УДК 004.423 **Информационные технологии**

Халевин Тимофей Анатольевич, *студент направления подготовки информатики и вычислительной техники, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

Голубничий Артем Александрович, *научный руководитель, старший преподаватель кафедры ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

ВИЗУАЛИЗАЦИЯ ДАННЫХ В PYTHON С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ MATPLOTLIB: ОСНОВЫ И ПРИМЕРЫ

Аннотация: Данная статья представляет обзор основ визуализации данных с использованием библиотеки Matplotlib на языке программирования Python.

Ключевые слова: Python, визуализация, данные, matplotlib.

Annotation: This article provides an overview of the basics of data visualization using the Matplotlib library in the Python programming language.

Keywords: Python, visualisation, data, matplotlib.

В наше информационное время объем данных, генерируемых и собираемых, постоянно растет. Стремительное увеличение объема данных создает необходимость в их эффективной обработке и анализе. Визуализация данных является ключевым инструментом в этом процессе, позволяя визуально воспринимать сложные структуры и зависимости в данных, что особенно важно в контексте принятия обоснованных решений на основе данных.

Python, ставший одним из наиболее популярных языков программирования, заслужил свою репутацию в области анализа данных. Его простота, гибкость и богатые библиотеки делают его предпочтительным

выбором для специалистов в области данных. Важно отметить, что Python обладает богатым набором инструментов для визуализации, и в этом контексте, библиотека matplotlib играет ведущую роль [1].

Matplotlib – это мощная библиотека для создания разнообразных графиков и визуализаций в Python. Она предоставляет широкий спектр возможностей для создания статических, анимированных и интерактивных графиков. В этой статье мы исследуем основные принципы работы библиотеки, рассмотрим различные типы графиков, и рассмотрим примеры визуализации данных с использованием matplotlib [2].

Matplotlib основан на иерархии компонентов, которые взаимодействуют для создания графиков. Основные компоненты включают в себя:

1. Фигуры (Figure) – представляют собой весь рисунок или окно для визуализации. Они служат контейнером для всех других элементов графика.

2. Подграфики (Axes) – представляют собой конкретные области рисунка, в которых размещаются графики. Фигура может содержать один или несколько подграфиков.

3. Элементы управления (Widgets) – matplotlib также предоставляет элементы управления для взаимодействия с графиками. Это могут быть ползунки, кнопки и другие виджеты.

Рассмотрим некоторые типы графиков в библиотеке matplotlib.

Линейные графики являются одними из наиболее распространенных и полезных видов графиков. Программный код для построения графика на примере функции sin(x) и его кастомизация представлен на рисунке 1.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)

plt.plot(x, y, label='sin(x)', color='blue', linestyle='--', linewidth=2, marker='o')
plt.title('График синусоиды')
plt.xlabel('Угол (радианы)')
plt.ylabel('Значение синуса')
plt.legend()
plt.grid(True)
plt.show()
```

Рисунок 1 – Программный код для графика синусоиды

Рассмотрим код подробнее:

1. В качестве x выступает массив данных от 0 до 2*π. Элементов в этом массиве 100, шаг определяется автоматически.

2. В качестве y выступает массив значений x, к каждому значению которого применена функция `np.sin(x)`.

3. Для создания графика используется функция `plt.plot()`.

4. Для заголовка используется функция `plt.title('График синусоиды')`.

5. Для подписей осей используются функции `plt.xlabel('Угол (радианы)')` и `plt.ylabel('Значение синуса')`.

6. Для вывода метки (в нашем случае это `label='sin(x)'`) используется функция `plt.legend()`

7. Для отображения сетки на графике используется функция `plt.grid(True)`.

8. Для вывода графика используется функция `plt.show()`.

График синусоиды построенный в примере выше представлен на рисунке 2.

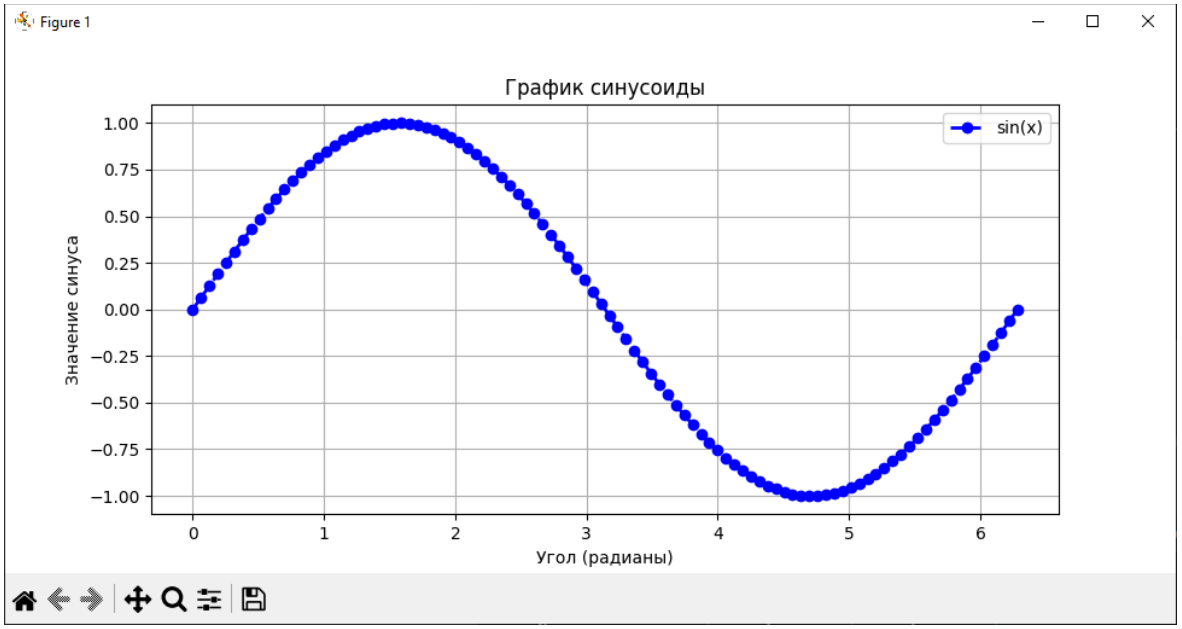

Рисунок 2 – График синусоиды

Рассмотрим столбчатую и круговую диаграммы. Диаграммы предоставляют эффективный способ визуализации категориальных данных.

Программный код для построения столбчатой диаграммы представлен на рисунке 3.

```
import matplotlib.pyplot as plt
import numpy as np

categories = ['Категория A', 'Категория B', 'Категория C']
values = [3, 7, 5]

plt.bar(categories, values, color='green')
plt.title('Столбчатая диаграмма')
plt.xlabel('Категории')
plt.ylabel('Значения')
plt.show()
```

Рисунок 3 – Программный код столбчатой диаграммы

В данном коде диаграмма создается с помощью функции `plt.bar()`. Визуализация диаграммы представлена на рисунке 4.

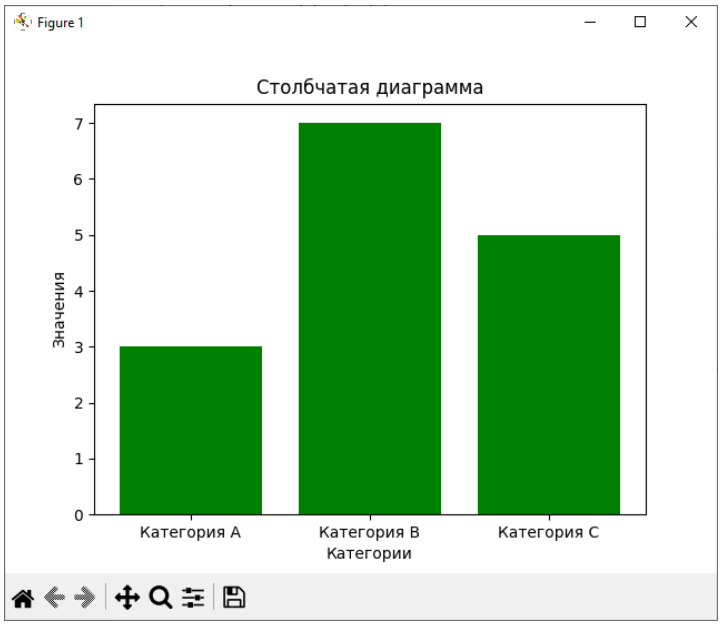

Рисунок 4 – Визуализация столбчатой диаграммы

Рассмотрим программный код для круговой диаграммы. Он представлен на рисунке 5.

```
import matplotlib.pyplot as plt

labels = ['Категория A', 'Категория B', 'Категория C']
sizes = [3, 7, 5]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['red', 'yellow', 'orange'])
plt.title('Круговая диаграмма')
plt.axis('equal')  # Чтобы диаграмма выглядела как круг
plt.show()
```

Рисунок 5 – Программный код круговой диаграммы

В данном программном коде для создания круговой диаграммы используется функция `plt.pie()`. Рассмотрим параметры этой функции:

1. Параметр `autopct='%1.1f%%'` используется для форматирования текста внутри каждого сектора диаграммы. `%1.1f` говорит о том, что нужно отображать один знак после запятой, а `%%` добавляет символ % к отформатированным значениям.

2. Параметр `startangle=90` используется для определения начального угла (в градусах), с которого начинается построение диаграммы.

Так же в данной диаграмме используется функция `plt.axis('equal')`. Она используется для управления параметрами текущих осей. В нашем конкретном случае устанавливает соотношение масштабов по осям x и y равным, что гарантирует, что единицы измерения по обеим осям будут одинаковыми. Это важно для сохранения пропорций, особенно при визуализации круговых диаграмм или других элементов, где важна правильная интерпретация относительных размеров.

Результат построения круговой диаграммы представлен на рисунке 6.

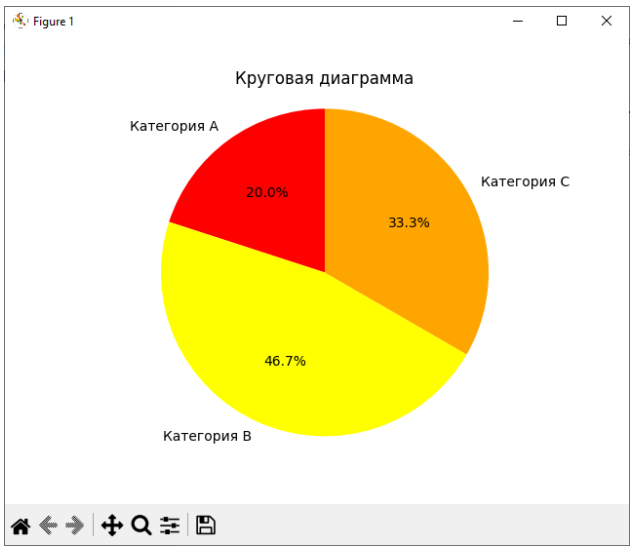

Рисунок 6 – Визуализация круговой диаграммы

Также рассмотрим диаграмму рассеяния. Диаграммы рассеяния эффективны для визуализации взаимосвязи между двумя переменными. Программный код диаграммы рассеяния представлен на рисунке 7.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.rand(100)
y = np.random.rand(100)

plt.scatter(x, y, color='blue')
plt.title('Диаграмма рассеяния')
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.show()
```

Рисунок 7 – Программный код диаграммы рассеяния

Для построения диаграммы используется функция `plt.scatter()`. В качестве значений x и y используются массивы случайно сгенерированных значений в промежутке от 0 до 1.

Визуализация диаграммы рассеяния представлена на рисунке 8.

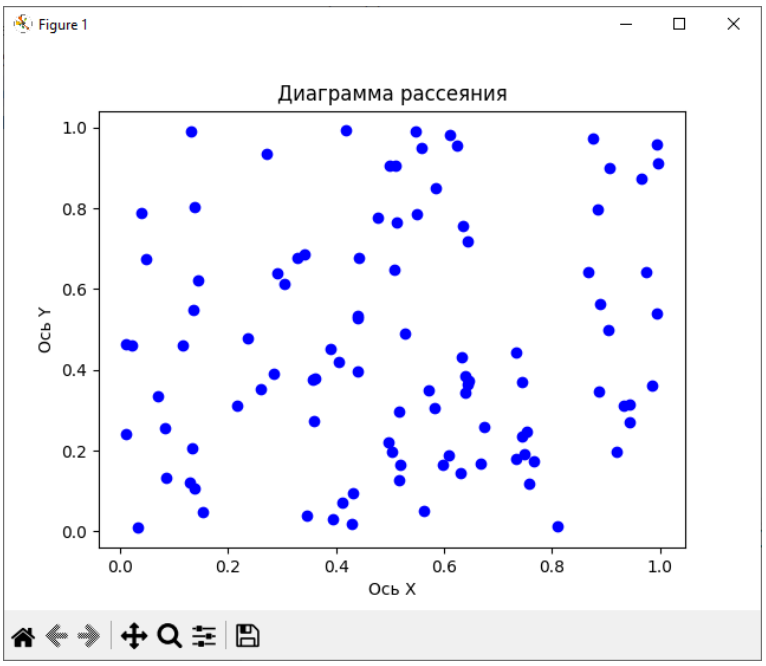

Рисунок 8 – Визуализация диаграммы рассеяния

Последний график, который будет рассмотрен – это контурный график. Контурные графики отображают уровни высоты для трехмерных данных. Программный код для реализации такого графика представлен на рисунке 9.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

plt.contour(X, Y, Z, cmap='viridis')
plt.title('Контурный график sin(sqrt(X^2 + Y^2))')
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.show()
```

Рисунок 9 – Программный код контурного графика

Рассмотрим функции, не использовавшиеся до этого:

1.Функция `np.meshgrid(x, y)` создает двумерные массивы X и Y, используя значения x и y соответственно. Эти массивы представляют собой координатные сетки для создания переменной Z.

2. Функция `plt.contour(X, Y, Z, cmap='viridis')` создает контурный график, используя значения из массивов X и Y для координат и Z для высот. Аргумент `cmap='viridis'` устанавливает цветовую карту.

Визуализация контурного графика представлена на рисунке 10.

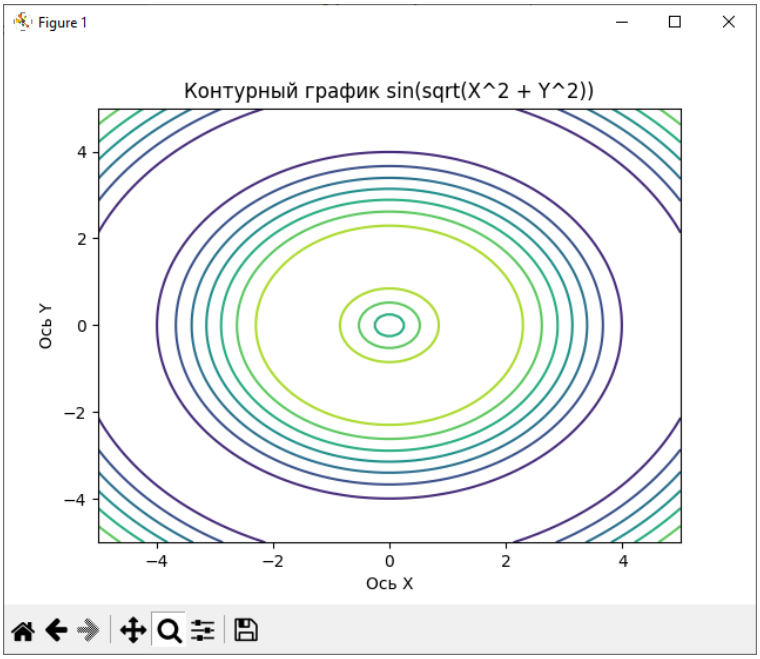

Рисунок 10 – Визуализация контурного графика

Заключение

В данной статье были рассмотрены основные аспекты визуализации данных с использованием библиотеки Matplotlib в языке программирования Python. Это руководство служит введением в основы визуализации данных с использованием Matplotlib, и позволяет читателям начать использовать библиотеку для создания информативных и красочных графиков в их проектах. Однако, для полного охвата возможностей Matplotlib, рекомендуется дополнительное изучение документации и проведение более глубоких экспериментов с данными.

Библиографический список:

1. Гэддис Т. Начинаем программировать на Python [Текст] / Т. Гэддис. – СПб.: БХВ-Петербург, 2021. – 768 с.

2. Маккинли У. Python и анализ данных [Текст] / У. Маккинли. – М.: ДМК

Пресс, 2015. – 482 с.