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

**Автор**

**Кенеман Михаил Георгиевич**

**Введение**

Визуализация данных - это графическое представление информации и данных. Используя визуальные элементы, такие как диаграммы и картографические изображения, инструменты визуализации данных предоставляют доступный способ увидеть и понять тенденции, отклонения и закономерности в данных. Эта практика имеет решающее значение в процессе обработки данных, поскольку помогает сделать данные более понятными и пригодными для применения широким кругом пользователей, от бизнес-профессионалов до специалистов по обработке данных.

**Что такое визуализация данных?**

Визуализация данных переводит сложные наборы данных в

визуальные форматы, которые легче воспринимаются человеческим

мозгом. Это может включать различные визуальные инструменты,

такие как:

* **Диаграммы**: столбчатые, линейчатые, круговые диаграммы и т.д.
* **Графики**: точечные диаграммы, гистограммы и т.д.
* **Карты**: Географические карты, тепловые карты и т.д.
* **Информационные панели**: интерактивные платформы,

объединяющие несколько визуализаций.

Основная цель визуализации данных - сделать данные более

доступными и простыми для интерпретации, позволяя пользователям быстро выявлять закономерности, тенденции и отклонения. Это

особенно

важно в контексте больших данных, где сам объем информации может быть непосильным без эффективных методов визуализации. Точная

визуализация данных очень важна для маркетинговых исследований, где можно визуализировать как числовые, так и категориальные данные, что помогает повысить эффективность анализа.

## Почему важна визуализация данных?

### Визуализация данных:

### - выявляет тенденции в области обработки данных

### - дает представление о данных

### - помещает данные в правильный контекст

### - экономит время

### - рассказывает историю данных

**Некоторые методы визуализации данных**

**Столбчатые диаграммы:** Столбчатые диаграммы, идеально

подходящие для сравнения категориальных данных или отображения частот, обеспечивают четкое визуальное представление

значений.

**Линейные диаграммы:** Линейные диаграммы, соединяющие точки данных и выявляющие закономерности и колебания, идеально

подходят для иллюстрации тенденций во времени.

**Круговые диаграммы:** Круговые диаграммы эффективны для

отображения частей целого и предлагают простой способ понять

пропорции и проценты.

**Точечные диаграммы:** Демонстрируют взаимосвязи между двумя

переменными, выявляя закономерности и выбросы по разбросанным точкам данных.

**Гистограммы:** Изображают распределение непрерывной

переменной, предоставляя представление о лежащих в основе

данных шаблонах.

**Тепловые карты:** Визуализируйте сложные наборы данных с помощью цветового кодирования, подчеркивая различия и корреляции в

матрице.

**Пузырьковые диаграммы:** Улучшите точечные диаграммы, введя третье измерение с помощью различных размеров пузырьков, что

позволяет получить дополнительную информацию.

**Древовидные карты:** Эффективно представляют иерархические структуры данных, разбивая категории на вложенные

прямоугольники.

**Графики скрипки**: Графики скрипки сочетают аспекты

прямоугольных графиков и графиков плотности ядра,

обеспечивая подробное представление распределения данных.

**3D-графики поверхности**: 3D-графики поверхности

визуализируют трехмерные данные, иллюстрируя, как изменяется

переменная отклика по отношению к двум предикторным

переменным.

## Преимущества и недостатки визуализации данных

### Преимущества визуализации данных:

* **Расширенное сравнение:** визуализация характеристик двух элементов или сценариев упрощает анализ, экономя время по сравнению с традиционным изучением данных.
* **Усовершенствованная методология:** Графическое представление данных обеспечивает лучшее понимание ситуаций, примером чего являются такие инструменты, как Google Trends, иллюстрирующие отраслевые тенденции в графической форме.
* **Эффективный обмен данными:** Визуальное представление данных облегчает эффективную коммуникацию, делая информацию более усваиваемой и привлекательной по сравнению с обменом необработанными данными.
* **Анализ продаж:** Визуализация данных помогает специалистам по продажам понимать тенденции продаж продукции, выявлять влияющие факторы с помощью таких инструментов, как тепловые карты, а также понимать типы клиентов, влияние географии и поведение постоянных клиентов.
* **Выявление связей между событиями:** Обнаружение корреляций между событиями помогает компаниям понять внешние факторы, влияющие на их эффективность, такие как резкий рост онлайн-продаж в праздничные сезоны.
* **Изучение возможностей и тенденций:** Визуализация данных позволяет бизнес-лидерам выявлять закономерности и возможности в обширных наборах данных, что позволяет глубже понимать поведение клиентов и понимать новые бизнес-тенденции.

### Недостатки визуализации данных:

* **Может отнимать много времени:**Создание визуализаций может быть трудоемким процессом, особенно при работе с большими и сложными наборами данных.
* **Может вводить в заблуждение:** Хотя визуализация данных может помочь выявить закономерности и взаимосвязи в данных, она также может ввести в заблуждение, если выполнена неправильно. Визуализации могут создавать впечатление закономерностей или тенденций, которых может и не быть, что приводит к неверным выводам и неправильному принятию решений.
* **Может быть сложно интерпретировать:** Некоторые типы визуализаций, например, те, которые включают 3D или интерактивные элементы, могут быть сложными для интерпретации и понимания.
* **Может подходить не для всех типов данных:** определенные типы данных, такие как текстовые или аудиоданные, могут плохо поддаваться визуализации. В этих случаях могут оказаться более подходящими альтернативные методы анализа.
* **Может быть доступно не всем пользователям:**У некоторых пользователей могут быть нарушения зрения или другие нарушения, которые затрудняют или делают невозможным интерпретацию визуализаций. В этих случаях могут потребоваться альтернативные методы представления данных для обеспечения доступности.

***Matplotlib***

Одной из важных частей в анализе данных является графическое визуализация. В Python есть много дополнительных библиотек для создания статических или динамических визуализаций. Рассмотрим три из них. Это matplotlib, seaborn и plotly. Matplotlib является наиболее известной библиотекой в Python. Это связано, видимо, с тем, что она является одной из первых разработанных для Python библиотек для визуализаций со столь обширными возможностями.

Matplotlib используют для визуализации данных любой сложности. Библиотека позволяет строить разные варианты графиков: линейные, трёхмерные, диаграммы рассеяния и другие, а также комбинировать их.

Библиотека Matplotlib — это пакет для визуализации данных в Python. Pyplot — это модуль в пакете Matplotlib. Его часто можно видеть в коде как matplotlib.pyplot. Модуль помогает автоматически создавать оси, фигуры и другие компоненты, не задумываясь о том, как это происходит. Именно Pyplot используется в большинстве случаев.

***Линейные графики***

Основной функцией для построения линейных графиков является plt.plot(). В качестве аргументов в функцию нужно передать значения независимой переменной (например, x) и зависимой (например, y). Если других аргументов не передавать, то будет отрисован график по значениям x и y с параметрами по умолчанию. В качестве аргументов в функцию могут быть переданы различные параметры графика, а именно цвет линии color=’ ‘, тип линии linestyle=’ ‘ (напимер, штриховая, штрих-пунктирная, пунктирная и т.п), наличие маркера, а также его форма и размер marker=’ ‘, markersize=’ ‘. Для обозначения осей, названия графиков и других обозначений служат функции plt.xlabel(), plt.ylabel(), plt.title(), plt.legend(). Аргументы этих функций позволяют задать параметры шрифта вывода и расположения надписей. Для вывода на экран подготовленного графика служит функция plt.show().

На одной координатной сетке можно вывести несколько графиков. Для этого достаточно задать в качестве аргументов в функцию plt.plot() несколько пар зависимой и независимой переменных. Для каждого графика можно вызывать отдельную функцию plt.plot(). Каждый график может быть выведен со своими параметрами по цвету, типу линии и т. п.

***Диаграмма рассеяния***

Диаграмма рассеяния создаётся с помощью метода plt.scatter(). Аргументами в данном случае могут быть две переменных, заданных списками. Диаграмма представляет собой совокупность точек и наглядно показывает взаимосвязь переменных. По плотности расположения точек можно судить о вероятности некоторых событий, корреляции переменных. Обозначение осей, название графика, легенда создаются аналогично линейным графикам.

***Столбчатая диаграмма***

Столбчатая диаграмма — это график, представляющий категорию данных с помощью прямоугольных столбцов с длиной и высотой, пропорциональной значениям, которые они представляют. Столбчатые диаграммы строятся при помощи метода plt.bar(). Исходные данные и оформление графиков аналогично диаграммам рассеяния.

***Гистограммы***

Гистограммы – популярный метод визуализации данных. Они используются для отображения принципов распределения информации. Чем больше в диапазоне значений, тем выше (или длиннее, если гистограмма горизонтальная) окажется получившаяся полоса. При работе с гистограммами одна из осей отображает некоторую категорию, а вторая соответствует её дискретному значению.

Для построения гистограммы используется метод plt.hist(). В качестве аргументов необходимо передать набор значений, который принимает некоторая переменная, и число диапазонов, внутри каждого из которых будет подсчитываться количество значений переменной. Набор значений может быть представлен списком, а для указания числа интервалов служит параметр bins. Таким образо фрагмент кода построения гистограммы может быть представлен, как

values = […, …, …]

plt.hist(values, bins=8)

plt.show()

В данном примере в качестве исходных данных принимается список из, например, чисел с плавающей запятой, диапазон изменения этих чисел делится на 8 поддиапазонов, внутри каждого из которых будет подсчитываться число попавших туда значений исходных данных.

***Круговые диаграммы***

Круговые диаграммы используются в основном для отображения состава некоторых групп. К примеру можно в качестве исходных данных взять количество деревьев различных пород на заданной площади леса.

Тогда площадь сектора на круговой диаграмме , соответствующего определённой породе, быдет пропорциональна количеству деревьев этой породы.

Для построения круговых диаграмм используется метод plt.pie(vals, labels=labels). Здесь vals – список исходных данных. В нашем примере это целочисленные значения, соответствующие количеству деревьев разных пород, а labels – название пород.

Для детализации круговой диаграммы можно использовать параметр autopct. Если теперь записать в код для построения круговой диаграммы строчку

plt.pie(vals, labels=labels, autopct=’%1.1f%%’),

то на диаграмме появятся размеры долей в процентах с точностью до десятых долей. Можно выделить некоторые сектора дополнительно, «вытащив» на некоторое расстояние сектор из круга. Для этого служит параметр explode, представляющий из себя кортеж, число элементов которого соответствует числу секторов диаграммы. Если элемент не равен 0, то сектор,соответствующий по индексу элементу списка labels, будет выдвинут из круговой диаграммы на соответствующую величину. Таким образом можно выделять сектора с наибольшим или с наименьшим значением.

***Отображение нескольких графиков с помощью***

***plt.subplot()***

Если необходимо вывести несколько графиков в разных координатных осях, то для этого служит функция plt.subplot(). Аргументами этой функции служат три целых числа. Первые два числа определяют число и расположение полей графиков аналогично расположению элементов матрицы, а третье число — в какое поле будет помещён график. Таким образом plt.subplot(3, 4, 1) будет означать, что мы имеем три строки полей графиков, четыре столбца, а график будет помещён в первое (т. е. в левое верхнее) поле. Для определения размеров общего поля для всех полей используется функция plt.figure(figsize=()). Аргумент в виде кортежа из двух чисел определяет размер общего поля.

***Отображение нескольких графиков с использованием***

***subplots()***

Функция plt.subplots() позволяет строить несколько графиков на общем поле, при этом каждый график размещается в своём отдельном поле. Функция позволяет разметить общее поле на строки и столбцы из подграфиков. Общий вид обращения к функции выглядит так:

fig, ax = plt.subplots(nrows=2, ncols=3)

Это означает, что общее поле графиков разбито на две строки и три столбца, состоящие из полей подграфиков. Для размещения графика в конкретное поле служит следующая строчка:

ax [0, 2].plot(x, y3)

Здесь график зависимости y3=f(x) будет размещён в верхнюю строку в третий столбец. Для заголовка отдельного графика используется функция

ax [0, 2].set\_title(‘Заголовок’),

а для заголовка общего поля графиков

fig.suptitle(‘Заголовок’)

Иногда может возникнуть необходимость использовать несколькими графиками общей оси, но при этом графики располагаются каждый в своём поле. Допустим двумя графиками совместно используется ось Х. В этом случае поля графиков располагаются друг над другом, а строчки кода запишутся следующим образом

fig, (ax1, ax2) = plt.subplots(2, sharex=True)

ax1.plot(x, y1)

ax2.plot(x, y2)

Здесь параметр sharex установлен в значение True, что означает, что созданные графики будут совместно использовать ось Х.

***3D-графика***

Иногда визуализация данных требует трёхмерной перспективы. И хотя Matplotlib в основном это библиотека для отображения 2D-графики, она обладает инструментами для создания трёхмерных объектов. Для этого необходимо создать трёхмерную систему координат

ax = plt.axes(projection = ‘3d’),

а затем построить график. Это может быть, например, линейный график, график рассеяния и т. д.

***Seaborn***

Со временем matplotlib породила ряд дополнительных наборов инструментов для визуализации данных, которые используют matplotlib в качестве «ядра». Одним из таких инструментов является seaborn.

Seaborn – это библиотека, которая в основном используется для построения статистических грвфиков. Она построена на основе matplotlib и предоставляет красивые стили по умолчанию и цветовые палитры. Это позволяет сделать статистические графики более привлекательными.

***Использование Seaborn совместно с Matplotlib***

Поскольку Seaborn построена на основе Matplotlib, то совместное их использование – процесс естественный. Достаточно вызвать функцию построения графиков seaborn, а затем использовать функции настройки графиков из matplotlib. Если к примеру мы имеем некоторый dataset под названием ‘iris’, то строчка кода

data = sns.load\_dataset(‘iris’)

присвоит переменной data соответствующий dataset, а

sns.lineplot(x=’length’, y=’width’, data=data)

построит статистический график, где по оси Х расположатся данные из столбца length, а по оси Y соответствующие им данные из столбца width нашего dataset. Теперь для нанесения на график наименования его осей, названия собственно графика и других необходимых атрибутов можно использовать соответствующие функции matplotlib, такие как plt.title, plt.xlabel, plt.ylabel и другие.

***Настройка внешнего вида графиков***

Метод set\_style() позволяет корректировать эстетику графика. В seaborn имеется пять стилей: darkgrid, whitegrid, dark, white, ticks. Если к примеру задать стиль dark, то график будет отрисовываться на тёмном фоне.

***Построение нескольких графиков***

Seaborn как и matplotlib предоставляет возможности для построения нескольких графиков, например при помощи метода FacetGrid().

Класс FacetGrid помогает визуализировать распределение одной переменной, а также взаимосвязь между несколькими переменными по отдельности в подмножествах набора данных с использованием нескольких панелей. Этот метод позволяет создавать трёхмерные графики.

Объект FacetGrid принимает в качестве входных данных фрейм данных и имена переменных, которые будут формировать измерения строк, столбцов или оттенков сетки. Строчка кода

plot = sns.FacetGrid(df, col=’region’)

означает, что в качестве исходных данных используется DataFrame, созданный при помощи pandas, общее поле графиков будет размечено под то количество графиков, сколько различных регионов имеется в столбце’ region’. Затем функция

plot.map(plt.scatter, ‘annual\_income’, ‘purchase\_amout’)

построит в данном случае точечные графики зависимости суммы покупок от доходов покупателей для каждого из имеющихся в исходных данных регионов.

***Создание различных типов графиков***

В seaborn как и в matplotlib имеются возможности для создания различных типов графиков. Для создания диаграммы рассеяния используется метод

sns.scatterplot(x=’x’, y=’y’, data=’data’)

Здесь data – исходные данные в виде dataset, x и y – соответствующие столбцы из исходных данных, взаимосвязь которых будет отображена на графике.

Линейные графики создаёт функция

sns.lineplot(x=’x’, y=’y’, data=’data’),

а столбчатые диаграммы

sns.barplot(x=’x’, y=’y’, data=’data’)

Эти и многие другие функции аналогичны соответствующим функциям библиотеки matplotlib.

***Seaborn catplot – визуализация категориальных данных***

Функция catplot() в Seaborn используется для создания графиков на уровне фигур на основе Seaborn FacetGrid. Можно настроить тип визуализации, используя параметр kind=. Если передать в функцию минимальный набор параметров catplot(x=’x’, y=’y’, data=’data’),то будет построена точечная диаграмма. Если указать параметр kind=bar, то диаграмма будет столбчатой. Высоты столбцов будут соответствовать средним значениям из столбца ‘y’.

Чтобы добавить дополнительную переменную в диаграмму следует использовать параметр hue=. Он может разделить столбец на несколько цветов. Например, если в качестве категориальной переменной фигурируют животные, то их можно дополнительно разделить по полу.

Если вместо параметра hue= использовать параметр col= , то произойдёт разделение на два отдельных графика для каждого пола.

В тех случаях, когда количество графиков достаточно велико, используются параметры row= и col=. Тогда, например, в двух строках из графиков будут животные разного пола, а в столбцах – разного вида.

***Изменение меток осей в catplot***

Для изменения меток осей нужно сделать следующее:

cat = sns.catplot()

Теперь функции cat.fig.suptitle(), cat.set\_titles(), cat.set\_xlabel(),cat.set\_ylabel() будут соответственно задавать название всего поля графиков, название отдельных графиков на общем поле, метки осей X и Y.

***Plotly***

Библиотека Plotly в Python — это библиотека с открытым исходным

кодом, которую можно использовать для визуализации данных и

понимания данных просто и легко. Plotly поддерживает различные типы графиков, такие как линейные диаграммы, диаграммы рассеяния,

гистограммы, диаграммы ящиков и т. д. Почему Plotly лучше других инструментов или библиотек визуализации. Вот несколько причин:

* В Plotly предусмотрены инструменты наведения, которые позволяют нам обнаруживать любые выбросы или аномалии в большом количестве точек данных.
* Он визуально привлекателен и может быть принят широким кругом пользователей.
* Plotly, как правило, позволяет нам бесконечно настраивать наши графики и делает наш сюжет более значимым и понятным для

других.

В структуре Plotly имеются два модуля: plotly.express , импортируемый обычно как px, и plotly.graph\_objects, импортируемый как go. Plotly естественно располагает возможностями строить все те типы графиков, что и Matplotlib и Seaborn. Для этого служат соответствующие инструменты. Так

Import plotly.express as px

fig = px.line(x=[1, 2, 3], y=[1, 2, 3])

fig.show()

построит линейный график по точкам в виде прямой линии. Аналогичным образом можно построить столбчатую диаграмму

fig = px.bar(df, x="x ", y="y ")

по параметрам из исходных данных (df) , гистограмму

fig = px.histogram(df, x="x ", y="y ")

диаграмму рассеяния

fig = px.scatter(df, x="x ", y="y ")

В качестве параметров можно задавать размеры точек size=, разбивку по цвету color= и другие.

***Контурные графики***

Контурный график имеет функцию двух переменных кривых, вдоль которых функция имеет постоянные значения, так что эти кривые соединяют точки с равными значениями. То есть если Z = f (X, Y), то на поле с координатной сеткой X, Y наносятся линии , соответствующие равным значениям Z. В качестве примера можно привести карту прогноза погоды, на которую в координатах долгота – широта нанесены изобары, т.е. линии равного атмосферного давления, приведённого к уровню моря, или изотермы – линии равной температуры. Таким образом на двухмерном графике можно проследить за изменениями третьей переменной.

## Заключение

## Matplotlib – библиотека для создания в основном 2D-визуализаций и обладающая очень широкими возможностями.

## Seaborn представляет из себя развитие библиотеки Matplotlib с более лаконичным кодом. Эта библиотека лучше адаптирована для работы со статистическими данными при посредничестве Pandas/

## Plotly – библиотека, обладающая теми же возможностями, что и предыдущие две и имеющая некоторые дополнительные возможности. Визуализации, созданные Plotly, интерактивны и дают возможность пользователю при помощи курсора получать дополнительную информацию, вращать 3D-графики, запускать предварительно созданную мультипликацию.

Визуализация данных служит краеугольным камнем в современном ландшафте интерпретации информации. Ее способность преобразовывать сложные данные в понятные визуальные форматы, такие как диаграммы и графы, играет важную роль в улучшении процессов принятия решений в различных секторах.