ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Matplotlib, пожалуй, самая популярная библиотека для Python. Она используется для работы с данными и визуализации машинного обучения по всему миру. Джон Хантер начал разработку Matplotlib в 2003 году. Ее целью была эмуляция команд программы MATLAB, которая в то время являлась научным стандартом. Несколько функций, таких как глобальный стиль MATLAB, были введены в Matplotlib, чтобы облегчить переход на Matplotlib пользователям MATLAB.

Прежде чем мы начнем работать с Matplotlib для создания наших первых визуализаций, мы поймем и постараемся понять концепции, лежащие в основе сюжетов.

Matplotlib состоит из множества модулей. Модули наполнены различными классами и функциями, которые иерархически связаны между собой.

Создание рисунка в matplotlib схоже с рисованием в реальной жизни. Так художнику нужно взять основу (холст или бумагу), инструменты (кисти или карандаши), иметь представление о будущем рисунке (что именно он будет рисовать) и, наконец, выполнить всё это и нарисовать рисунок деталь за деталью.

В matplotlib все эти этапы также существуют, и в качестве художника-исполнителя здесь выступает сама библиотека. От пользователя требуется управлять действиями художника-matplotlib, определяя что именно он должен нарисовать и какими инструментами. Обычно создание основы и процесс непосредственно отображения рисунка отдаёт полностью на откуп matplotlib. Таким образом, пользователь библиотеки matplotlib выступает в роли управленца. И чем проще ему управлять конечным результатом работы matplotlib, тем лучше.

Так как matplotlib организована иерархически, а наиболее простыми для понимания человеком являются самые высокоуровневые функции, то знакомство с matplotlib начинают с самого высокоуровневого интерфейса **matplotlib.pyplot**. Так, чтобы нарисовать гистограмму с помощью этого модуля, нужно вызывать всего одну команду: matplotlib.pyplot.hist(arr).

Пользователю не нужно думать как именно библиотека нарисовала эту диаграмму. Если бы мы рисовали гистрограмму самостоятельно , то заметили бы, что она состоит из повоторяющихся по форме фигур - прямоугольников. А чтобы нарисовать прямоугольник, нужно знать хотя бы координату одного угла и ширину/длину. Рисовали же бы мы прямоугольник линиями, соединяя угловые точки прямоугольника.

Этот пример отображает иерархичность рисунков, когда итоговая диаграмма (высокий уровень) состоит из простых геометрических фигур (более низкий, средний уровень), созданных несколькими универсальными методами рисования (низкий уровень). Если бы каждый рисунок нужно было бы создавать вот так, с нуля, это было бы очень долго и утомительно.

Интерфейс matplotlib.pyplot является набором команд и функций, которые делают синтаксис графических matploltib команд похожим на команды, используемые в среде MATLAB(c). Изначально matplotlib планировался как свободная альтернатива MATLAB(c), где в одной среде имелись бы средства как для рисования, так и для численного анализа. Именно так в Matplotlib появился pylab, который объединяет модули pyplot и numpy в одно пространство имён.

2. ОБЗОР ГРАФИКОВ В MATPLOTLIB

Участки в Matplotlib имеют иерархическую структуру, которая гнездит объекты Python для создания древовидной структуры. Каждый участок инкапсулирован в объект Figure. Этот рисунок является контейнером верхнего

уровня визуализации. Он может иметь несколько осей, которые в основном являются отдельные участки внутри этого контейнера верхнего уровня. Переходя на более глубокий уровень, мы снова находим объекты Python, которые управляют осями, галочками, легендами, заголовками, текстовыми полями, сеткой и многими другими объектами. Все эти объекты можно настроить.

Двумя основными компонентами графика являются следующие:

- Рисунок

Рисунок является крайним контейнером и используется в качестве холста для рисования. Он позволяет нарисовать несколько графиков внутри него. Он не только содержит объект Axes, но и имеет возможность настроить Title.

- Axes

Топоры - это фактический график, или подплот, в зависимости от того, хотите ли вы построить одну или несколько визуализаций. Его подобъекты включают оси x и y, позвоночники и легенды.

Наблюдая за этим дизайном на более высоком уровне, мы видим, что эта иерархическая структура позволяет нам создавать сложную и настраиваемую визуализацию.

Глядя на "анатомию" рисунка, которая показана на следующей диаграмме, мы получаем представление о сложности проницательной визуализации. Matplotlib дает нам возможность не только просто отображать данные, но и проектировать весь рисунок вокруг него, настраивая Grid, x и y, отмечая метки, и Legend. Это означает, что мы можем изменять каждый бит графика, начиная с Title и Legend, и заканчивая даже главными и второстепенными галочками на корешках, чтобы сделать его более выразительным:

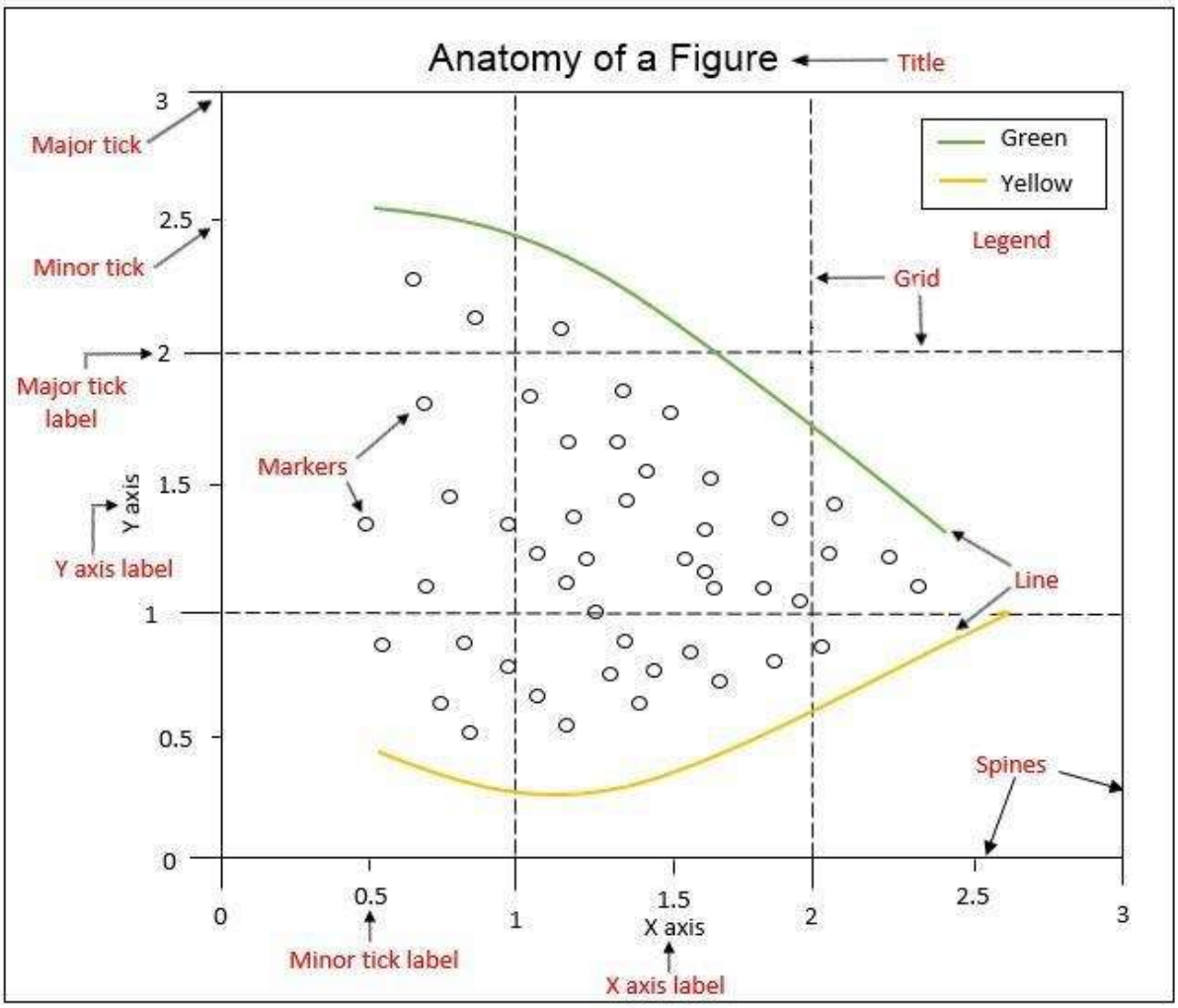

Figure 3.1: Анатомия фигуры Матплотлиба.

Глубоко изучив анатомию объекта "Figure", мы можем наблюдать следующие компоненты:

- **Spines**: Линии, соединяющие отметки оси

  Title: Текстовая метка всего объекта рисунка
- **Legend**: Они описывают содержание сюжета

  Grid: Вертикальные и горизонтальные линии, используемые в качестве продолжения маркеров
- **X/Y axis label**: Текстовая метка для оси X/Y под корешками
- **Minor tick:** Индикаторы малого значения между основными маркерами
- **Minor tick label**: Text label that will be displayed at the minor ticks
- **Major tick**: Основные стоимостные показатели на позвоночнике

- **Major tick label**: Текстовая метка, которая будет отображаться в основной строке тиков: Тип чертежа, который соединяет точки данных с линией.
- **Markers**: Тип построения графиков, который рисует каждую точку данных с определенным маркером.

3. STACKED AREA CHART

plt.stackplot(x, y)

Важные параметры:

x: Определяет x-значения ряда данных.

y: Определяет y-значения серии данных. Для нескольких серий, либо в виде массива 2d, либо в виде любого количества 1D массивов, необходимо вызвать следующую функцию: **plt.stackplot(x, y1, y2, y3, ...).**

labels (необязательно): Указание меток в виде списка или кортежа для каждого ряда данных.

Example:

```
x = np.arange(0, 11, 1)
y1 = np.array([(-0.2)*i**2+2*i for i in x])
y2 = np.array([(-0.4)*i**2+4*i for i in x])
y3 = np.array([2*i for i in x])
labels = ["y1", "y2", "y3"]
fig, ax = plt.subplots()
ax.stackplot(x, y1, y2, y3, labels=labels)
ax.legend(loc='upper left')
```

Результат применения предыдущего кода показан на следующей диаграмме:

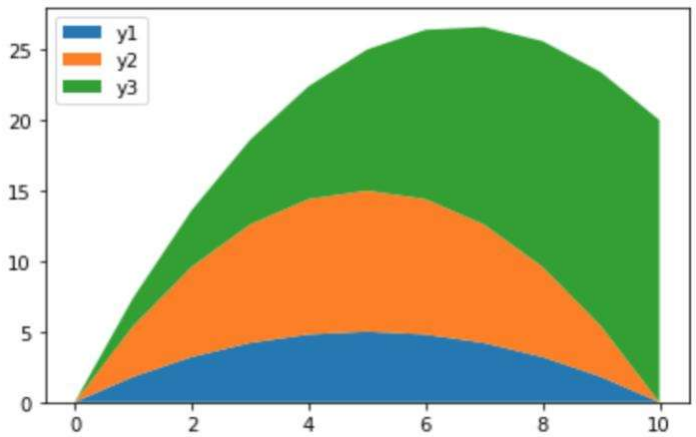

Figure 1: Stacked area chart

4. BAR CHART

Длина Бара кодирует значение. Существует два варианта гистограммы: вертикальная гистограмма и горизонтальная гистограмма.

Используется:

- Хотя оба они используются для сравнения числовых значений по категориям, вертикальные
- Иногда гистограммы используются для отображения одной переменной с течением времени.

Что в барах есть, а чего в барах нет:

- Не путайте вертикальные бары с гистограммами. Бары сравнивают различные переменные или категории, а гистограммы показывают распределение для одной переменной. Гистограммы будут рассмотрены далее в этой главе.
- Другой распространенной ошибкой является использование баров для отображения центральных тенденций между группами или категориями. Используйте box plot или violin plots, чтобы показать статистические показатели или распределения в этих случаях.

Examples:

```
np.random.seed(123)
groups = [f"P{i}" for i in range(7)]
counts = np.random.randint(3, 10, len(groups))
plt.bar(groups, counts)
```

На следующем рисунке показана вертикальная гистограмма. Каждый столбец показывает оценки из 10, полученные семью студентами за тест:

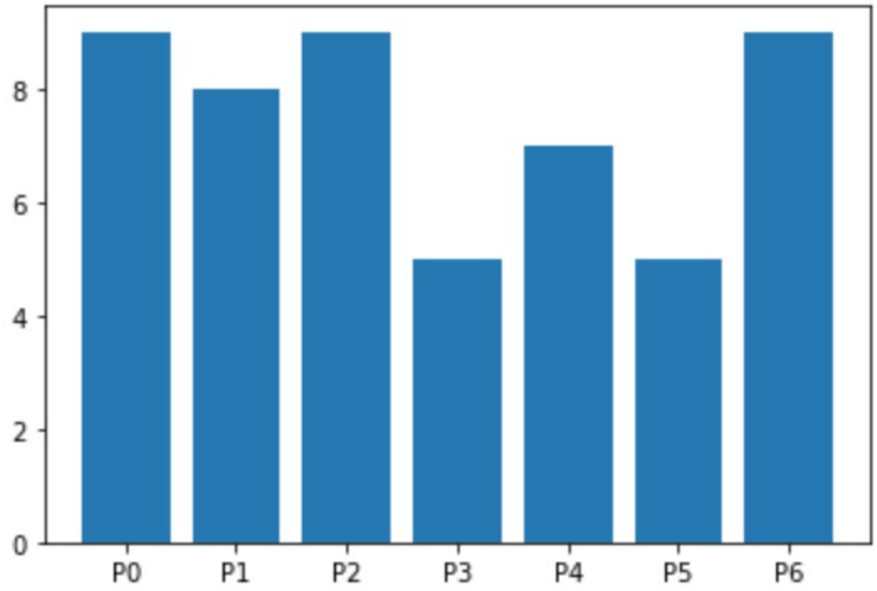

Figure 2: Вертикальный bar с использованием данных студенческих тестов

Если заменим bar() на barh() получим горизонтальную диаграмму:

plt.barh(groups, counts)

На следующем рисунке показана горизонтальный bar. Каждый столбец показывает оценки из 10, полученные семью студентами за тест:

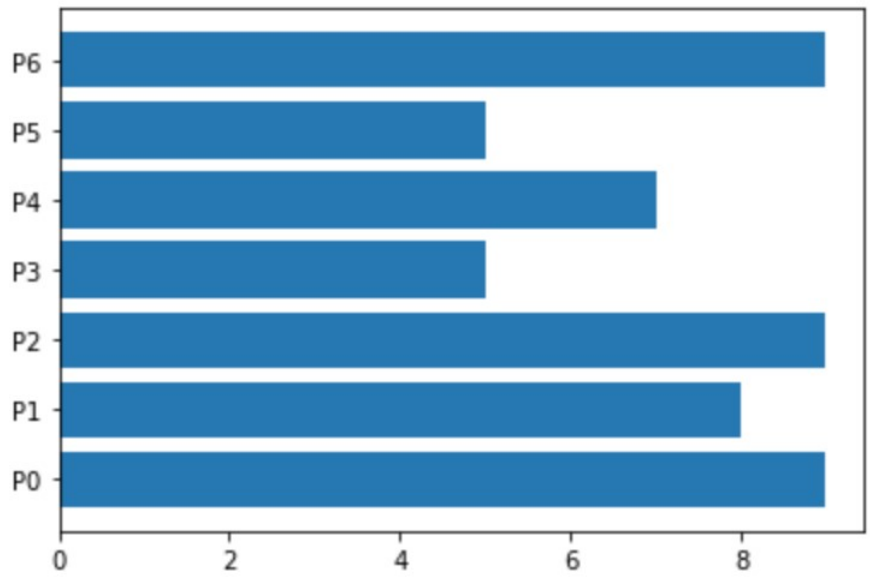

Figure 3: Горизонтальный bar с использованием данных студенческих тестов

Следующая диаграмма сравнивает рейтинги фильмов, давая две разные оценки. Томатометр - это процент одобренных критиков, которые дали положительный отзыв на фильм. Следующая диаграмма сравнивает рейтинги фильмов, давая два разных балла. Оценка зрительских симпатий - это процент пользователей, которые дали оценку 3,5 или выше из 5. Как мы видим, "Марсианин" - единственный фильм с высоким баллом "Томатометр" и оценкой зрительских симпатий. Хоббит: Неожиданное путешествие имеет относительно высокий балл зрительских симпатий по сравнению с томатометром, что может быть связано с огромной базой фанатов:

```
cat_par = [f"P{i}" for i in range(5)]
g1 = [10, 21, 34, 12, 27]
g2 = [17, 15, 25, 21, 26]
width = 0.3
x = np.arange(len(cat_par))
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, g1, width, label='g1')
rects2 = ax.bar(x + width/2, g2, width, label='g2')
ax.set_title('Пример групповой диаграммы')
ax.set_xticks(x)
ax.set_xticklabels(cat_par)
ax.legend()
```

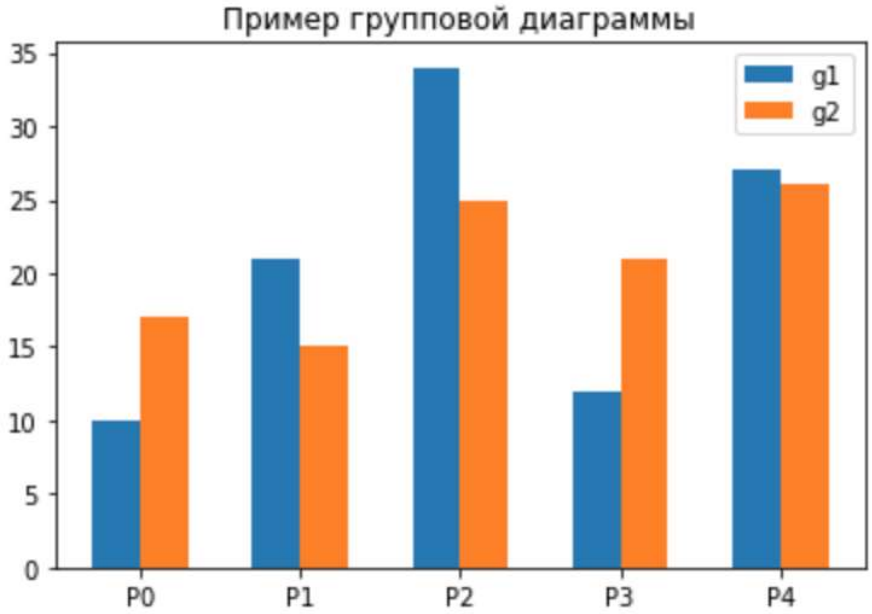

Figure 4: Сравнительный bar chart

Практика проектирования:

- Ось, соответствующая числовой переменной, должна начинаться с нуля. Начало с другого значения может ввести в заблуждение, так как оно делает небольшую разницу значений похожей на большую.
- Используйте горизонтальные метки, то есть до тех пор, пока количество баров маленькое и график не выглядит слишком загроможденным.

5. HISTOGRAM

Гистограмма визуализирует распределение одной числовой переменной. Каждый столбец представляет собой частоту для определенного интервала. Гистограммы помогают получить оценку статистических показателей. Вы видите, где сконцентрированы значения, и можете легко обнаружить промахи. Вы можете построить гистограмму с абсолютными значениями частоты или нормализовать гистограмму. Если вы хотите сравнить распределения нескольких переменных, то для столбцов можно использовать различные цвета.

Используется:

- Получить представление об основном распределении набора данных.

Example:

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

# Fixing random state for reproducibility
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.xlim(40, 160)
plt.ylim(0, 0.03)
plt.grid(True)
plt.show()
```

На следующей диаграмме показано распределение коэффициента интеллекта (IQ) для тестовой группы:

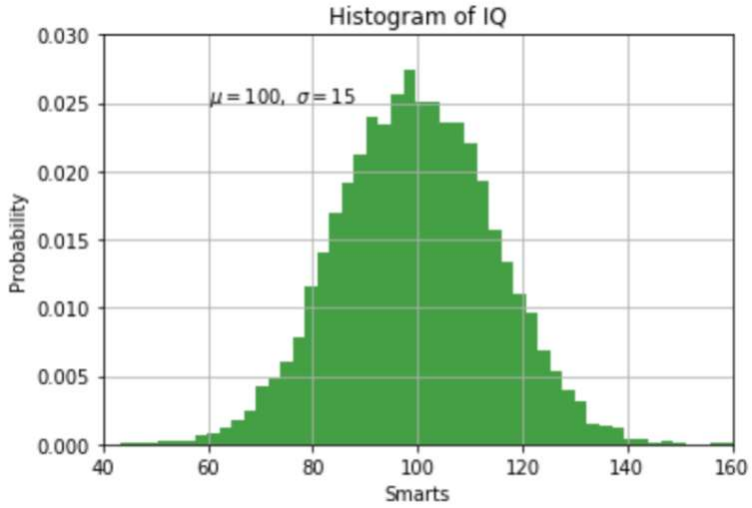

Figure 5: Распределение коэффициента интеллекта (IQ) для тестовой группы из ста взрослых.

Практика проектирования:

- Попробуйте разное количество бункеров, так как форма гистограммы может варьироваться значительно.

6. ПРАКТИЧЕСКАЯ ЧАСТЬ

1. **Задание 1: Stacked Area Chart**

Добавляем интересующие нас библиотеки для реализации данного задания:

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

Создаем массивы для обозначения области определения двух надставленных областей и соответствующие им области значений:

```
1. x = [0, 1, 2, 3, 4, 5]
2.
3. y1 = [2, 6, 4, 8, 2, 4]
4. y2 = [2, 4, 2, -1, 4, -2]
```

Создаем область Figure, а после добавляем область Axes:

```
1. fig, ax = plt.subplots()
```

Создаем надставленные области на основе созданных нами массивов:

```
1. ax.stackplot(x, [y1, y2])
```

С помощью разных свойств мы можем изменить ширину, высоту, цвет и многое другое. Добавьте еще 3 свойства для того, чтобы изменить наш Figure.

```
1. fig.set_figwidth(12)     #  ширина и
2. fig.set_figheight(6)     #  высота "Figure"
3. fig.set_facecolor('floralwhite')
4. ax.set_facecolor('seashell')
```

Демонстрируем полученный Figure.

```
1. plt.show()
```

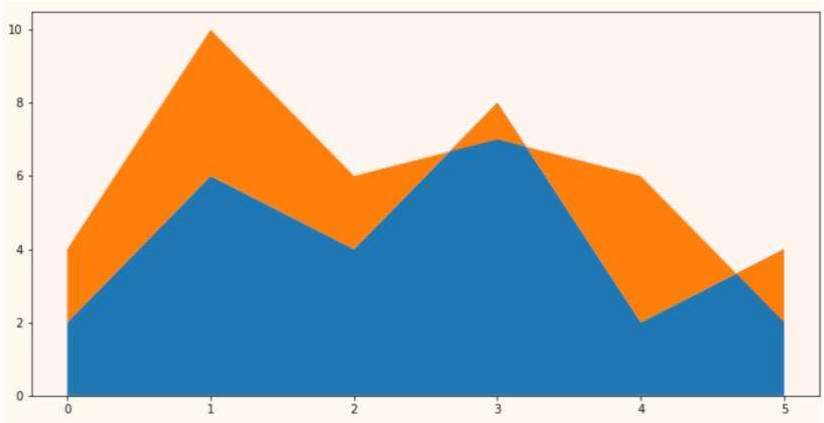

Figure 6: Результат выполнения задания 1

2. **Задание 2: Bar Chart**

Добавляем интересующие нас библиотеки для реализации данного задания:

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

Создаем массив для дальнейшей работы с ним

```
1. data = [[30, 25, 50, 65],[25, 35, 70, 15],[40, 30, 30, 45]]
```

Создаем одномерный массив, используя библиотеку numpy

```
1. X = np.arrange(4)
```

Создаем область Figure, а после добавляем область Axes:

```
1. fig = plt.figure()
2. ax = fig.add_axes([0,0,1,1])
```

Мы можем построить несколько гистограмм, играя с толщиной и положением баров. Переменная данных содержит три ряда из четырех значений. Следующий скрипт покажет три гистограммы по четыре бара. Толщина баров составит 0.25 единицы. Каждый баровый график будет смещен на 0.25 единицы по сравнению с предыдущим.

```
1. ax.bar(X + 0.00, data[0], color = ‘b’, width = 0.25)
2. ax.bar(X + 0.25, data[1], color = ‘g’, width = 0.25)
3. ax.bar(X + 0.50, data[2], color = ‘r’, width = 0.25)
```

Демонстрируем полученный Figure.

```
1. plt.show()
```

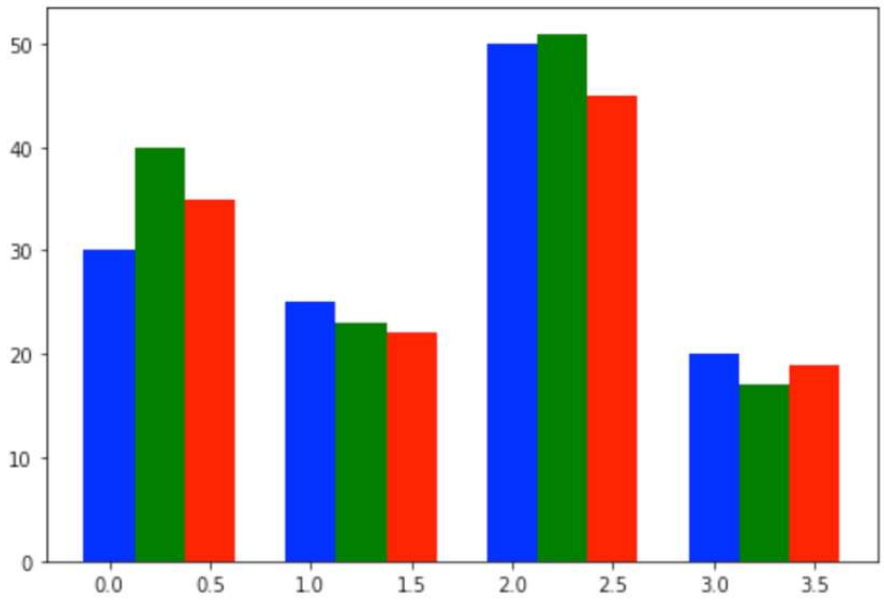

Figure 7: Результат выполнения задания 2

3. **Задание 3: Histogram**

Добавляем интересующие нас библиотеки для реализации данного задания:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import matplotlib
```

Создаем данные для дальнейшей работы, используя библиотеку numpy

```
1. np.random.seed(10**7)
2. n_bins = 20
3. x = np.random.randn(10000, 3)
4.
5. colors = ['green', 'blue', 'lime']
```

Создаем гистограмму, сразу применяя к ней свойства лейбла и цвета

```
1.  plt.hist(x, n_bins, density = True,
2.           histtype ='bar',
3.           color = colors,
4.           label = colors)
```

Изменяем размер легенд и название заголовка

```
1. plt.legend(prop ={'size': 10})
2.
3. plt.title('matplotlib.pyplot.hist() function Example\n\n',
4.           fontweight ="bold")
```

Демонстрируем полученный Figure.

```
1. plt.show()
```

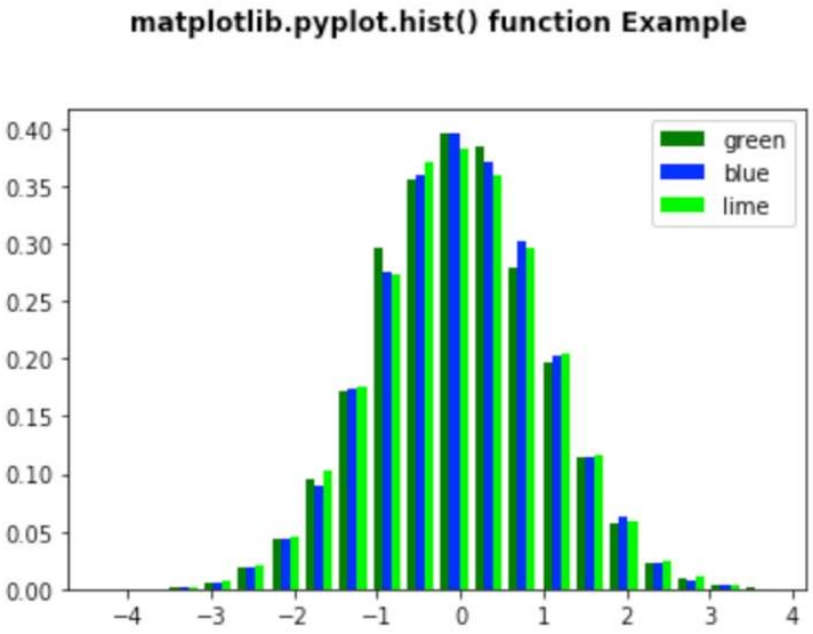

Figure 8: Результат выполнения задания 3

4. Создайте гистограмму, которая будет отражать показатели посещаемости студентов вашей группы на лабораторных работах за последние 2 недели

5. Создайте **Bar Chart** , которая будет отражать показатели посещаемости студентов вашей группы на лабораторных работах за последние 2 недели, добавьте в вывод что наиболее удобно для работы и почему

6. Нарисуйте 3 уровня лестницы с помощью **Stacked Area Chart** различными цветами, добавьте лейблы и название графика

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики были изучены базовые инструменты визуализации в Python, был собран материал, необходимый для написания отчета.

Данная практика является хорошим практическим опытом для дальнейшей самостоятельной деятельности. За время пройденной практики я познакомился с новыми методами представления графических данных. Закрепил свои теоретические знания, лучше ознакомилась со своей профессией, а также данный опыт послужит хорошей ступенькой в моей дальнейшей карьерной лестнице.

СПИСОК ЛИТЕРАТУРЫ

1. Data Visualization with Python Copyright Mario Döbler and Tim Großmann © 2019 Packt Publishing.