Дипломная работа по теме:

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

Автор: Быков Алексей Сергеевич

Оглавление:

1. Введение

Обоснование выбора темы3

1. Основные понятия и определения4
2. Методы сравнения и анализа данных5
3. Обзор популярных инструментов для визуализации данных на Python Matplotlib6

Seaborn8

Plotly 9

1. Проведение анализа графических библиотек9

Линейный график10

Столбчатая диаграмма15

Круговая диаграмма 18

1. Сравнение библиотек 23
2. Заключение 27

Дальнейшие планы 27

1. Источники 28
2. **Введение**

**Обоснование выбора темы**

Качественная визуализация данных имеет критическое значение для анализа данных и принятия решений на их основе. Визуализация позволяет быстро и легко интерпретировать связи и взаимоотношения, а также выявлять развивающиеся тенденции, которые нельзя обнаружить в виде необработанных данных.

Рынок инструментов визуализации данных является важным сегментом многих отраслей, характеризующихся постоянным ростом и инновациями.

Визуализация данных должна быть представлена таким образом, чтобы передавать информацию, обеспечивать реальную ценность и удовлетворять потребности целевой аудитории.

В большинстве случаев для интерпретации графических представлений не требуется специальное обучение, что сокращает вероятность недопонимания.

Человек воспринимает и обрабатывает визуальную информацию быстрее. Исследования показали, что человек способен запомнить более двух тысяч изображений. Интересный факт: мозг человека воспринимает текстовую информацию, как набор картинок.

Визуализация данных находит широкое применение в научных и статистических исследованиях (в частности, в прогнозировании, интеллектуальном анализе данных, бизнес-анализе), в педагогическом дизайне для обучения и тестирования, в новостных сводках и аналитических обзорах.

Самые популярные способы визуализации: графики, mind-карты, диаграммы, презентации, интерактивный и видео формат, иллюстрации. Для каждого типа данных можно подобрать свой способ визуализации – в каком именно представлении информация будет донесена понятней и проще.

Предприятиям в современном мире необходимо обрабатывать большие объемы данных, генерируемых ежедневно. Одной из важнейших задач для большинства компаний является создание различных финансовых отчетов. Расшифровать информацию из отчетов довольно непросто из-за объема информации и необработанных цифр, содержащихся в этих отчетах. Для акционеров и обычных пользователей финансовой отчетности визуализация данных может обеспечить привлекательную и легко понятную историю, преобразуя данные в визуальные элементы, такие как диаграммы или гистограммы, которые помогут им получать полезную информацию и оперативно принимать обоснованные решения.

1. **Основные понятия и определения**

Визуализация (от лат. visualis – зрительный) – создание условий для зрительного наблюдения. В общем смысле – метод представления информации в виде оптического изображения (например, в виде рисунков и фотографий, графиков, диаграмм, структурных схем, таблиц, карт и так далее).

Визуализация данных – изображение информации в виде графиков, которую нужно донести до собеседника или аудитории. Она играет ключевую роль в преобразовании данных в понятную и наглядную форму. Представление информации визуально помогает видеть не только результаты, но и контекст, что способствует более глубокому пониманию и принятию взвешенных решений на их основе. Визуализация данных помогает выделить в наборе данных самое важное, удаляя несущественное и преобразуя информацию в более простой для понимания формат.

Инструменты визуализации данных – это программы, приложения, языки программирования с набором библиотек предназначенные для конкретной цели представления данных в визуальном виде. Каждый инструмент может иметь различные наборы функций, которые могут сделать его лучше для определенных целей, чем другие.

Цели визуализации – это реализация основной идеи информации, это то, ради чего нужно показать выбранные данные, какого эффекта нужно добиться – выявления отношений в информации, показа распределения данных, композиции или сравнения данных.

Библиотека – **это набор функций и инструментов, написанных другими программистами ранее, которые помогают в написании кода и расширяют возможности языка программирования Python.**

Зависимость – это дополнительные программы и пакеты необходимые для корректного функционирования какой-либо библиотеки, которая использует их функции.

Определения терминов из используемого массива данных:

Выручка – это сумма средств, которую компания получает от своей основной деятельности без учета расходов.

Валовая прибыль – это разница между выручкой, полученной от продажи услуги или товара и его себестоимостью. Рассчитывается по формуле: Валовая прибыль = Выручка – Себестоимость товара.

Чистая прибыль – материальные средства, остающиеся в распоряжении предприятия после уплаты всех налогов и обязательных платежей. Используется для увеличения оборотных средств компании, формирования резервов, реинвестиций, выплаты дивидендов.

1. **Методы сравнения и анализа данных**

Одним из подходов к анализу данных является их визуализация с использованием графического представления.

Выделяют следующие типы методов визуализации:

- стандартные 2D/3D-образы – гистограммы, линейные графики и т.п.;

- геометрические преобразования - диаграмма разброса данных, параллельные координаты и т.п.;

- отображение иконок – линейчатые фигуры (needle icons) и звёзды (star icons);

- методы, ориентированные на пикселы – рекурсивные шаблоны, циклические сегменты и т.п.;

- иерархические образы - древовидные карты и наложение измерений.

За основу массива данных будет взята финансовая отчетность Microsoft Corporation за последние 14 лет. В качестве методов визуализации будут использоваться самые распространенные: линейные модели, столбчатые и круговые диаграммы. Оценка будет производиться по следующим критериям: интерактивность, интеграция в Web, типы диаграмм и простота использования.

Программная реализация задачи разбита на три модуля:

* graf\_PLT.py
* graf\_SNS.py
* graf\_PLT.py

Логика работы в файлах Python заключается в том, что сначала происходит чтение (загрузка) данных из файла формата csv с запятой в качестве разделителя. Далее на основе этих данных строятся три варианта графиков, которые в интерпретаторе выводятся либо в дополнительном окне, либо в браузере в зависимости от используемой библиотеки.

1. **Обзор популярных инструментов для визуализации данных на Python**

**Matplotlib** – прародительница инструментов визуализации на Python. Эта библиотека предлагает полный контроль над каждым элементом графика, что делает ее идеальной для создания готовых к публикации графиков и рисунков. Это основа, на которой строятся многие другие библиотеки визуализации. Matplotlib написан и поддерживался в основном Джоном Д. Хантером, и первая общедоступная версия Matplotlib версии 0.1 была выпущена в 2003 году. Matplotlib работает совместно с математической библиотекой NumPy. Она работает как кроссплатформенная (может работать на различных операционных системах, в том числе на Windows, macOS и Linux) библиотека визуализации данных, построенная на массивах NumPy, и предназначена для работы в стеке Scipy. Поддерживается Python 2, Python 3 и IPython. Фактически, Matplotlib была впервые разработана как патч для IPython, для обеспечения интерактивной прорисовки в стиле MATLAB из оболочки IPython. Библиотека позволяет работать с данными на нескольких уровнях: с помощью модуля Pyplot, который рассматривает график как единое целое и через объектно-ориентированный интерфейс, когда каждая фигура или ее часть является отдельным объектом, – это позволяет выборочно менять их свойства и отображение.

Для установки библиотеки достаточно в терминале выполнить команду:

pip install matplotlib

и пакет со всеми зависимостями будет установлен. Также можно зайти в репозиторий: <https://pypi.org> набрать в поиске matplotlib и появится список доступных пакетов.

Согласно документации: <https://matplotlib.org/stable/install/dependencies.html>

для полноценной работы matplotlib устанавливаются следующие пакеты (зависимости):

- contourpy

- cycler

- dateutil

- fontTools

- kiwisolver

- NumPy

- packaging

- Pillow

- pyparsing

- importlib-resources

И если с помощью команды “pip list” вывести список пакетов, то среди них будут и приведенные в списке.

После завершения установки можно использовать Matplotlib в коде Python, импортировав его:

import matplotlib

**Seaborn** – это библиотека для создания статистической инфографики на Python. Она построена поверх Matplotlib, также поддерживает структуры данных NumPy и Pandas. Она также поддерживает статистические единицы из SciPy. Библиотека хорошо передает суть данных с помощью красивых визуализаций, выходящих за рамки базового построения графиков.

Автор Seaborn – Майкл Васком (Филадельфия), сотрудник Центра нейронных исследований (Нью-Йорк), выпускник Стэнфорда. Разработка seaborn это его хобби. Библиотека названа в честь Сэмюэла Нормана Сиборна (S.N.S. – именно поэтому import seaborn is sns), героя сериала Западное крыло (The West Wing), “один из самых важных телесериалов в истории по версии журнала Times”. Таким образом автор отдал дань уважения любимому сериалу.

Для установки библиотеки в терминале необходимо выполнить команду:

pip install seaborn

и пакет со всеми зависимостями будет установлен. Также можно зайти в репозиторий: <https://pypi.org> набрать в поиске Seaborn и появится список доступных пакетов. По умолчанию, устанавливается последняя версия.

Seaborn имеет следующие зависимости:

- NumPy

- SciPy

- pandas

- matplotlib

Для использования библиотеки, достаточно ее импортировать:

import seaborn

**Plotly** – это бесплатная графическая библиотека, позволяющая создавать визуализации данных. Plotly – это веб-приложение основанное на JavaScript-библиотеке Plotly. Состоит из нескольких частей: Front-End на JS, Back-End на Python (за основу взята библиотека Seaborn), Back-End на R. Plotly поддерживает веб-графики, 3D и анимацию. Обеспечивает высокую степень настройки и интерактивности, а также поддерживает несколько языков, всплывающие подсказки, увеличение и уменьшение масштаба, выбор рамкой, загрузку в формате png и сравнение данных при наведении курсора.

Plotly – техническая вычислительная компания со штаб-квартирой в Монреале (Квебек), которая разрабатывает онлайн-инструменты для анализа данных и визуализации. Компания Plotly была основана Алексом Джонсоном, Джеком Пармером, Крисом Пармером и Мэтью Сандквистом. Канадская биржа инноваций назвала Plotly одной из 20 самых популярных инновационных компаний Канады.

Установить библиотеку также просто, в терминале необходимо выполнить команду:

pip install plotly

Или можно зайти в репозиторий: <https://pypi.org>, набрать в поиске plotly и появится список доступных пакетов.

Для использования библиотеки выполняется процедура импорта:

import Plotly

1. **Проведение анализа графических библиотек**

Для проведения анализа необходимо в интерпретаторе Python создать новый проект и установить требуемые графические библиотеки с помощью командной строки в терминале или через интерфейс в настройках программы. После создаем модули с краткими названиями библиотек (для удобства: graf\_PLT.py, graf\_SNS.py, graf\_PX.py) в которых будут проводиться вычисления и построение графиков. Также необходимо файл с расширением csv с массивом разместить в пакете проекта. Файл MSFT.csv содержит в себе данные отдельных параметров финансовой отчетности Microsoft Corporation с 2011 по 2014 год и состоит из следующих полей:

* год (year)
* выручка (earnings)
* чистая прибыль (revenue)
* валовая прибыль (gross\_profit)

Код будет начинаться с импорта как самих библиотек, так и вспомогательных инструментов, например, Pandas для анализа данных с последующей их визуализацией. Далее загружаем данные из файла формата csv с запятой в качестве разделителя:

df = pd.read\_csv('MSFT.csv', sep=',')

Для облегчения оперирования данными в коде сохраняем их в переменные:

a = df['year'] # Год  
b = df['earnings'] # Выручка  
c = df['revenue'] # Чистая прибыль  
d = df['gross\_profit'] # Валовая прибыль

Теперь приступаем к созданию графиков.

**Линейный график** (Line plot) – это один из наиболее простых типов графиков, который используется для отображения зависимости одной переменной от другой на плоскости. Линейный график может быть полезным инструментом для визуализации и выявления тенденций, а также для выявления аномалий в данных. Он легко читается и может использоваться как для быстрой визуализации данных, так и для более глубокого анализа. График представляет собой ломаную линию, которая проходит через точки, которые соответствуют значениям переменных. График используется для представления взаимосвязи между двумя данными X и Y на разных осях. Линейный график может быть построен как для непрерывных, так и для дискретных значений переменных. Этот тип графика может использоваться для отображения изменений величин со временем или для отображения корреляций между переменными.

Примеры использования линейного графика:

- отображение изменения цен на бирже за определенный период времени;

- отображение изменения климатических параметров со временем;

- отображение зависимости доходности от времени.

**Matplotlib**

Для начала зададим размер область (поле) на котором будет размещаться график

plt.figure(figsize=(10, 8))

Сообщим его заголовок и немного увеличим шрифт

plt.title('Финансовые показатели Microsoft Corporation с 2011-2024гг.', fontsize=16)

Дадим название осям Х и Y и поменяем цвет их имени

plt.xlabel('Год', color='gray')  
plt.ylabel('Млрд. $', color='gray')

С помощью**функции plot()** создадим график с указанием переменных, в которые ранее сохранили числовые данные, с заданием разных цветов линий, маркеров, их размеров и типа отрисовки одной кривой как варианта

plt.plot(a, b, color='green', marker='^', markersize=6)  
plt.plot(a, c, 'r--', linewidth=3, marker='o', markersize=6)  
plt.plot(a, d, color='blue', marker='\*', markersize=7)

Немаловажная часть любого графика – легенда. Вносим название отражаемых данных списком из неопределенного числа аргументов \*args с указанием местоположения легенды и ее названием

plt.legend(['Чистая прибыль', 'Выручка', 'Валовая прибыль'], loc=2, title='Условные обозначения:')

Добавим отображение сетки на графике без излишеств, по умолчанию

plt.grid()

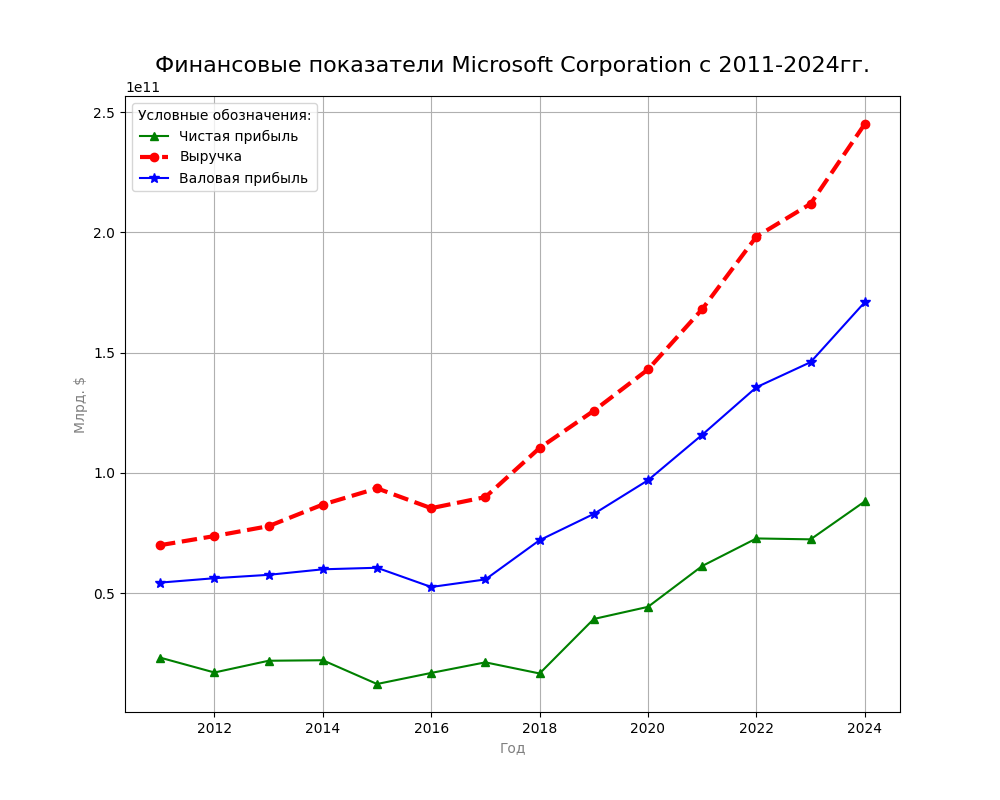
Командой savefig определим сохранение будущего изображения с указанием имени и расширения файла

plt.savefig('linear\_PLT.png')

Последняя важная команда вызова изображения

plt.show()

Итог:

**Seaborn** 

**Поскольку библиотека является расширением Matplotlib и создана для упрощения, о чем упоминалось выше, команды библиотеки могут быть теми же, и чем детальнее и сложнее нужен график, тем больше команд из Matplotlib.**

**Поэтому** размер области графика повторяем

plt.figure(figsize=(10, 8))

**Применим команду заливки фона (стиль Seaborn)**

sns.set\_style('darkgrid')

**Воспользуемся встроенной функцией** **Seaborn** для построения линейного графика lineplot() и зададим параметры: укажем переменные с данными, цвет линий, тип маркера и его размер, и наименование выводимых данных для отражения их в легенде (при применении легенды **Matplotlib возник конфликт с некорректным ее отображением)**.

sb = sns.lineplot(x=a, y=b, color='green', marker='^', markersize=7, label='Чистая прибыль')  
sb1 = sns.lineplot(x=a, y=c, color='blue', marker='d', markersize=7, label='Выручка')  
sb2 = sns.lineplot(x=a, y=d, color='red', marker='h', markersize=7, label='Валовая прибыль')

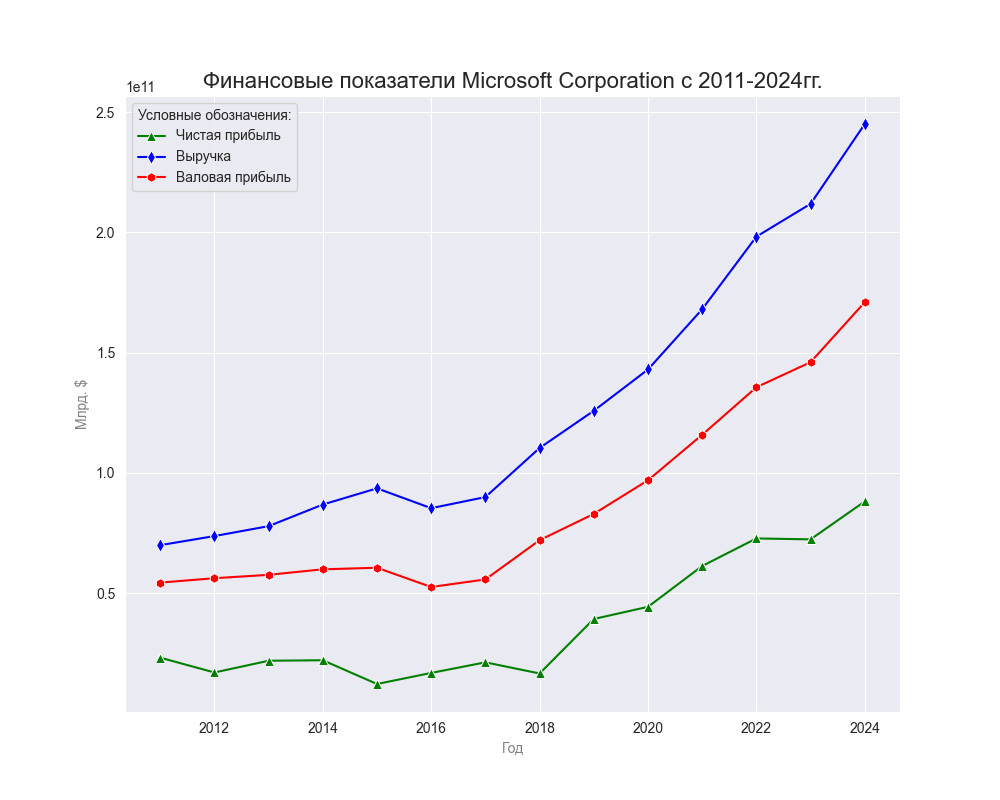
Сообщим заголовок графика и немного увеличим шрифт

sb.set\_title('Финансовые показатели Microsoft Corporation с 2011-2024гг.', fontsize=16)

**И командами из Matplotlib проработаем оси, название легенды, сохранение изображения и команду вызова**

plt.xlabel('Год', color='gray')  
plt.ylabel('Млрд. $', color='gray')  
plt.legend(title='Условные обозначения:')  
plt.savefig('linear\_SNS.png')  
plt.show()

**Итог:**



**Plotly**

**Линейный, впрочем любой, график в Plotly создается командой go.Figure(), если мы используем Plotly Go**

fig = go.Figure()

**Затем начинаем эту фигуру рисовать (задаем параметры, в которых хранятся данные, устанавливаем тип отображения линий с маркерами и указываем имена данных, которые появятся во всплывающем окне при наведении курсора на маркер)**

**Указываем название фигуры (обновляем расположение)**

fig.update\_layout(title='Финансовые показатели Microsoft Corporation с 2011-2024гг.')

**Задаем названия осей, размер шрифта заголовка, его центровку, ориентацию легенды и размер ее шрифта**

fig.update\_layout(yaxis\_title='Млрд. $', xaxis\_title='Год', title={'font':

dict(size=50), 'x': 0.5}, legend\_orientation="h",

legend\_font\_size=25)

**Определяем параметры осей Х и Y (размер шрифта названия осей, и размер шрифта меток на осях). Также проведем линии основных меток осей и промежуточных для лучшего визуального восприятия графика.**

fig.update\_xaxes(gridcolor='black', minor\_griddash='dot',

titlefont=dict(size=22), tickfont=dict(size=19))  
fig.update\_yaxes(titlefont=dict(size=22), tickfont=dict(size=19))

**Оформляем всплывающую подсказку (какие атрибуты будут в ней содержаться и их шаблон)**

fig.update\_traces(hoverinfo='all', hovertemplate='Год: %{x}<br>Объем: %{y}')

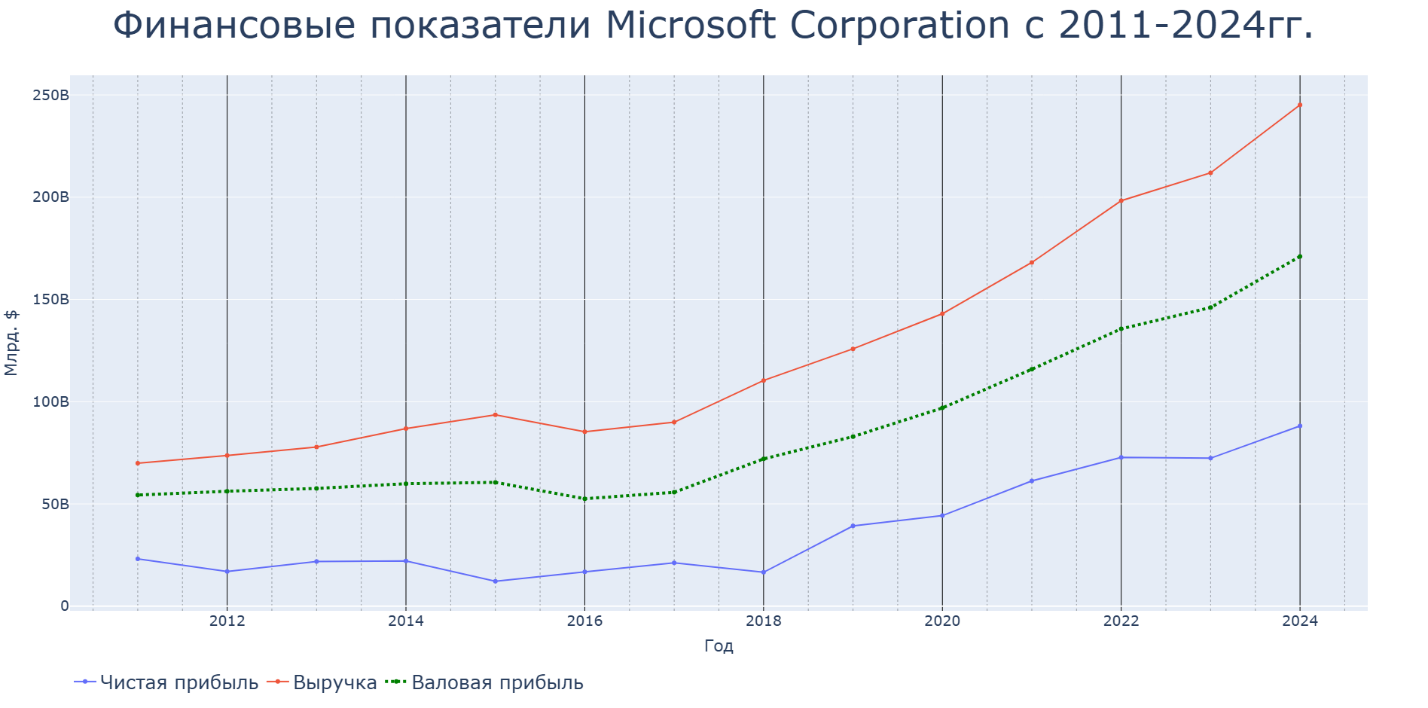
**Затем сохраняем график**

fig.write\_image('linear\_PX.png')

**И выводим его**

fig.show()

**Итог:**

**Проводя промежуточный итог по линейным графикам можно сказать, что с применением простых эффектов справляются все библиотеки. При этом объем кода более “плотнее” в Plotly, но здесь применяем полновесный низкоуровневый Plotly Go, а не Express. Можно оставить преимущество за Plotly как более информативной библиотеки за счет ее интерактивности.**

**Столбчатая диаграмма** (Bar plot) **– это график, представляющий категорию данных прямоугольными столбцами с длиной и высотой, пропорциональными значениям, которые они представляют.** Столбчатые диаграммы могут быть нанесены горизонтально или вертикально. Используя определенным образом подготовленные данные можно строить групповые диаграммы.

**Matplotlib**

Столбчатая диаграмма описывает сравнения между отдельными категориями. Ее можно создать с помощью метода bar(). Эта функция принимает на вход основные численные данные, а также другие параметры, такие как цвета, размеры, метки, ориентацию. Начало построения диаграммы аналогично линейному графику: задание области построения, название диаграммы, именование осей

plt.figure(figsize=(10, 8))  
plt.title('Финансовые показатели Microsoft Corporation с 2011-2024гг.', fontsize=16)  
plt.xlabel('Год', color='gray')  
plt.ylabel('Млрд. $', color='gray')

Далее непосредственно рисование: указание переменных и задание цвета, которое можно сократить до одной первой буквы основной палитры

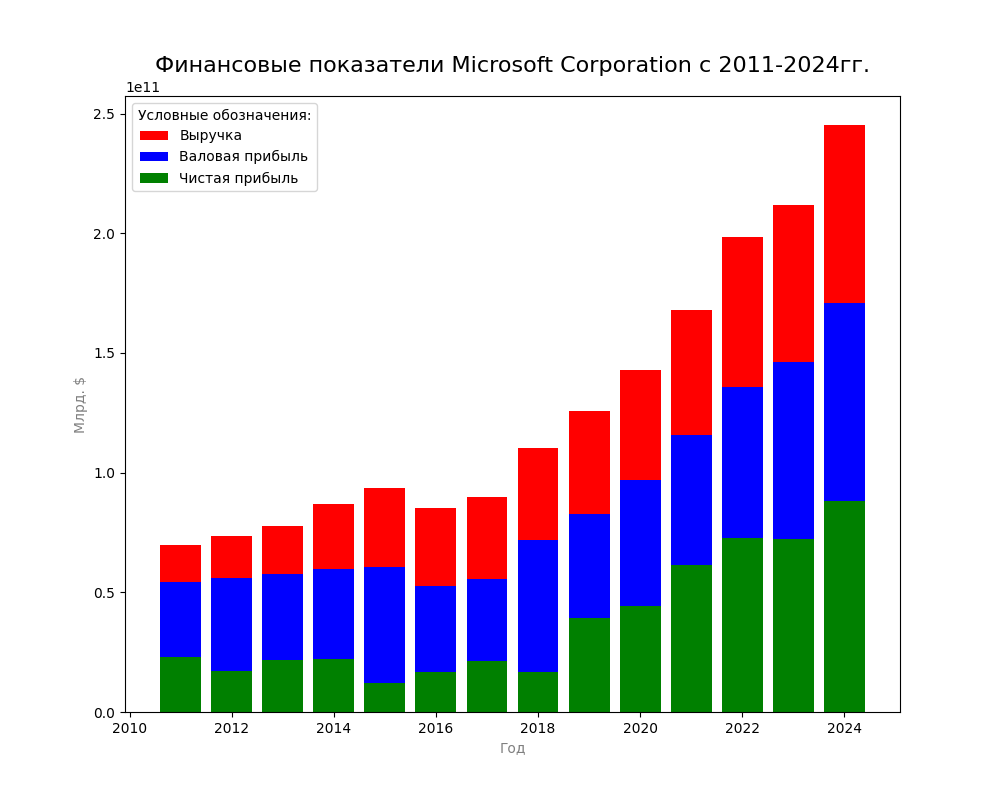
plt.bar(a, c, color='r')  
plt.bar(a, d, color='b')  
plt.bar(a, b, color='g')

Вызов легенды, функция сохранения и показа графика остаются такими же

plt.legend(['Выручка', 'Валовая прибыль', 'Чистая прибыль'], loc=2,

title='Условные обозначения:')  
plt.savefig('bar\_PLT.png')  
plt.show()

Итог:



**Seaborn**

График строится с помощью функции barplot(), из модуля seaborn, за настройку вида отвечает функция set() (добавить заголовок, настроить стиль, цвет, фон, шрифт и его размер, отрисовку сетки).

Начинаем с указания размера области построения графика

plt.figure(figsize=(14, 8))

Установим стиль немного расширив параметры в отличие от линейного графика. Установим метки на оси, зададим контекст позволяющий изменить размер и масштаб элементов графика в зависимости от его предполагаемого использования и размер шрифта меток осей

sns.set(style='ticks', context='paper', font\_scale=1.5)

Отрисуем задав данные, цвет столбцов и имена для легенды (при применении легенды **Matplotlib возник конфликт с цветовой окраской)**.

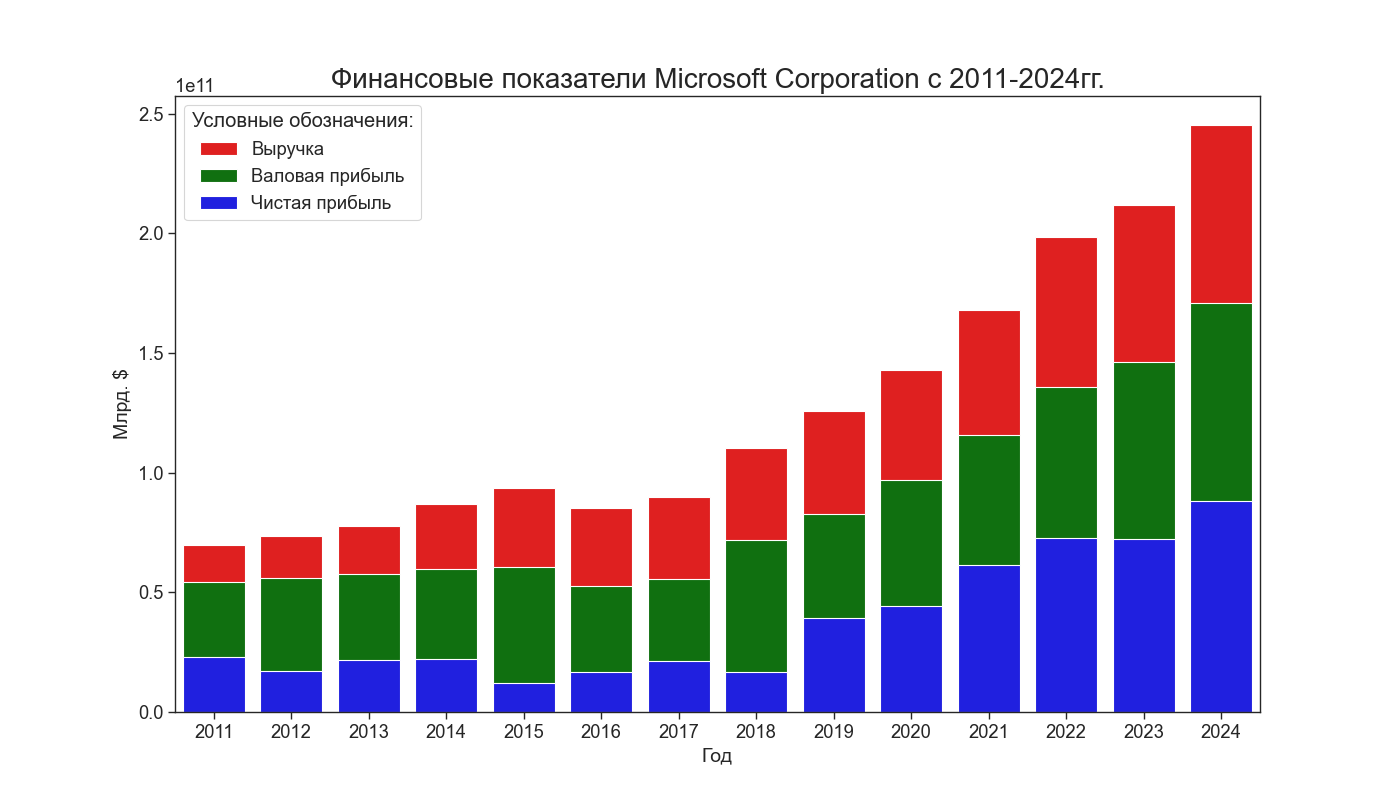
sbn = sns.barplot(x=a, y=c, data=df, color='red', label='Выручка')  
sbn1 = sns.barplot(x=a, y=d, data=df, color='green', label='Валовая прибыль')  
sbn2 = sns.barplot(x=a, y=b, data=df, color='blue', label='Чистая прибыль')

Зададим название графика и осей с выбором размера шрифта функционалом seaborn

sbn.set\_title('Финансовые показатели Microsoft Corporation с 2011-2024гг.', fontsize=20)  
sbn.set\_xlabel('Год', fontsize=14)  
sbn.set\_ylabel('Млрд. $', fontsize=14)

Название легенды, сохранение графика и его вызов

plt.legend(title='Условные обозначения:')  
plt.savefig('bar\_SNS.png')  
plt.show()

Итог:

**Plotly**

Воспользуемся функцией для построения столбчатой диаграммы bar(). Поскольку имеет не один набор данных, всё оформляется списком данных с их именами для отражения в легенде

fig = go.Figure(data=[  
 go.Bar(x=a, y=c, name='Выручка'),  
 go.Bar(x=a, y=d, name='Валовая прибыль'),  
 go.Bar(x=a, y=b, name='Чистая прибыль'),  
])

**Задаем вид построения столбцов название графика, название осей, размер шрифта заголовка, его центровку, ориентацию легенды и размер ее шрифта. Также выставляем размеры шрифта названия осей и меток.**

fig.update\_layout(barmode='overlay', title='Финансовые показатели Microsoft

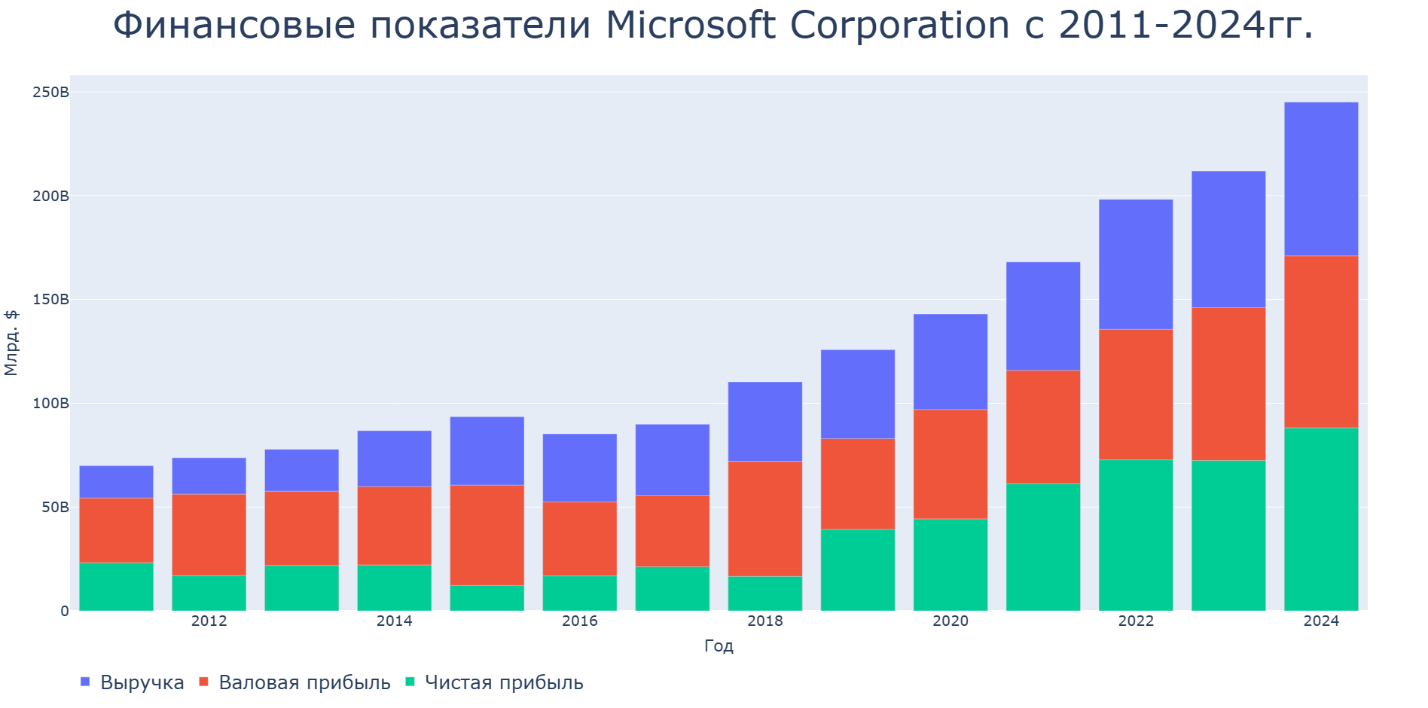
Corporation с 2011-2024гг.')  
fig.update\_layout(yaxis\_title='Млрд. $', xaxis\_title='Год', title={'font':

dict(size=50), 'x': 0.5}, legend\_orientation='h',

legend\_font\_size=25)  
fig.update\_xaxes(titlefont=dict(size=22), tickfont=dict(size=19))  
fig.update\_yaxes(titlefont=dict(size=22), tickfont=dict(size=19))

И оформляем всплывающую подсказку, сохраняем изображение, выводим график

fig.update\_traces(hoverinfo='all', hovertemplate='Год: %{x}<br>Объем: %{y}')  
fig.write\_image('bar\_PX.png')  
fig.show()

Итог:

**Проводя промежуточный итог по столбчатым диаграммам можно сказать, что графики Matplotlib и Seaborn выглядят более красочно. По размеру кода на одну строку впереди Matplotlib. За Plotly остается ее интерактивность.**

**Круговая диаграмма** (piechart) – это графическое представление данных, которое используется для показа соотношения отдельных частей к общей сумме. Она хорошо подходит для визуализации категориальных данных или процентного соотношения. Круговые диаграммы применяются для визуализации различных долей или процентных соотношений внутри одного целого. Примерами могут служить доли доходов от разных источников, расходы на различные категории товаров и услуг и т.д. Круговая диаграмма удобна для визуализации относительных долей и частот в категориях данных. Она позволяет быстро и наглядно показать соотношение всех категорий, а также их вклад в общую сумму данных.

Круговая диаграмма состоит из нескольких ключевых элементов, которые помогают сделать данные более понятными и наглядными.

Основные элементы круговой диаграммы:

Круг – основная форма диаграммы, представляющая 100% данных. Круг является базовой структурой, внутри которой располагаются все остальные элементы.

Секторы – части круга, каждая из которых соответствует определенной категории данных. Размер каждого сектора пропорционален значению категории, которую он представляет.

Цвета – различные цвета используются для визуального различения секторов. Это помогает легко различать категории данных.

Легенда – объясняет, какие данные представлены каждым сектором. Легенда обычно располагается рядом с диаграммой и содержит описание категорий и их соответствующих цветов.

**Matplotlib**

В библиотеке matplotlib для создания круговых диаграмм используется функция pie(). Эта функция принимает на вход данные, которые нужно отобразить в виде круговой диаграммы, а также другие параметры, такие как цвета, размеры и метки.

Опять же все начинается с создания области построения

plt.figure(figsize=(10, 10))

Отрисовываем диаграмму задав срез данных массива с их суммированием по категориям. В параметрах задаем: указать на секторах процентное соотношение, уберем излишние подписи секторов, укажем размеры отделения одной из долей диаграммы в связи с применением данного эффекта

df[df.columns[1:4]].sum().plot.pie(autopct='%1.1f%%', labels=None,

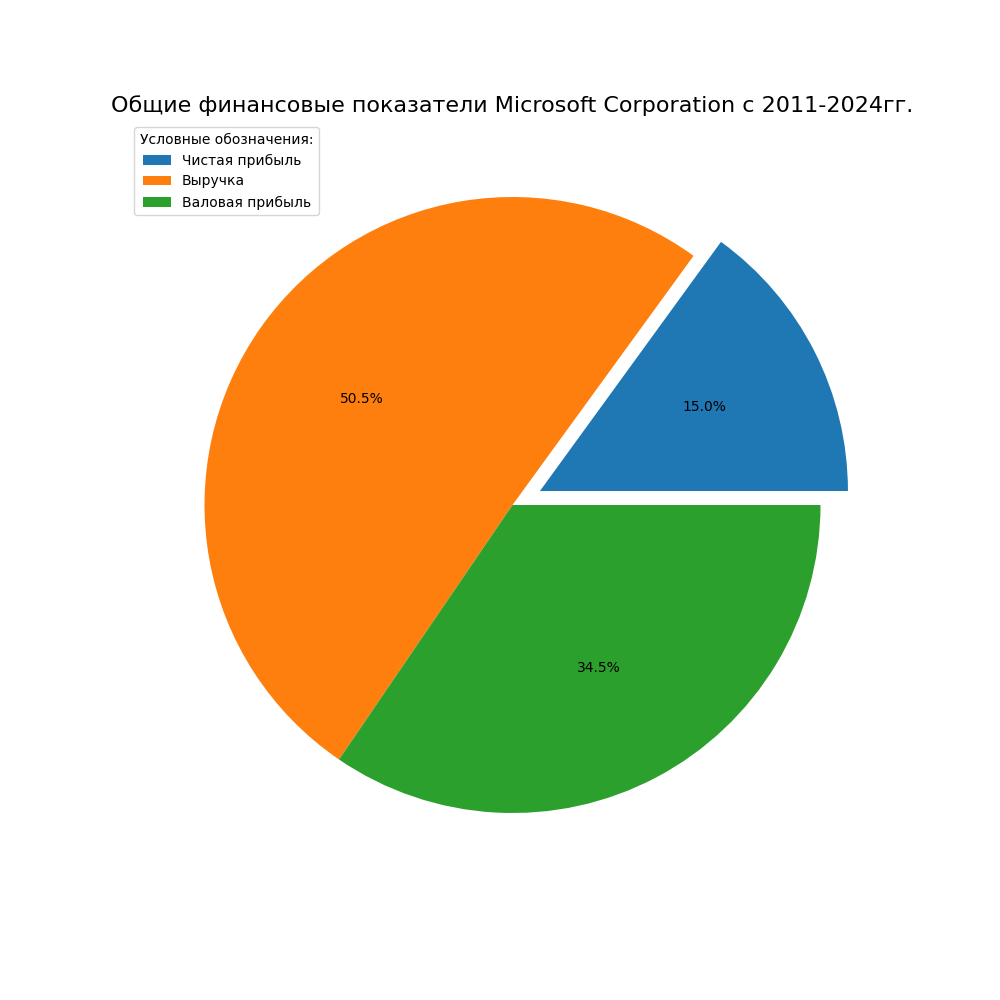
explode=[0.1, 0, 0])

Укажем имена для легенды, местоположение и ее название. Имя графика и размер его шрифта. Сохраним изображение и выведем на экран.

plt.legend(['Чистая прибыль', 'Выручка', 'Валовая прибыль'], loc=2,

title='Условные обозначения:')  
plt.title('Общие финансовые показатели Microsoft Corporation с 2011-2024гг.',

fontsize=16)  
plt.savefig('circular\_PLT.png')  
plt.show()

Итог:

**Seaborn**

В библиотеке визуализации данных Python Seaborn нет функции по умолчанию для создания круговых диаграмм, поэтому используется синтаксис Matplotlib для создания круговой диаграммы, но с добавлением цветовой палитры Seaborn.

Определим область для изображения и создадим переменную для сохранения параметров цвета из функционала Seaborn

plt.figure(figsize=(10, 10))  
colors = sns.color\_palette('deep')[0:3]

Запишем функцию создания круговой диаграммы указав срез данных из массива. В параметрах укажем вывод процентов на диаграмме, уберем подписи секторов, применим ранее выбранную цветовую палитру и добавим тень

df[df.columns[1:4]].sum().plot.pie(autopct='%1.1f%%', labels=None,

colors=colors, explode=[0.2, 0.1, 0],

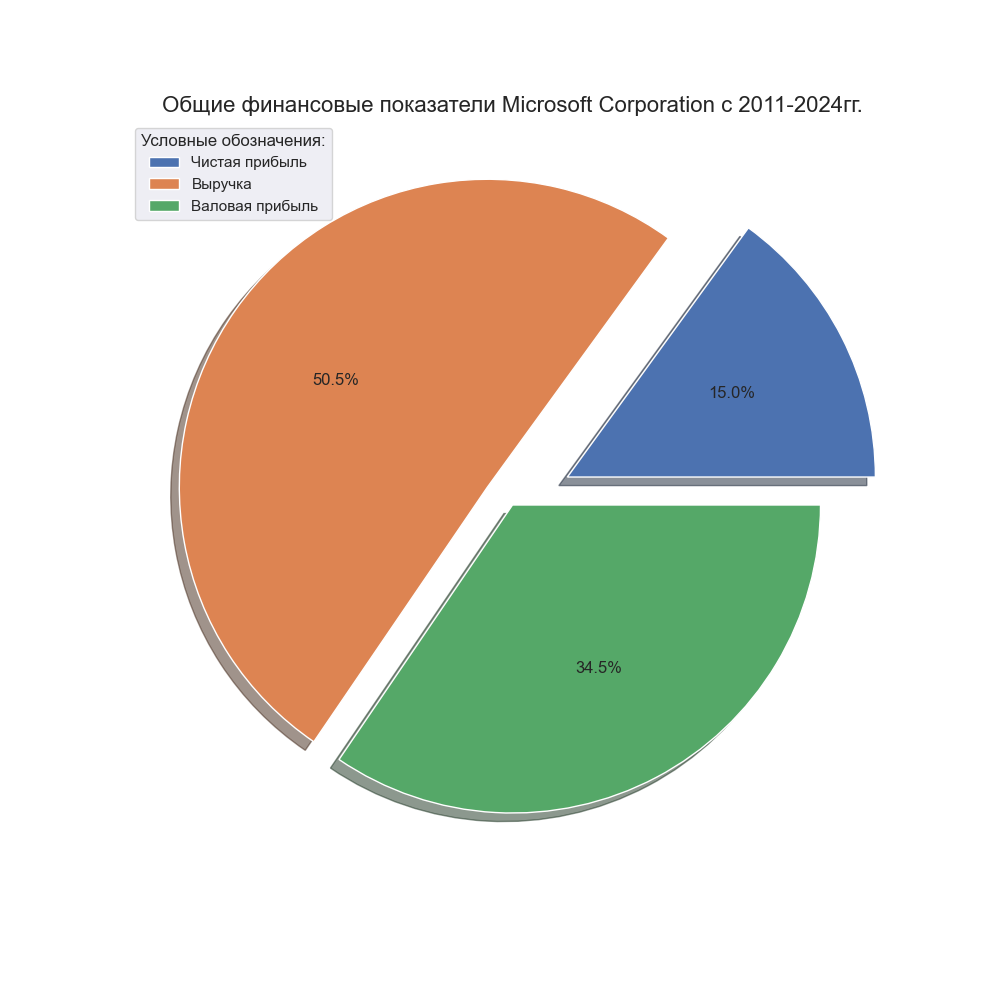
shadow=True)

Также далее указываем легенду с параметрами, заголовок графика с увеличенным шрифтом, сохраняем и выводим изображение

plt.legend(['Чистая прибыль', 'Выручка', 'Валовая прибыль'], title='Условные

обозначения:', loc=2)  
plt.title('Общие финансовые показатели Microsoft Corporation с 2011-2024гг.',

fontsize=16)  
plt.savefig('circular\_SNS.png')  
plt.show()

Итог:

**Plotly**

Создадим переменную, куда сохраним сумму по каждой категории данных массива

data = [df['earnings'].sum(), df['revenue'].sum(), df['gross\_profit'].sum()]

Переменную с именованием категорий для легенды

labels = ['Чистая прибыль', 'Выручка', 'Валовая прибыль']

В функции создания круговой диаграммы укажем параметры с данными, имена категорий для легенды и заголовок графика

fig = px.pie(df, values=data, names=labels, title='Общие финансовые

показатели Microsoft Corporation с 2011-2024гг.')

Изменим размер шрифта заголовка и расположим название по центру, также зададим имя легенде и увеличим ее шрифт, а также увеличим шрифт подписей на диаграмме

fig.update\_layout(title={'font': dict(size=25), 'x': 0.5},

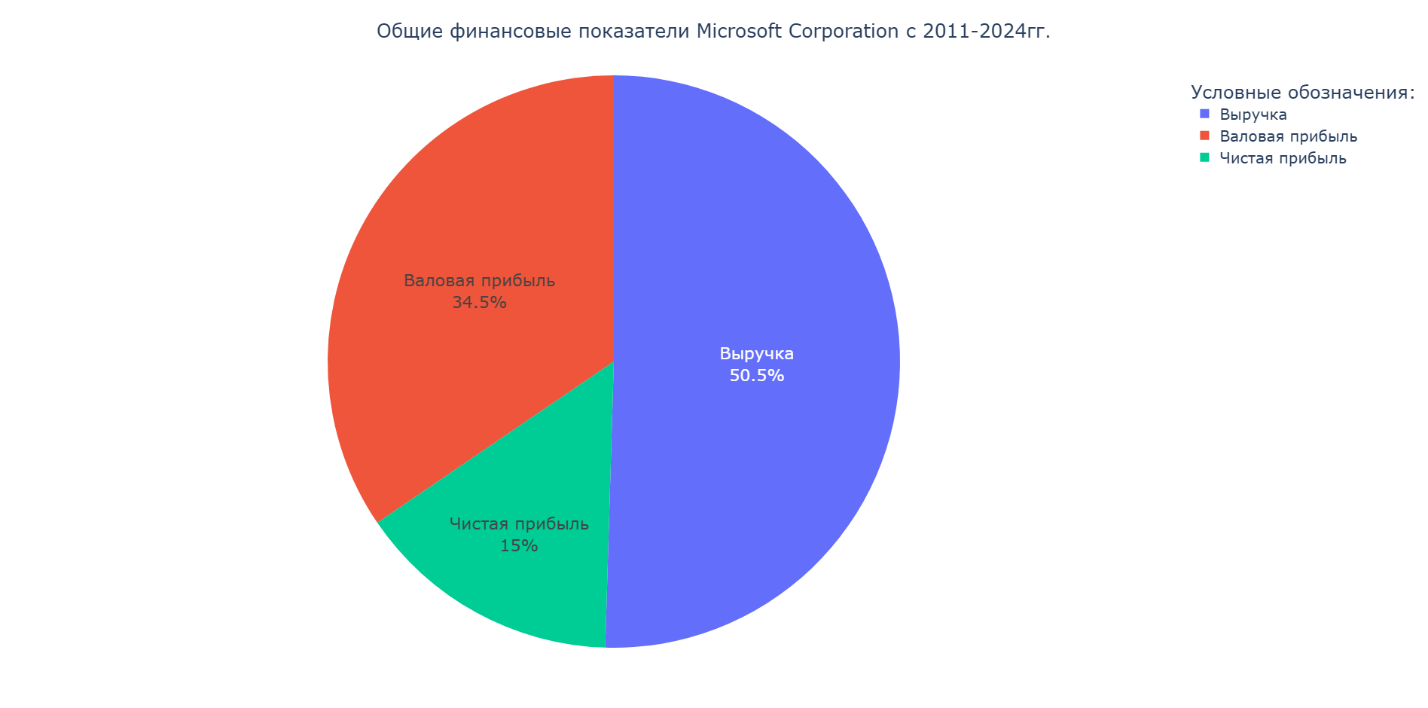
legend\_title='Условные обозначения:',  
 legend=dict(font=dict(size=20)), font={'size': 18})

Как вариант можно добавить подписи, помимо легенды, непосредственно на диаграмму

fig.update\_traces(textposition='inside', textinfo='percent+label')

Сохраним изображение и дадим команду на вывод графика

fig.write\_image('circular\_PX.png')  
fig.show()

Итог:

**Проводя промежуточный итог по круговым диаграммам можно сказать, что все библиотеки имеют достаточный набор инструментов для создания простого кругового графика. В приведенных примерах отражено разнообразие применяемых эффектов, но в каждой библиотеке есть возможность отделить сектора и добавить тень и много чего другого. Объем кода, учитывая эффекты, можно сказать одинаков и здесь все библиотеки справились со своей задачей.**

1. **Сравнение библиотек**

Matplotlib – популярный пакет для построения графиков, который постоянно развивается. Он предлагает множество интерфейсов для рендеринга и использует подробный синтаксис, что обеспечивает высокую степень гибкости и настраиваемости графиков. Известно, что Matplotlib, которая является полностью графической библиотекой, основана на NumPy. Поэтому возможно передавать в ней списки в качестве аргументов. Это нужно как для представления данных, так и для того, чтобы задавать границы осей. Внутри эти списки конвертируются в [массивы NumPy](https://pythonru.com/biblioteki/strukturirovanie-massivov-numpy). Таким образом можно прямо добавлять в качестве входящих данных массивы NumPy. Массив данных, обработанный Pandas, может быть использован Matplotlib без дальнейшей обработки. В документации Matplotlib имеется описание каждого параметра, что очень упрощает реализацию. Можно подтвердить, что Matplotlib не зря хвалят за ее универсальность и способность построить график из чего угодно с помощью достаточного количества команд. Получаемые изображения, к примеру, могут быть использованы в качестве иллюстраций в публикациях.

Переходя к seaborn, можно сказать, если Matplotlib упрощает и делает сложные вещи возможными, то Seaborn пытается упростить и эту сложную задачу. Но Seaborn не является альтернативой Matplotlib. Построенная на основе библиотеки Matplotlib, Seaborn расширяет ее функциональность, облегчая создание сложных визуализаций. Это позволяет использовать лаконичный, но ограниченный подход к быстрой визуализации наборов данных с использованием более привлекательных стилей по умолчанию, чем Matplotlib. Поскольку Seaborn построен поверх Matplotlib, он также поддерживает структуры данных NumPy и Pandas. Она особенно хорошо подходит для статистического анализа, так как предоставляет высокоуровневый интерфейс для рисования информативных статистических графиков. Любая диаграмма Seaborn может быть изменена с помощью функций из библиотеки Matplotlib. Эта механика может пригодиться в определенных случаях и позволяет Seaborn использовать возможности Matplotlib без необходимости переписывать все ее функции.

Иными словами, главное отличие между Seaborn и Matplotlib заключается в уровне сложности и удобстве. Matplotlib предоставляет более низкоуровневый доступ к созданию графиков и требует больше кода для настройки. С другой стороны, Seaborn делает процесс создания графиков более простым и позволяет создавать красивые графики с минимальными усилиями. Seaborn – это библиотека визуализации на Python с открытым исходным кодом. Seaborn не имеет встроенных возможностей 3D и анимации без использования Matplotlib. Используется Seaborn, например, когда нужно сделать быстро и красиво, но когда необходимо больше детализации и проработки, то используется Matplotlib.

Plotly является более информативной и удобной библиотекой для визуализации данных. Библиотека имеет “минимальный” код для красочной визуализации данных, интерактивные элементы для выделения и исследования данных, возможность существенной детализации отображаемой информации. У библиотеки есть оболочка Plotly Express, которая представляет собой интерфейс более высокого уровня для Plotly. Plotly Express быстр и прост в использовании в качестве отправной точки для создания наиболее распространенных диаграмм с простым синтаксисом, но ему не хватает функциональности и гибкости, когда речь заходит о более сложных типах диаграмм или настройках. В отличие от Plotly Express, Plotly Go (Graph Objects) – это графический пакет более низкого уровня, который, как правило, требует больше кода, но гораздо более настраиваемый и гибкий.

По сравнению с Matplotlib и Seaborn, Plotly предлагает обширный список вариантов построения графиков, начиная с обычных и заканчивая анимационными или графическими картами. В Plotly реализованы многие базовые графики, в частности, линии, гистограммы и барчарты, боксплоты, тепловые карты, геокарты, а также некоторые 3d-графики. Создавать собственные типы графиков средствами Plotly, к сожалению, нельзя. У Plotly есть функционал выделения области разными вариантами, удобен просмотр расположения маркера, а также есть возможность посмотреть конкретное значение маркера.

Plotly подходит, если используются такие языки, как Python, R, MATLAB, Perl и Julia, или если нужны интерактивные или анимированные веб-графики. Легкая интеграция библиотеки с веб-технологиями позволяет встраивать визуализации в веб-страницы, повышая интерактивность и доступность данных. К сожалению, у анимационных графиков нет возможности сохранения в gif, что очень неудобно. Диаграммы не сохраняются как изображения, а сериализуются как JSON, что делает их открытыми для чтения с помощью R, MATLAB, Julia и других языков. Графики могут быть использованы для печати и публикации. Также можно отметить, что у Plotly очень удобная и понятная документация, можно легко найти какой вариант визуализации нужен для работы. Plotly мощный инструмент, но настройка и создание графиков в нем занимают много времени, и эта настройка не является интуитивно понятной.

Таблица сравнения библиотек по критериям:

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Matplotlib | Seaborn | Plotly |
| Интерактивность | По умолчанию предоставляет статические визуализации и требует дополнительного кода для интерактивности. Для создания простых графиков обычно требует меньшего объема кода | Сам Seaborn не предоставляет встроенных интерактивных функций, его можно комбинировать с другими библиотеками или инструментами для достижения интерактивности | Известен своими интерактивными возможностями, позволяющими пользователям наводить курсор на точки данных, увеличивать и уменьшать масштаб и взаимодействовать с графиками |
| Интеграция в Web | Представляет собой в первую очередь библиотеку для локального создания графиков и не предоставляет встроенных функций для Web | Как и Matplotlib, библиотека для локального создания графиков без встроенных функций для Web | Предлагает онлайн-платформу Plotly Chart Studio, где пользователи могут создавать, редактировать и делиться графиками. Это позволяет легко делиться визуализациями с другими |
| Типы диаграмм | Больше ориентирован на 2d-графики, хотя и предлагает некоторые ограниченные 3d-возможности | Имеет множество графиков разного типа из коллекции для визуализации распределений, корреляций, трендов и других результатов статистического анализа. Возможностей 3d нет без использования Matpoltlib | Предоставляет более широкий выбор типов диаграмм. Plotly поддерживает 3d-графики, контурные графики, свечные диаграммы и многое другое |
| Простота использования | Считается основной библиотекой для базового построения графиков в Python и существует уже давно. Он имеет большую базу пользователей и обширную документацию, что значительно облегчает начало работы новичкам | Благодаря удобному API Seaborn позволяет создавать сложные графики с минимальным количеством кода | Хотя и удобен для пользователя, может потребовать более сложного обучения из-за расширенных функций и синтаксиса |

1. **Заключение**

 Все 3 представленные в работе библиотеки предназначены для визуализации больших массивов данных, имеют готовые шаблоны визуализации и довольно широкий набор возможностей. Matplotlib и Seaborn выводят статичные графики без возможности интерактивного взаимодействия с ними. Вывод графиков происходит в дополнительном окне, визуализации появляются последовательно друг за другом после закрытия предыдущей, так как код для данной работы был написан в одном модуле для каждой библиотеки.

Matplotlib имеет гибкие возможности для настройки отображения графиков, осей и подписей. В Seaborn нет возможности вывести дополнительно оси и настроить их, для этого в программный код интегрируют вызов библиотеки Matplotlib. К отличительным особенностям Plotly можно отнести вывод результатов работы программы в отдельные страницы браузера с возможностью смотреть их в необходимом порядке, переключаться произвольно между вкладками и интерактивность самих графиков и диаграмм (при наведении курсора видны всплывающие подсказки). Таким образом можно исследовать детальнее график, приближать, отдалять, выделять области, сохранять изображение.

**Дальнейшие планы**

1. Изучить существующие графические библиотеки для Python.
2. Разработать новые инструменты и расширить функционал существующих библиотек.
3. Разработать дашборд с графическими библиотеками и возможностью выбора нужных инструментов для обработки данных и построения графиков.
4. Провести тестирование разработанных инструментов.
5. Создать обучающий материал по использованию новых графических возможностей Python.
6. **Источники**

[www.matplotlib.org](http://www.matplotlib.org)

<https://pythonim.ru/libraries/biblioteka-matplotlib-v-python>

[www.seaborn.pydata.org](http://www.seaborn.pydata.org)

<https://habr.com/ru/companies/otus/articles/540526>

[www.plotly.com](http://www.plotly.com)

<https://python-school.ru/blog/визуализация-данных/plotly-basics>

<https://marketcap.ru/stocks/MSFT/financial-statements/income-statement>

Абдрахманов М. И. Библиотека Matplotlib

Коритес Б. Графика на Python

Python. Визуализация данных