Matplotlib*:

1. Гибкость и универсальность

Matplotlib обладает высокой степенью гибкости и универсальности:

- Предоставляет широкий спектр инструментов для создания различных типов графиков и диаграмм.
- Позволяет легко настраивать и кастомизировать визуализацию.
- Интегрируется с другими популярными библиотеками Python, такими как NumPy и Pandas.

2. Простота использования

Matplotlib известен своей простотой применения:

- Предлагает два основных способа использования - быстрый (pyplot) и объектно-ориентированный.
- Быстрый способ подходит для простых графиков, а объектно-ориентированный - для более сложных визуализаций.
- Имеет MATLAB-подобный интерфейс, что облегчает использование для тех, кто знаком с MATLAB.

3. Качество публикаций и широкое применение

Matplotlib широко используется в научных и бизнес-кругах благодаря своим преимуществам:

- Позволяет создавать качественные графики для научных публикаций и презентаций.
- Широко применяется в различных областях, включая научные исследования, бизнес-аналитику и образование.
- Имеет богатую документацию и активное сообщество разработчиков.

5.2 *Ключевые особенности Seaborn*

1. Улучшенный интерфейс для статистических графиков

Seaborn предоставляет более удобный и эстетически привлекательный интерфейс для создания статистических графиков по сравнению с Matplotlib:

- Предоставляет готовые функции для создания различных типов графиков, таких как countplot, barplot, boxplot и другие.
- Интегрирует стили и цветовые схемы, делая графики более профессиональными.
- Позволяет легко настраивать параметры визуализации.

2. Гибкость в работе с категориальными переменными

Seaborn особенно удобен для работы с категориальными данными:

- Позволяет быстро создавать графики, показывающие зависимость целевой переменной от нескольких категориальных признаков.
- Поддерживает различные виды столбчатых диаграмм (grouped, stacked) для сравнения частот между категориями.

- Легко добавлять дополнительные признаки для создания вложенных графиков (faceted plots).

3. Интеграция с Pandas и Matplotlib

Seaborn тесно интегрируется с другими популярными библиотеками:

- Работает на основе данных из Pandas DataFrame.
- Использует Matplotlib для рендеринга графиков, что обеспечивает высокую производительность и широкие возможности настройки.
- Предоставляет удобный способ управления стилями и параметрами через rcParams.

Эти три особенности делают Seaborn мощным инструментом для создания качественных статистических визуализаций в Python, особенно при работе с категориальными данными.

5.3 *Ключевые особенности* Plotly

1. Интерактивность и высококачественные графики

Plotly известен своей способностью создавать интерактивные, качественные графики:

- Позволяет создавать различные типы графиков, включая линейные диаграммы, точечные графики, столбчатые диаграммы, круговые диаграммы и многое другое.
- Графики можно легко настроить и кастомизировать для достижения желаемого внешнего вида.
- Поддерживает создание сложных визуализаций, включая 3D графики и интерактивные элементы.

2. Удобство использования

Plotly предлагает удобный и интуитивно понятный интерфейс:

- Библиотека предоставляет высокоуровневый API (Plotly Express) для быстрого создания графиков с минимальным кодом.
- Поддерживает различные форматы входных данных, включая Pandas DataFrames, NumPy массивы и xarrays.
- Легко добавлять легенды, метки и другие элементы интерактивности к графикам.

3. Гибкость и расширяемость

Plotly предлагает широкие возможности для настройки и расширения функциональности:

- Позволяет создавать сложные графики с множеством элементов и взаимодействий.
- Поддерживает создание анимаций и интерактивных элементов, таких как кнопки и слайдеры.
- Имеет встроенную поддержку для создания веб-приложений с помощью Dash, что позволяет создавать полноценные аналитические приложения.

6 Сравним преимущества и недостатки

Matplotlib

Преимущества:

- Большая гибкость и универсальность
- Широкие возможности для настройки и кастомизации
- Поддержка широкого спектра типов графиков

Недостатки:

- Высокая сложность освоения из-за сложного синтаксиса
- Требует написания больше кода для достижения определенных типов визуализаций

Seaborn

Преимущества:

- Улучшенный интерфейс для статистических графиков
- Более эстетически привлекательные визуализации
- Легче использовать по сравнению с Matplotlib

Недостатки:

- Меньше гибкости по сравнению с Matplotlib
- Ограниченный набор типов графиков

Plotly

Преимущества:

- Интерактивность графиков
- Возможность создания сложных визуализаций
- Хорошая поддержка 3D графиков и географических данных

Недостатки:

- Низкая производительность при работе с большими данными
- Зависимость от веб-технологий может быть проблемой в ограниченных средах
- Увеличение размера файла для интерактивных графиков

7 Ключевые различия:

Я не пыталась найти и сопоставить все различия, я выделила 5 ключевых, знания которых (на мой взгляд) будет достаточно, чтобы также помочь вам с выбором необходимой библиотеки.

Я сразу разместила библиотеки в порядке убывания их критериев (от сильной к менее сильной):

1. Interactivity (Интерактивность): Plotly > Seaborn > Matplotlib

2. Ease of use (Простота использования): Plotly > Seaborn > Matplotlib
3. Aesthetics (Визуальное оформление: Plotly > Seaborn > Matplotlib
4. Performance (Производительность): Matplotlib > Plotly > Seaborn
5. Customization (Настройка): Matplotlib > Plotly > Seaborn

А теперь дам пояснения относительно данных мной оценок:

1.) **Оценка Интерактивности** между Plotly, Seaborn и Matplotlib основана на следующем:

Plotly

- Предоставляет наиболее высокий уровень интерактивности среди трех библиотек.
- Графики Plotly могут быть полностью интерактивными, позволяя пользователю взаимодействовать с ними непосредственно в браузере.
- Поддерживает элементы интерфейса, такие как кнопки, слайдеры и выпадающие списки, которые пользователь может использовать для изменения визуализации.
- Позволяет создавать интерактивные dashboards и веб-приложения для аналитики данных.

Seaborn

- Не имеет встроенной поддержки интерактивности.
- Создает статичные графики, которые не могут быть изменены пользователем после их создания.
- Интерактивность ограничена возможностью сохранения графиков в форматах, поддерживающих интерактивность (например, HTML).

Matplotlib

- Также не имеет встроенной поддержки интерактивности.
- Создает статичные графики, которые не могут быть изменены после их создания.
- Интерактивность может быть добавлена с помощью дополнительных библиотек или инструментов.

Таким образом, Plotly предлагает наибольшую степень интерактивности. Seaborn и Matplotlib ограничены созданием статичных графиков и не предназначены для интерактивности по умолчанию.

2.) **Оценка Простоты использования** между Plotly, Seaborn и Matplotlib основана на следующем:

Plotly

- Предлагает высокоуровневый API (Plotly Express) для быстрого создания графиков с минимальным кодом.

- Интуитивно понятный интерфейс, позволяющий создавать сложные визуализации с относительно простым кодом.
- Поддерживает различные форматы входных данных и легко настраивается для различных типов графиков.

Seaborn

- Улучшенный интерфейс по сравнению с Matplotlib, но менее гибкий, чем Plotly.
- Предоставляет готовые функции для создания различных типов статистических графиков.
- Более прост в использовании, чем Matplotlib, но менее мощный, чем Plotly.

Matplotlib

- Требует написания больше кода для достижения желаемых результатов.
- Имеет более низкий уровень абстракции, что может быть сложно для начинающих.
- Хотя и предоставляет широкие возможности настройки, это может усложнить процесс создания простых графиков.

Таким образом, Plotly предлагает наиболее интуитивный и удобный способ работы с данными, за счет своей системы Plotly Express и современного интерфейса. Seaborn находится посередине, предоставляя более удобный интерфейс, чем Matplotlib, но менее гибкий, чем Plotly. Matplotlib требует наибольших навыков программирования, но предоставляет максимальную гибкость в настройке визуализаций.

3.) **Оценка Визуального оформления** между Plotly, Seaborn и Matplotlib основана на следующем:

Plotly обычно предоставляет наиболее привлекательный визуальный дизайн по умолчанию, за которым следует Seaborn, а затем Matplotlib.

Основные причины этого отличия:

- Modern look: Plotly имеет современный и привлекательный внешний вид по умолчанию, который часто выглядит более профессиональным и модерним 4.
- Built-in themes: Plotly предлагает встроенные темы стилей, которые создают эстетически привлекательные графики с минимальной настройкой 4.
- Color schemes: Plotly использует хорошо продуманные цветовые схемы, которые делают графики более привлекательными и информативными.
- Typography: Plotly часто применяет современные шрифты и форматирование текста, улучшающее общее вид графиков.
- Consistency: Визуальный стиль Plotly более консистентен и хорошо продуман для разных типов графиков.

Seaborn также предлагает улучшенный визуальный дизайн по сравнению с Matplotlib, но менее продвинутый, чем у Plotly.

Matplotlib предоставляет базовый набор инструментов для создания графиков и требует больше ручной настройки для достижения желаемого визуального оформления.

Таким образом, при выборе библиотеки для создания графиков с привлекательным внешним видом Plotly обычно является лучшим выбором из-за своего современного и эстетически привлекательного дизайна по умолчанию.

4.) **Оценка Производительности** между Plotly, Seaborn и Matplotlib основана на следующем:

Matplotlib

- Matplotlib обычно имеет лучшую производительность, особенно при работе с большими наборами данных.
- Это связано с тем, что Matplotlib генерирует статичные изображения без необходимости в рендеринге интерактивных элементов.
- Matplotlib оптимизирован для быстрого отображения графиков, даже при обработке больших объемов данных.

Seaborn

- Seaborn построен на основе Matplotlib и наследует его преимущества в плане производительности.
- Seaborn также работает с статическими изображениями, поэтому его производительность близка к Matplotlib.
- Однако Seaborn может иметь небольшое снижение производительности из-за дополнительных функций и стилей, которые он применяет.

Plotly

- Plotly предлагает интерактивные графики, что значительно замедляет процесс рендеринга.
- При работе с большими наборами данных Plotly может стать очень медленным.
- Это связано с необходимостью отрисовки интерактивных элементов и передачей данных клиенту для обработки в браузере.

Таким образом, при работе с большими объемами данных Matplotlib обычно демонстрирует наилучшую производительность, за которым следует Seaborn, а затем Plotly. Это важный фактор при выборе библиотеки для визуализации данных, особенно когда речь идет о обработке больших наборов данных.

5.) **Оценка Настроек** (Кастомизация)между Plotly, Seaborn и Matplotlib основана на следующем:

Matplotlib

Matplotlib предлагает наибольшую гибкость и детализированную настройку:

- Предоставляет очень тонкий контроль над всеми аспектами визуализации, включая цвета, шрифты, размеры, положение элементов и т.д.
- Позволяет создавать полностью кастомизированные графики с минимальными ограничениями.
- Требует написания больше кода для достижения желаемого результата, но предоставляет максимальную свободу действий.

Plotly

Plotly также предлагает хорошие возможности настройки:

- Позволяет изменять цвета, стили и другие параметры графиков.
- Поддерживает создание сложных визуализаций, включая 3D графики.
- Предоставляет широкие возможности для добавления интерактивности и анимации.

Seaborn

Seaborn предлагает наименьшую гибкость в плане настройки:

- Предоставляет готовые стили и цветовые схемы.
- Позволяет легко настраивать некоторые параметры, но не настолько глубоко, как Matplotlib.
- Более ограничен в возможности создания полностью кастомизированных графиков.

Таким образом, Matplotlib предлагает наиболее детальную настройку, Plotly дает хороший баланс между готовыми решениями и возможностью настройки, а Seaborn предлагает наименее гибкую, но более удобную в использовании настройку.

Для визуализации этих критериев я построила самый простой график - горизонтальный бар-график, где каждая из библиотек (Plotly, Seaborn, Matplotlib) будет отображаться на разных уровнях для каждого из 5 критериев. Использовала библиотеку Plotly.

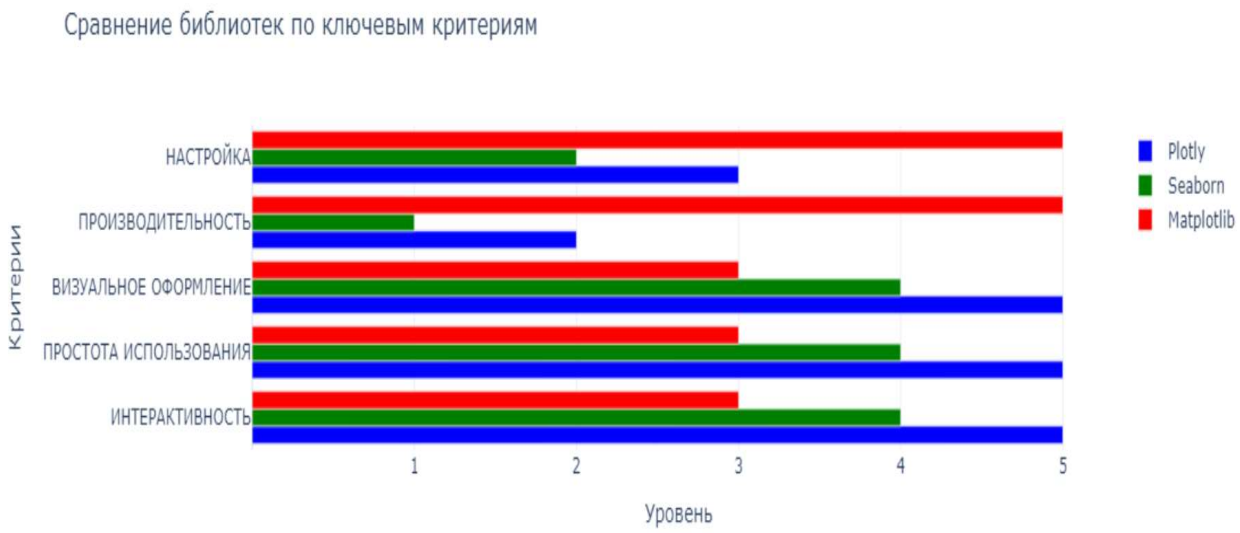
Для его создания я оценила по 5-бальной шкале каждую библиотеку по всем 5 критериям

	Matplotlib	Seaborn	Plotly
Интерактивность	5	4	3
Простота использования	5	4	3
Визуальное оформление	5	4	3
Производительность	2	1	5
Настройка	2	2	5

Итак, будем считать, что:

- **Ось X:** Уровень (например, от 1 до 5), где 1 — самый низкий, а 5 — самый высокий уровень по каждой из категорий.
- **Ось Y:** Категории: интерактивность, простота использования, эстетика, производительность, персонализация.
- Каждая категория будет иметь три столбца (по одному для каждой библиотеки) с разными цветами для каждой библиотеки (синим для Plotly, зелёным для Seaborn и красным для Matplotlib).

У нас получился вот такой симпатичный и наглядный график, благодаря которому мы можем сразу же сделать выбор в пользу той или иной библиотеки, если знаем, какие критерии для нас являются преимущественными.



Например: если для нас является самым важным визуальное оформление, то стоит выбрать библиотеку Plotly, а если мы новичок и нам сложности с настройками ни к чему, то стоит присмотреться к Matplotlib.

ВИЗУАЛИЗАЦИЯ – это бомба!!!!

Я три листа распиналась, приводя для вас все различия, чтобы вам помочь с выбором той или иной библиотеки. Вы потратили уйму времени, чтобы не просто прочитать, но и вникнуть в содержание моего текста. Вы держали в голове (ну или старались держать) всю прочитанную информацию, чтобы ее проанализировать и наконец-то суметь сделать выбор и тут вдруг: БАХ!

Вы увидели самый простейший график и моментально смогли сделать выбор. Здорово, правда?

Это как раз то, о чем я говорила ещё на самой первой странице.

Визуализация – это очень важно и актуально, это мощнейший инструмент в нашей жизни. Теперь вы согласны со мной?

Ниже я привела пример кода, написанного для библиотеки Plotly, благодаря которой у меня получился такой замечательный график

```python
import plotly.graph_objects as go

# Данные
categories = ['ИНТЕРАКТИВНОСТЬ', 'ПРОСТОТА ИСПОЛЬЗОВАНИЯ', 'ВИЗУАЛЬНОЕ ОФОРМЛЕНИЕ', 'ПРОИЗВОДИТЕЛЬНОСТЬ', 'НАСТРОЙКА']
plotly_scores = [5, 5, 5, 2, 3]
seaborn_scores = [4, 4, 4, 1, 2]
matplotlib_scores = [3, 3, 3, 5, 5]

# Создание графика
fig = go.Figure()

# Добавление данных для каждой библиотеки
fig.add_trace(go.Bar(
    y=categories,
    x=plotly_scores,
    name='Plotly',
    orientation='h',
    marker=dict(color='blue')
))

fig.add_trace(go.Bar(
    y=categories,
    x=seaborn_scores,
    name='Seaborn',
    orientation='h',
    marker=dict(color='green')
))

fig.add_trace(go.Bar(
    y=categories,
    x=matplotlib_scores,
    name='Matplotlib',
    orientation='h',
    marker=dict(color='red')
))

# Обновление макета
fig.update_layout(
    barmode='group',
    title='Сравнение библиотек по ключевым критериям',
    xaxis_title='Уровень',
    yaxis_title='Критерии',
    xaxis=dict(tickvals=[1, 2, 3, 4, 5], ticktext=['1', '2', '3', '4', '5']),
    template='plotly_white'
)

fig.show()
```

8 Устроим соревнования

Всё замечательно, мы нашли описание библиотек на сайтах документации, мы прочитали множество информации, сравнили их ключевые показатели.

Но мы не подумали ещё об одном очень важном факторе – а как, собственно, будут выглядеть графики?

На мой взгляд это так же является очень важным фактором при проведении сравнения библиотек, ведь если в конечном счете все графики будут выглядеть одинаково, объем написанного кода будет одинаковым, то программисту необходимо ориентироваться на личные предпочтения (выбрать ту библиотеку, с которой он больше знаком, чтобы быстро построить график и презентовать).

Все понимают, что написать код для графика – это только часть работы.

Самый сложный и трудоемкий процесс – это сбор тех самых данных (группировка, компоновка, чистка и т.п.). И на него уходит бо'льшая часть времени, а значит время на создание графика должно быть минимально возможным, чтобы в целом уложиться в отведенное для работы время.

Будем «испытывать все три наших библиотеки на трех одинаковых графиках:

- линейный график
- столбчатая диаграмма
- точечная диаграмма

Эти графики будут выглядеть похоже, но каждая библиотека имеет свои особенности в плане стиля и функциональности.

Для проведения своих испытаний я воспользовалась готовым набором данных о продажах с сайта **Kaggle.com**: https://www.kaggle.com/datasets/bhadramohit/customer-shopping-latest-trends-dataset/data

Чтобы не было проблем с ключами и логинами при обращении в базе данных и по этой причине у вас не возникло проблем с тестированием кода, я отформатировала скачанные данные в файл в формате Excel (на самом деле, на моей работе мне приходится работать именно с данными в файлах Excel). Файл с данными сохранен также в папке проекта на GitHub.

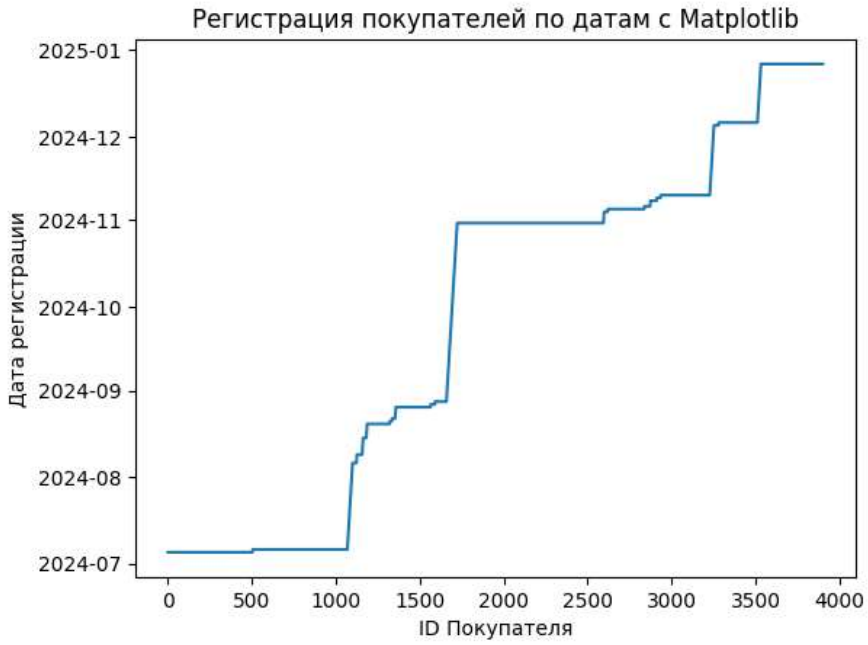
Так какие три одинаковых графика на основе данных о продажах можно построить с помощью библиотек Matplotlib, Seaborn и Plotly?

1. Линейный график (Тренды по времени)

Они используются для отображения данных, которые изменяются во времени (как изменение одной переменной влияет на другую). Хорошо подходят для таких Трендов, как: динамика изменения цен, температура, колебания продаж. Помогают выявить цикличность, сезонность и тп.
Из моей базы данных я взяла сведения об ID Покупателей и датах регистрации, чтобы выявить в какие же дни было больше зарегистрированных, а в какие не очень.

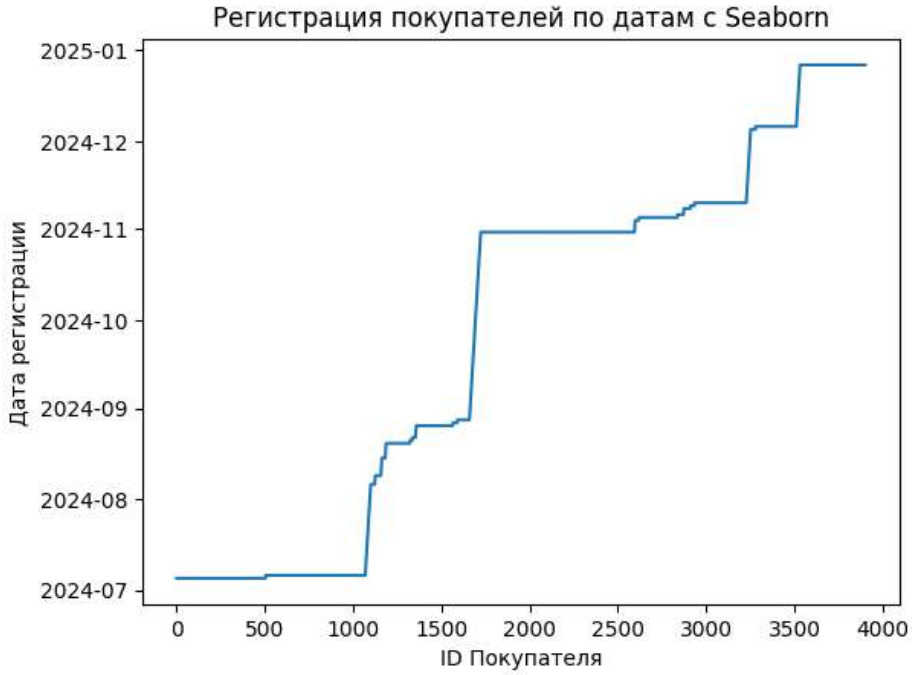
Matplotlib:

```
import matplotlib.pyplot as plt

customer_ID = data['IDПокупателя']
data_registration = data['Дата регистрации']

plt.plot(customer_ID, data_registration)
plt.title('Регистрация покупателей по датам с Matplotlib')
plt.xlabel('ID Покупателя')
plt.ylabel('Дата регистрации')
plt.show()
```



Seaborn:

```
import seaborn as sns
import matplotlib.pyplot as plt

customer_ID = data['IDПокупателя']
data_registration = data['Дата регистрации']

sns.lineplot(x=customer_ID, y=data_registration)
plt.title('Регистрация покупателей по датам с Seaborn')
plt.xlabel('ID Покупателя')
plt.ylabel('Дата регистрации')
plt.show()
```



Plotly:

```
import plotly.graph_objects as go

customer_ID = data['IDПокупателя']
data_registration = data['Дата регистрации']

fig = go.Figure(data=go.Scatter(x=customer_ID, y=data_registration, mode='markers'))
fig.update_layout(title='Регистрация покупателей по датам с Plotly', xaxis_title='ID Покупателя', yaxis_title='Дата регистрации')
fig.show()
```

Подведем итоги:

Характеристика	Matplotlib	Seaborn	Plotly
Простота использования	Высокая гибкость, но требуется больше кода	Легкий для создания, много настроек по умолчанию	Очень прост в использовании для интерактивных графиков
Кастомизация	Очень высокая (цвета, линии, аннотации, шрифты и т.д.)	Меньше кастомизации, но все еще достаточно для большинства задач	Очень высокая, поддержка анимаций и всплывающих подсказок
Интерактивность	Отсутствует по умолчанию, нужна дополнительная настройка	Отсутствует интерактивность, но возможно использовать Matplotlib	Полностью интерактивный, поддержка масштабирования, навигации
Анимация	Поддержка через FuncAnimation (достаточно сложная)	Нет встроенной поддержки анимации	Простая и мощная поддержка анимации (в том числе переходы)
Производительность	Очень хорошая для статичных графиков	Высокая для небольших данных	Отличная для интерактивных графиков с большими объемами
Статичность графиков	Отлично подходит для создания статичных графиков	Статичные графики, но с возможностью вывода в SVG, PNG	Отличная для статичных графиков в виде HTML

Вывод графиков	Вывод через Matplotlib (PNG, PDF, SVG и другие форматы)	Вывод в статичные изображения (можно интегрировать с Matplotlib)	Прямой вывод в интерактивном формате HTML/JS
Поддержка веб-интеграции	Ограниченная (через mpld3, matplotlib для web)	Можно экспортировать в HTML, но без интерактивности	Интеграция в веб-приложения с возможностью взаимодействия

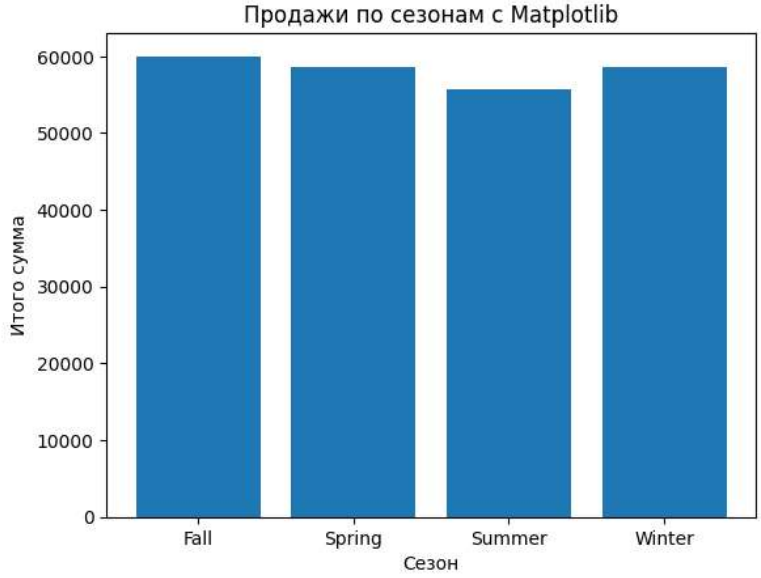
2. Столбчатая диаграмма (Сравнение продаж по регионам)

Они особенно полезны, когда нужно сравнить количество или величину разных категорий данных. Хорошо подходят для сравнения данных, которые принадлежат разным категориям (сумма продаж за месяц, сравнение оценок учеников), для визуализации абсолютных величин (количество произведенных товаров на каждом заводе), для отображения части целого (доля отдела в общей прибыли).

Matplotlib:

```
import matplotlib.pyplot as plt

season = data['Сезон']
sales_sum = data.groupby('Сезон')['Сумма'].sum().reset_index()

print(sales_sum)
plt.bar(sales_sum['Сезон'], sales_sum['Сумма'])
plt.title('Продажи по сезонам с Matplotlib')
plt.xlabel('Сезон')
plt.ylabel('Итого сумма')
plt.show()
```

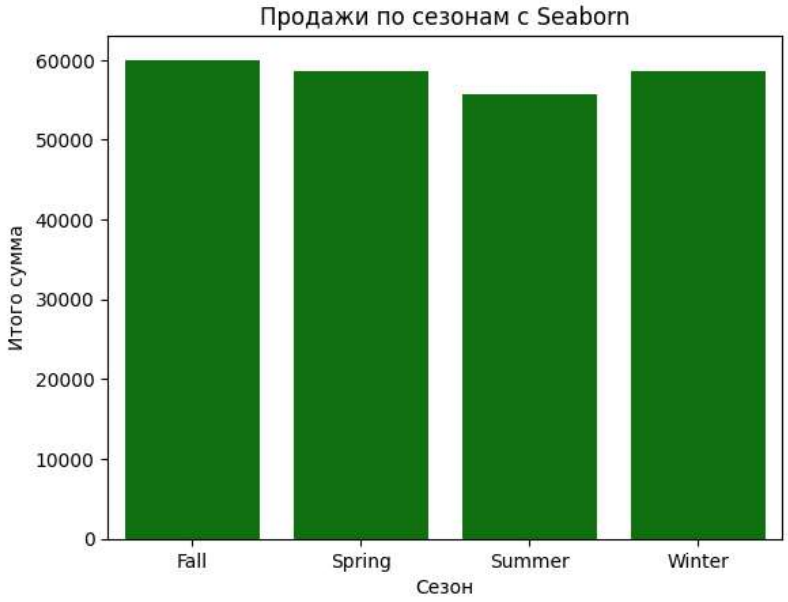
```
    Сезон  Сумма
0    Fall  60018
1  Spring  58679
2  Summer  55777
3  Winter  58607
```



Seaborn:

```
import seaborn as sns
import matplotlib.pyplot as plt

season = data['Сезон']
sales_sum = data.groupby('Сезон')['Сумма'].sum().reset_index()

print(sales_sum)
sns.barplot(x='Сезон', y='Сумма', data=sales_sum, color='green')
plt.title('Продажи по сезонам с Seaborn')
plt.xlabel('Сезон')
plt.ylabel('Итого сумма')
plt.show()
```

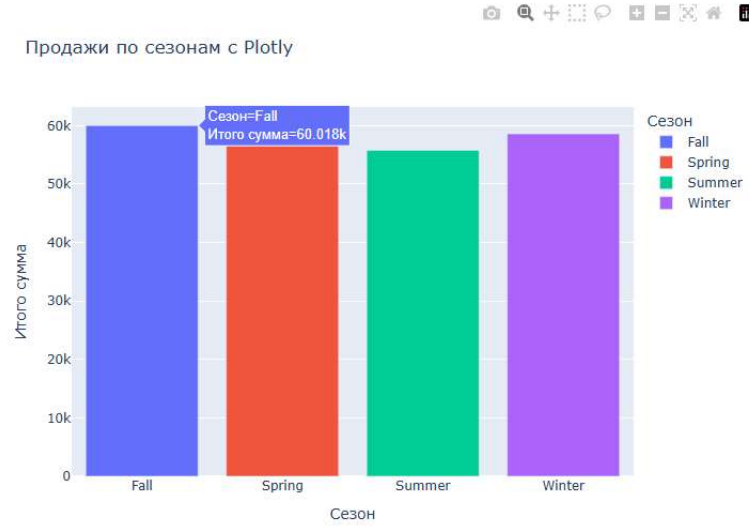
```
    Сезон  Сумма
0    Fall  60018
1  Spring  58679
2  Summer  55777
3  Winter  58607
```



Plotly:

```
import plotly.express as px

season = data['Сезон']
sales_sum = data.groupby('Сезон')['Сумма'].sum().reset_index()

print(sales_sum)
fig = px.bar(sales_sum,
             x='Сезон',
             y='Сумма',
             color='Сезон',  # Цвет столбцов по каждому клиенту
             title='Продажи по сезонам с Plotly',
             labels={'Сезон': 'Сезон', 'Сумма': 'Итого сумма'})

fig.show()
```

```
   Сезон  Сумма
0   Fall  60018
1 Spring  58679
2 Summer  55777
3 Winter  58607
```



Подведем итоги:

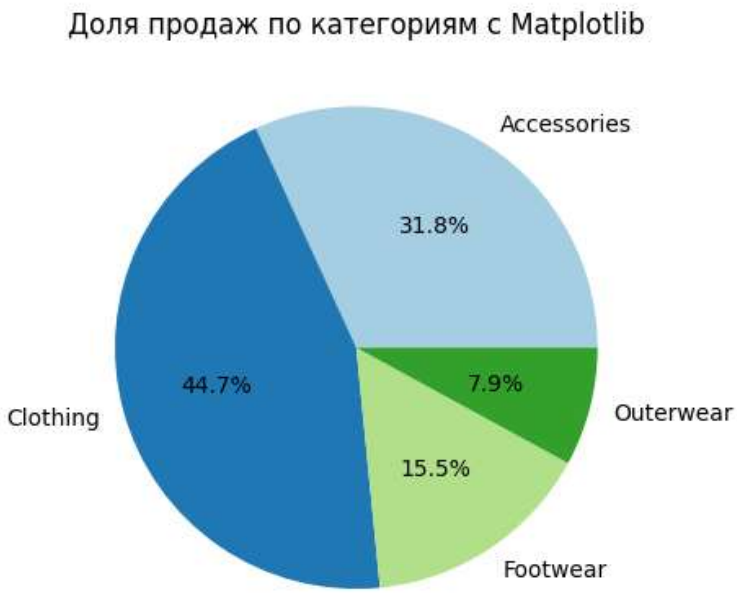
Характеристика	***Matplotlib***	***Seaborn***	***Plotly***
Простота использования	Прост в создании, но требует параметров для кастомизации	Очень удобен для создания с минимумом кода	Очень простой синтаксис, поддержка интерактивности
Кастомизация	Высокая, можно изменить цвета, толщину, стили линий	Хорошая кастомизация, поддерживает изменения цветов и стилей	Очень высокая, поддержка анимаций и переходов
Интерактивность	Не поддерживает интерактивность (без дополнительных библиотек)	Нет интерактивности по умолчанию	Полностью интерактивный, поддержка масштабирования, фильтрации
Анимация	Не поддерживает анимацию по умолчанию	Нет встроенной поддержки анимации	Поддержка анимации через динамичные данные
Производительность	Отличная для малых и средних объемов данных	Хорошая производительность для статистических графиков	Хорошая, но может снижаться с большими объемами данных
Статичность графиков	Отлично подходит для статичных визуализаций	Статичные графики, можно использовать Matplotlib для сохранения	Отлично подходит для статичных графиков (можно сохранить в виде PNG или JPEG)
Вывод графиков	Вывод через Matplotlib (PNG, PDF, SVG и другие форматы)	Вывод в статичные изображения через Matplotlib	Прямой вывод в HTML или PNG/JPEG (для статичных графиков)
Поддержка веб-интеграции	Ограниченная (через mpld3, matplotlib для web)	Можно экспортировать в HTML с использованием других инструментов	Отлично интегрируется в веб-приложения (встроенная поддержка JavaScript)

3. Круговая диаграмма (Доля продаж по категориям)

В основном используются для визуализации долей или процентов, которые составляют часть от целого. Круговая диаграмма наглядно показывает, как каждая категория (сегмент) соотносится с другим. Хорошо подходят для распределения продаж по регионам, для анализа рыночных долей компаний в определенной отрасли, процентное распределение клиентов по возрастным группам или категориям товаров.

Matplotlib:

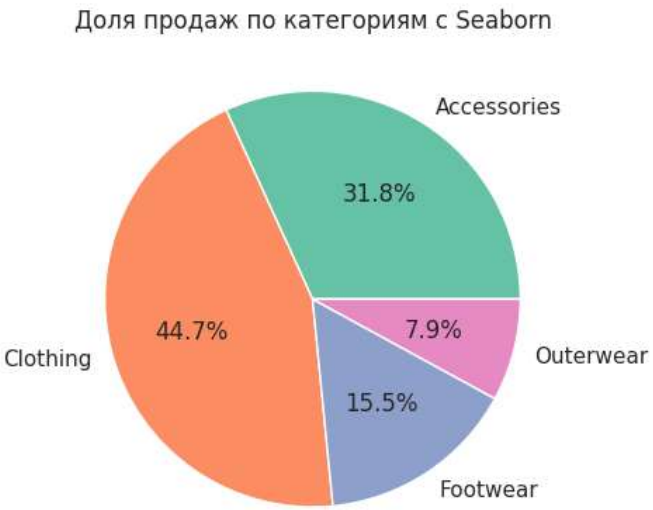
```
import matplotlib.pyplot as plt

sales_sum = data.groupby('Категория')['Сумма'].sum()

plt.pie(sales_sum, labels=sales_sum.index, autopct='%1.1f%%', colors=plt.cm.Paired.colors)
plt.title('Доля продаж по категориям с Matplotlib')
plt.show()
```



Seaborn:

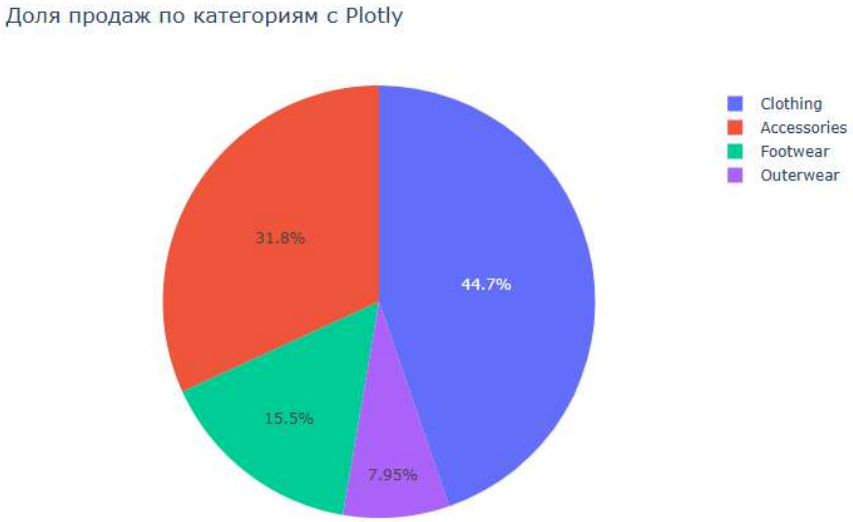
Seaborn не поддерживает прямое создание круговых диаграмм, но для этого можно использовать matplotlib

```
import seaborn as sns
import matplotlib.pyplot as plt

sales_sum = data.groupby('Категория')['Сумма'].sum()

sns.set(style="whitegrid")

plt.pie(sales_sum, labels=sales_sum.index, autopct='%1.1f%%', colors=sns.color_palette("Set2", len(sales_sum)))
plt.title('Доля продаж по категориям с Seaborn')
plt.show()
```



Plotly:

```
import plotly.express as px

sales_sum = data.groupby('Категория')['Сумма'].sum().reset_index()

fig = px.pie(sales_sum, names='Категория', values='Сумма', title='Доля продаж по категориям с Plotly')
fig.show()
```



Подведем итоги:

Характеристика	*Matplotlib*	*Seaborn*	*Plotly*
Простота использования	Прост в создании, но кастомизация ограничена	Кастомизация через matplotlib	Очень интуитивен для создания красивых графиков
Кастомизация	Ограниченная (не такой гибкий, как линейный или столбчатый)	Ограниченная кастомизация (для продвинутых пользователей)	Высокая, поддержка динамичных визуализаций и анимаций

Интерактивность	Нет интерактивности по умолчанию	Нет интерактивности, но можно интегрировать с Matplotlib	Полностью интерактивный, можно настраивать анимации
Анимация	Можно создать с помощью matplotlib.animation	Нет встроенной поддержки анимации	Множество возможностей для анимации и переходов
Производительность	Хорошая для небольших наборов данных	Хорошая для небольших наборов данных	Отличная производительность для динамичных графиков
Статичность графиков	Хорошо для статичных графиков	Используется для статичных графиков, интеграция с Matplotlib	Можно сохранять графики как HTML, SVG или PNG
Вывод графиков	Вывод через Matplotlib (PNG, PDF, SVG и другие форматы)	Использует Matplotlib для вывода	Прямой вывод в HTML с возможностью экспорта в другие форматы
Поддержка веб-интеграции	Ограниченная (через mpld3, matplotlib для web)	Можно экспортировать с помощью Matplotlib	Отличная поддержка интеграции в веб-приложения и отчетности

9 Самые красивые графики для каждой библиотеки

Поскольку речь изначально идет про библиотеки для визуализации, то эта самая визуализация должна быть красивой. И в процессе своего исследования я задалась вопросом: А чем же может «похвастаться» каждая из наших сравниваемых библиотек, на какую самую красивую диаграмму или график способна каждая из них?

Предлагаю провести решающий раунд!

9.1 Для библиотеки Matplotlib

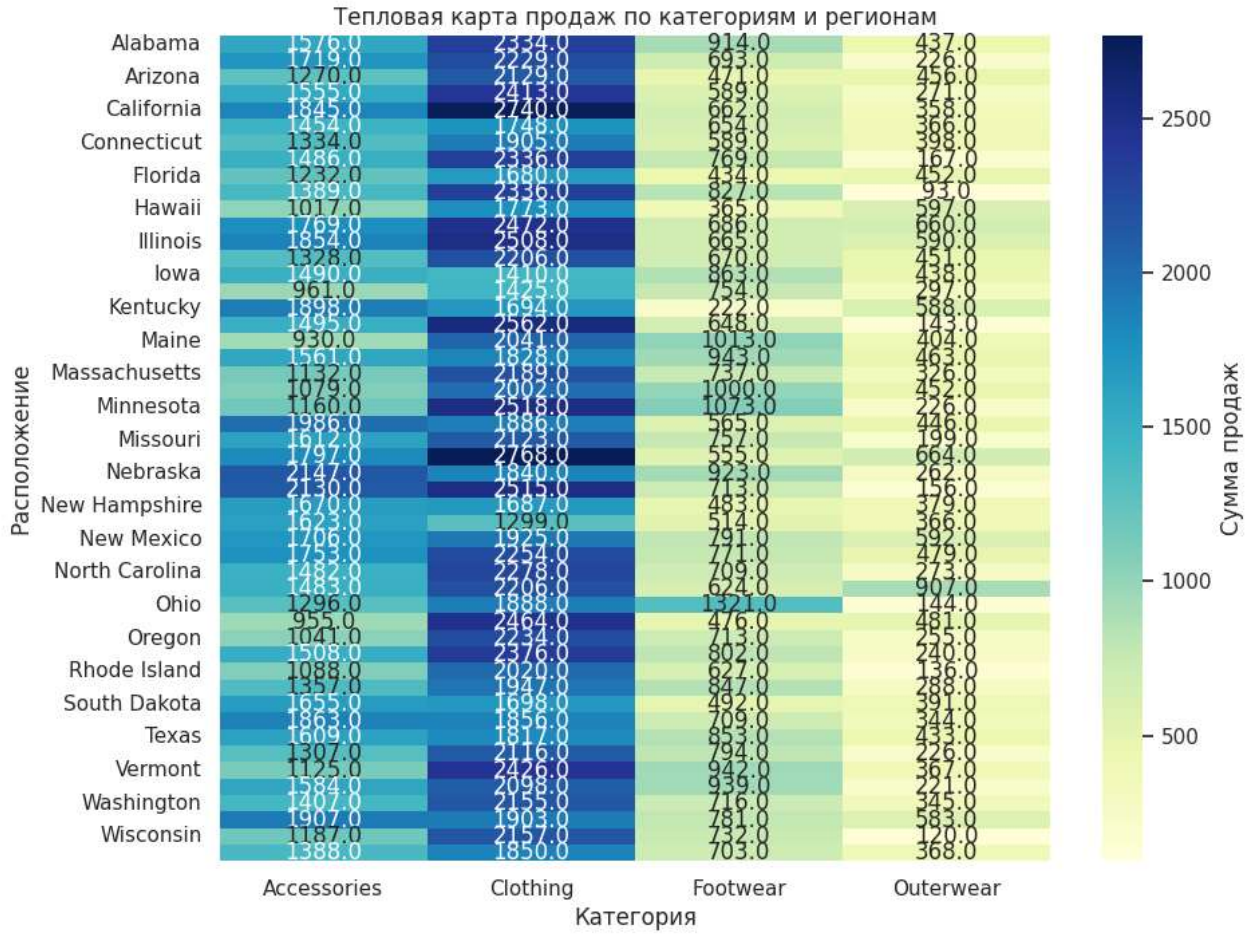
на наборе данных о продажах магазина, одним из самых визуально привлекательных и информативных графиков может быть **"Тепловая карта" (Heatmap)**, которая отображает корреляции или распределения значений, например, доходов по дням или категориям товаров. Эта визуализация помогает выявить скрытые закономерности и быстро понять распределение данных.

Тепловая карта в Matplotlib

Для примера, мы можем построить тепловую карту для отображения доходов по категориям товаров в разные дни. Тепловая карта представляет собой матрицу, где цвет ячеек соответствует величине значения в данной ячейке. Это очень удобный способ визуализации, который позволяет сразу увидеть, где были наибольшие продажи, а где наименьшие.

Пример кода

```python
import matplotlib.pyplot as plt
import seaborn as sns

pivot_table = data.pivot_table(values='Сумма', index='Расположение', columns='Категория', aggfunc='sum', fill_value=0)

# Настройка визуализации
plt.figure(figsize=(10, 8))  # Размер графика
sns.set(style="whitegrid")   # Стиль графика

# Строим тепловую карту
ax = sns.heatmap(pivot_table, annot=True, fmt='.1f', cmap='YlGnBu', cbar_kws={'label': 'Сумма продаж'})

# Добавляем заголовок
plt.title('Тепловая карта продаж по категориям и регионам')

# Показываем график
plt.show()
```



Почему это красивый график:

- **Тепловая карта** позволяет быстро увидеть, как распределяются доходы по различным дням и категориям товаров, а также выделить те моменты, когда продажи были наиболее или наименее успешными.
- Цветовая палитра помогает визуально воспринимать данные: более темные цвета показывают высокие значения (большие доходы), а более светлые — низкие значения.

9.2 Для библиотеки Seaborn,

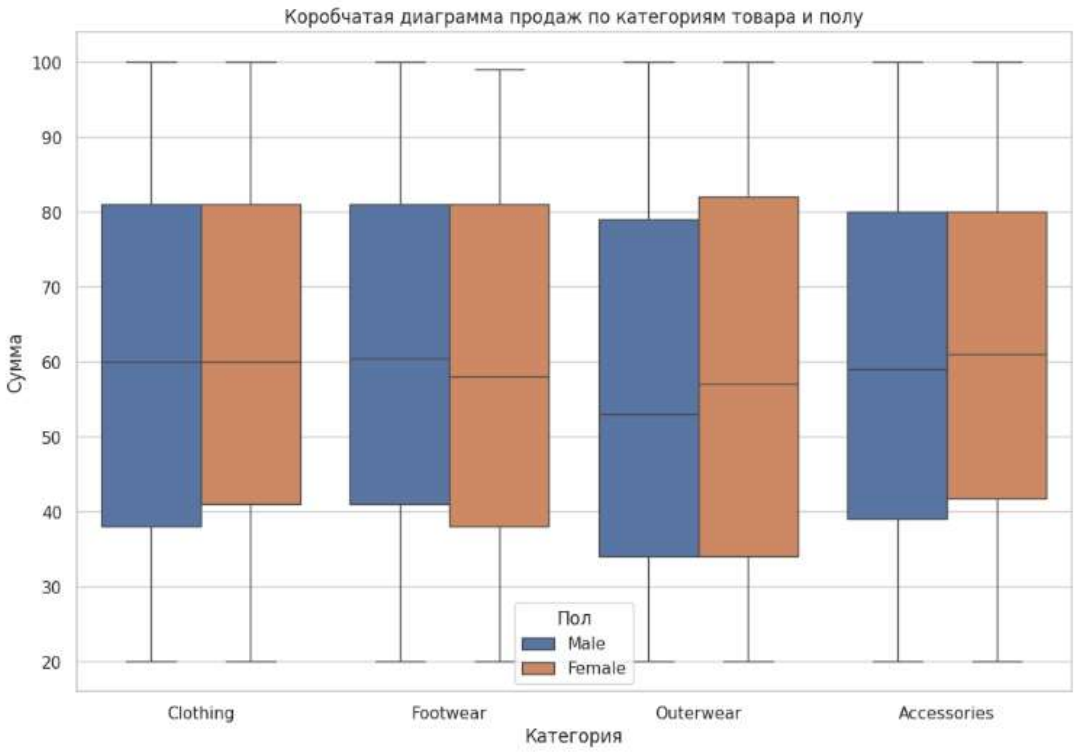
одним из самых красивых и информативных графиков, который можно построить на основе набора данных о продажах магазина, является **коробчатая диаграмма** (**Box Plot**). Это график, который отлично показывает распределение данных, включая медиану, квартили и выбросы. Он особенно полезен для анализа вариации доходов или цен по категориям товаров.

Пример графика: Коробчатая диаграмма (Box Plot)

Коробчатая диаграмма в Seaborn позволяет увидеть важную информацию о распределении значений и может быть полезной для анализа, например, распределения доходов по категориям товаров, выявления выбросов и т. д.

Пример кода

```
import seaborn as sns
import matplotlib.pyplot as plt

# Настройка визуализации
plt.figure(figsize=(12, 8))  # Размер графика

# Строим коробчатую диаграмму с использованием Seaborn
sns.set(style="whitegrid")  # Стиль графика
sns.boxplot(x='Категория', y='Сумма', hue='Пол', data=data)

# Добавляем заголовок
plt.title('Коробчатая диаграмма продаж по категориям товара и полу')

# Показываем график
plt.show()
```



Почему это красивый график:

- **Коробчатая диаграмма** предоставляет четкую информацию о распределении данных: она показывает медиану (линии внутри коробки), квартали (границы коробки), а также выбросы (точки за пределами "усы").
- Этот график помогает легко понять, есть ли какие-то аномалии (например, выбросы) в данных о доходах по категориям товаров.
- Цветовая палитра **Set2** делает график ярким и легко воспринимаемым.
- Дополнительная категориальная переменная для разделения данных на группы по полу. Это будет означать, что для каждого пола будет нарисована отдельная коробка, отображая различие в продажах по регионам в каждой категории товара.

9.3 Для библиотеки Plotly

один из самых красивых и интерактивных графиков, который можно построить на наборе данных о продажах магазина — это **график с пузырьками (Bubble Chart)**. Этот тип графика позволяет отобразить три переменные на двумерной плоскости, что делает его идеальным для анализа взаимосвязи между несколькими характеристиками, такими как цена, количество и доход.

Почему пузырьковая диаграмма в Plotly?

1. **Интерактивность**: Пользователи могут наводить курсор на пузырьки, чтобы увидеть точные значения переменных, а также масштабировать и перемещать график.
2. **Три переменные**: Пузырьки могут быть использованы для отображения трех переменных, где:
    - Позиция на оси X — одна переменная (например, категория),
    - Позиция на оси Y — другая переменная (например, сумма),
    - Размер пузырька — третья переменная (например, доход).

Пример кода

```
import plotly.express as px

# Строим анимированный график с пузырьками
fig = px.scatter(data,
                 x='Категория',  # Категории товара по оси X
                 y='Сумма',      # Сумма продаж по оси Y
                 size='Сумма',   # Размер пузырьков по сумме продаж
                 color='Частота покупок',  # Цвет пузырьков по частоте покупок
                 animation_frame='Сезон',  # Анимация по сезонам
                 animation_group='Категория',  # Группировка анимации по категориям товара
                 title='Анимация продаж с пузырьками по категориям товара и регионам')

# Показываем интерактивный график с пузырьками
fig.show()
```

Почему это красивый график:

1. **Интерактивность**: Пузырьковая диаграмма Plotly предоставляет пользователю возможность взаимодействовать с графиком, увеличивать его, перемещать и получать подробную информацию по каждому пузырьку.
2. **Три переменные**: Одним графиком можно отобразить сразу несколько аспектов данных (цена, количество, доход), что дает полное представление о взаимосвязях.
3. **Цветовая кодировка**: Разделение данных по категориям с использованием разных цветов помогает лучше воспринимать информацию и выделять ключевые различия.
4. **Подсказки**: Когда пользователь наводит курсор на пузырек, появляется дополнительная информация о товаре, что делает график еще более информативным.

Преимущества пузырьковой диаграммы в Plotly:

- **Интерактивность** позволяет пользователю исследовать данные, увеличивать области интереса и получать точную информацию о значениях в любое время.
- **Цвета и размеры пузырьков** делают график визуально привлекательным и информативным.
- Этот тип графика идеально подходит для анализа **взаимосвязей** между несколькими переменными и помогает выделить закономерности, такие как высокодоходные товары или товары с большими продажами.

11 Рекомендации по выбору библиотеки

Как вы уже поняли, вне зависимости от моего исследования, выбор необходимой библиотеки всё равно предстоит сделать Вам. И только Вам, в зависимости от конкретных требований Вашего проекта (типа графика, уровня интерактивности, сложности и т.п.) необходимого, уровня Ваших

знаний и предпочтений в стиле работы с данными необходимо остановить свой выбор в пользу какой-либо одной из библиотек, а лучше их комбинировать.

Совершенствуя свои навыки, расширяя свои знания, изучая всё новые и новые методы и подходы Вы будете прежде всего сами гордиться собой и своими достижениями, а значит и окружающие Вас оценят.

Мне очень хочется верить, что своей работой я смогла Вам помочь в вашем выборе.

Ну и в качестве «вишенки на торте» я сделала вот такую не хитрую табличку, указав свои рекомендации по выбору библиотек в зависимости от конкретных случаев использования

Характеристика	*Matplotlib*	*Seaborn*	*Plotly*
Гибкость и кастомизация	+	-	±
Сложные специфичные графики	+	-	±
Большие объемы данных	+	-	±
Научные исследования	+	-	±
Интеграция с другими библиотеками	+	-	±
Быстрое создание статистических графиков	-	+	±
Бизнес-аналитика и финансовые проекты	-	+	±
Препятствующие данные	-	+	±
Эстетически привлекательные графики	-	+	±
Проекты с ограниченным временем разработки	-	+	±
Интерактивные визуализации	-	-	+
Веб-интерфейсы	-	-	+
Географические данные	-	-	+
Финансовые графики	-	-	+
Проекты с большим количеством пользователей	-	-	+

Объяснение символов:

- Хорошо подходит для данного случая использования
- Менее подходит для данного случая использования
- ± - Может быть использована в данном случае, но не является оптимальным выбором