บทที่ 7 การแสดงผลข้อมูลด้วยกราฟ
(Data Visualization)

การแสดงผลข้อมูลด้วยกราฟ (Data Visualization) เป็นเครื่องมือที่สำคัญในการทำความเข้าใจข้อมูลและการสื่อสารผลการวิเคราะห์ กราฟต่าง ๆ ช่วยให้เราสามารถสรุปและแสดงแนวโน้ม รูปแบบ และความสัมพันธ์ในข้อมูลได้ง่ายขึ้น โดยการเลือกกราฟที่เหมาะสมจะทำให้การวิเคราะห์และการสื่อสารข้อมูลมีประสิทธิภาพมากขึ้น

1. การใช้กราฟเพื่อแสดงผลข้อมูลในรูปแบบที่เข้าใจง่าย

การแสดงผลข้อมูลด้วยกราฟ (Data Visualization) เป็นกระบวนการนำเสนอข้อมูลในรูปแบบกราฟิก เช่น กราฟ แผนภูมิ และแผนภาพ เพื่อให้ผู้ใช้สามารถทำความเข้าใจข้อมูลได้ง่ายขึ้น การแสดงผลข้อมูลช่วยให้เราสามารถเห็นภาพรวมของข้อมูล ค้นหารูปแบบ แนวโน้ม และความสัมพันธ์ที่อาจไม่สามารถมองเห็นได้อย่างชัดเจนจากการดูข้อมูลดิบเพียงอย่างเดียว

การแสดงผลข้อมูลที่ดีสามารถช่วยสื่อสารข้อมูลได้อย่างมีประสิทธิภาพ สนับสนุนการตัดสินใจ และทำให้การวิเคราะห์ข้อมูลเป็นไปอย่างรวดเร็วและมีประสิทธิภาพมากขึ้น

1.1 ความสำคัญของการแสดงผลข้อมูลด้วยกราฟ

1.1.1 ทำให้ข้อมูลเข้าใจง่ายขึ้น (Simplify Data Understanding) การแสดงข้อมูลในรูปแบบกราฟทำให้ข้อมูลที่ซับซ้อนและมีจำนวนมากเข้าใจได้ง่ายขึ้น โดยการลดจำนวนรายละเอียดที่ไม่จำเป็นและนำเสนอข้อมูลในลักษณะที่มองเห็นภาพรวมได้ชัดเจน

1.1.2 ค้นหารูปแบบและแนวโน้ม (Identify Patterns and Trends) กราฟช่วยให้เราค้นพบแนวโน้มและรูปแบบในข้อมูล เช่น การเพิ่มขึ้นหรือลดลงของยอดขายในแต่ละเดือน หรือการกระจายตัวของข้อมูลในแต่ละหมวดหมู่

1.1.3 สื่อสารผลการวิเคราะห์ (Communicate Insights Effectively) การแสดงผลข้อมูลช่วยให้การสื่อสารผลการวิเคราะห์กับผู้อื่นทำได้ง่ายขึ้น ไม่ว่าจะเป็นการนำเสนอในที่ประชุม การรายงาน หรือการสร้าง Dashboard สำหรับการตัดสินใจทางธุรกิจ

1.1.4 เปรียบเทียบข้อมูล (Compare Data) กราฟทำให้เราสามารถเปรียบเทียบข้อมูลระหว่างกลุ่มต่าง ๆ ได้ง่ายขึ้น เช่น การเปรียบเทียบยอดขายระหว่างผลิตภัณฑ์สองตัว หรือการเปรียบเทียบผลการดำเนินงานในแต่ละปี

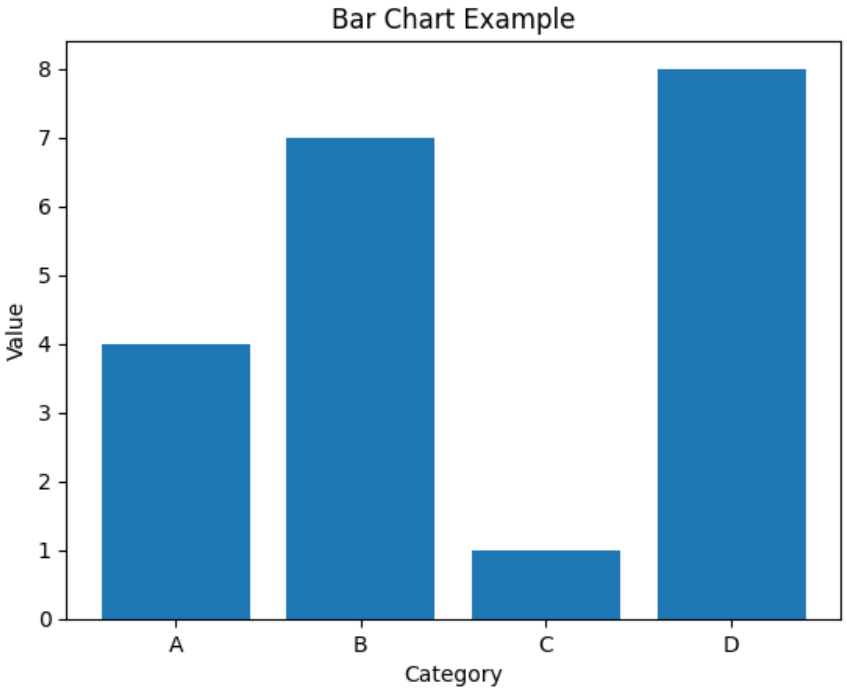
1.2 ประเภทของกราฟที่ใช้ในการแสดงผลข้อมูล

1.2.1 แผนภูมิแท่ง

แผนภูมิแท่ง (Bar Chart) ใช้แสดงการเปรียบเทียบข้อมูลระหว่างหมวดหมู่ต่าง ๆ โดยใช้ความยาวของแท่งเพื่อแสดงขนาดของข้อมูล เหมาะสำหรับการเปรียบเทียบข้อมูล เช่น ยอดขายในแต่ละเดือนหรือจำนวนผู้ใช้ในแต่ละภูมิภาค

ตัวอย่าง

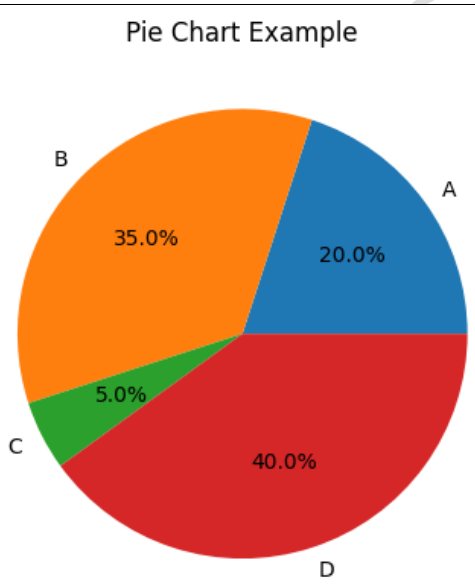
โค้ด	import matplotlib.pyplot as plt
	
	# ข้อมูลตัวอย่าง
	categories = ['A', 'B', 'C', 'D']
	values = [4, 7, 1, 8]
	
	# สร้าง Bar Chart
	plt.bar(categories, values)
	plt.title('Bar Chart Example')
	plt.xlabel('Category')
	plt.ylabel('Value')
	plt.show()

ผลลัพธ์	

1.2.2 แผนภูมิวงกลม

แผนภูมิวงกลม (Pie Chart) ใช้แสดงการแบ่งสัดส่วนหรือส่วนแบ่งของแต่ละหมวดหมู่ภายในข้อมูลทั้งหมด เหมาะสำหรับการแสดงสัดส่วน เช่น ส่วนแบ่งการตลาดของผลิตภัณฑ์แต่ละชนิด

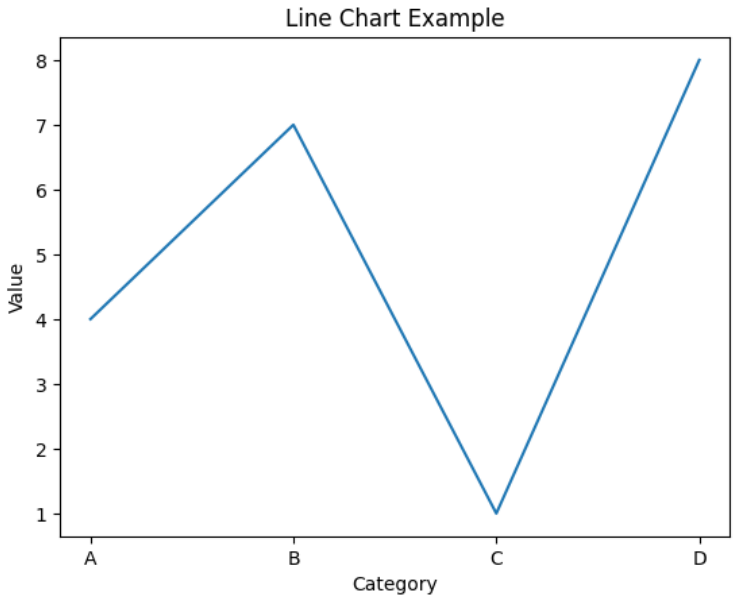
โค้ด	import matplotlib.pyplot as plt
	
	# สร้าง Pie Chart
	plt.pie(values, labels=categories, autopct='%1.1f%%')
	plt.title('Pie Chart Example')
	plt.show()

ผลลัพธ์	

1.2.3 กราฟเส้น

กราฟเส้น (Line Chart) ใช้แสดงการเปลี่ยนแปลงของข้อมูลเมื่อเวลาผ่านไป โดยการเชื่อมต่อข้อมูลแต่ละจุดด้วยเส้น เหมาะสำหรับการแสดงแนวโน้ม เช่น ยอดขายที่เปลี่ยนแปลงในแต่ละเดือน หรือจำนวนผู้ใช้งานที่เพิ่มขึ้นหรือลดลงในแต่ละปี

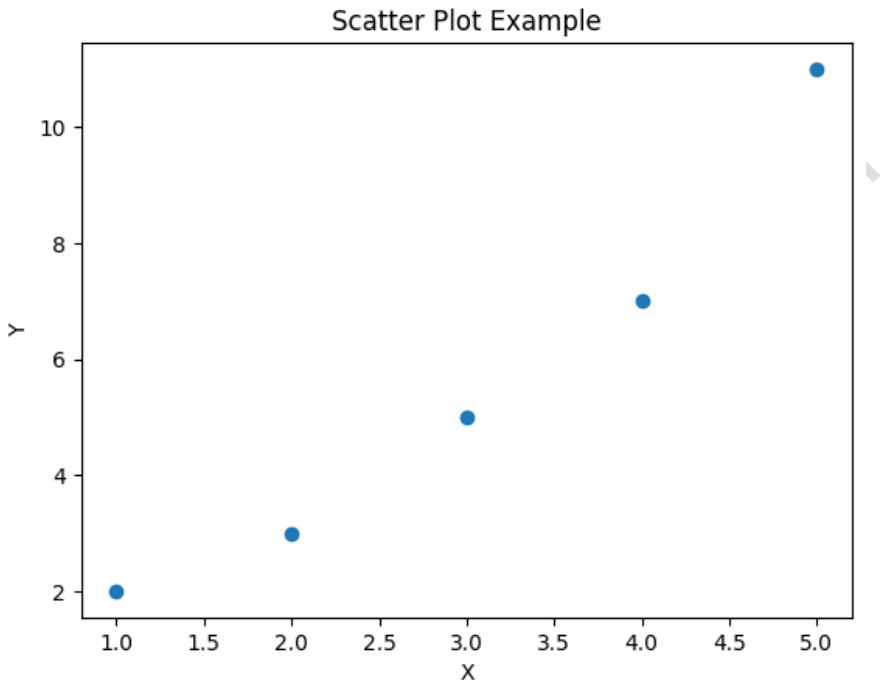
โค้ด	import matplotlib.pyplot as plt # สร้าง Line Chart

	plt.plot(categories, values) plt.title('Line Chart Example') plt.xlabel('Category') plt.ylabel('Value') plt.show()
ผลลัพธ์	

1.2.4 กราฟการกระจาย

กราฟการกระจาย (Scatter Plot) ใช้แสดงความสัมพันธ์ระหว่างตัวแปรสองตัว โดยใช้จุดเพื่อแสดงข้อมูล เหมาะสำหรับการวิเคราะห์ความสัมพันธ์ระหว่างตัวแปร เช่น ความสัมพันธ์ระหว่างอายุและรายได้ของผู้ใช้

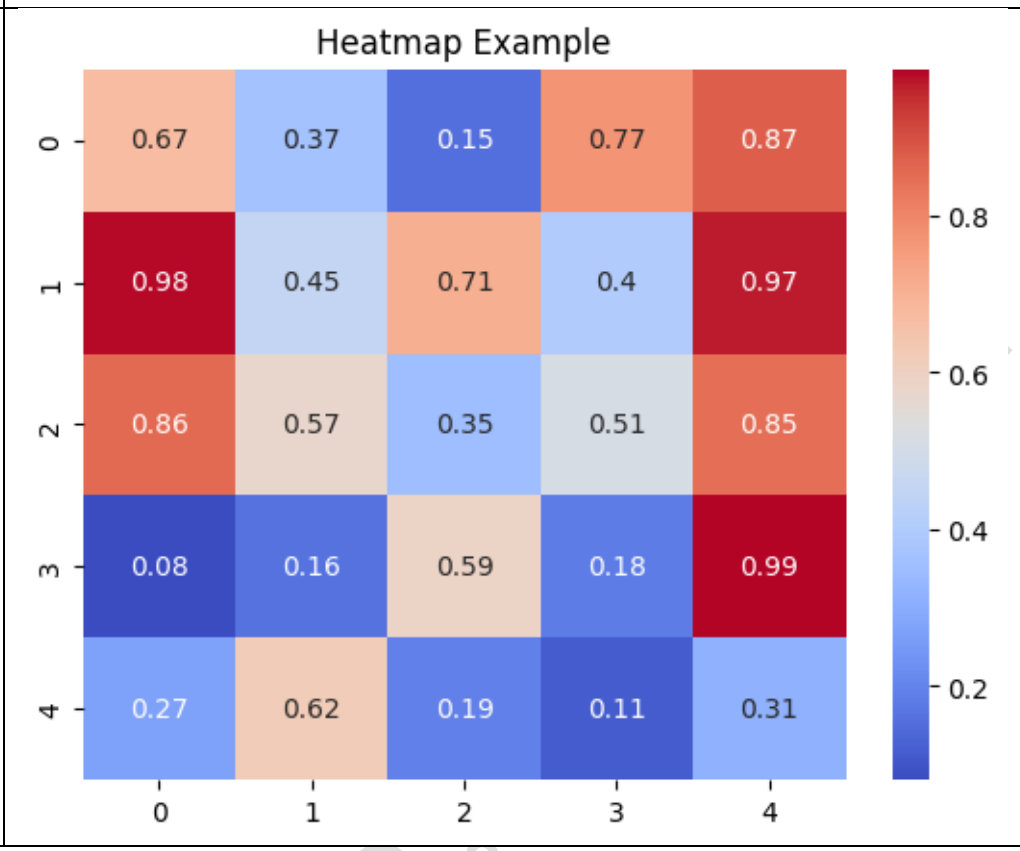
โค้ด	import matplotlib.pyplot as plt
	
	# ข้อมูลตัวอย่าง
	x = [1, 2, 3, 4, 5]
	y = [2, 3, 5, 7, 11]
	
	# สร้าง Scatter Plot
	plt.scatter(x, y)

	plt.title('Scatter Plot Example') plt.xlabel('X') plt.ylabel('Y') plt.show()
ผลลัพธ์	

1.2.5 ฮีทแมพ

ฮีทแมพ (Heatmap) ใช้แสดงค่าต่าง ๆ ในตารางหรือแมตริกซ์ โดยใช้สีเพื่อแสดงค่าที่แตกต่างกัน เหมาะสำหรับการวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรหลายตัว เช่น การวิเคราะห์ความสัมพันธ์เชิงสถิติ (Correlation Matrix)

โค้ด	import seaborn as sns
	import numpy as np
	
	# ข้อมูลตัวอย่าง (สร้างแมตริกซ์ 5x5 แบบสุ่ม)
	data = np.random.rand(5, 5)
	
	# สร้าง Heatmap
	sns.heatmap(data, annot=True, cmap='coolwarm')
	plt.title('Heatmap Example')

	plt.show()
ผลลัพธ์	

1.3 เครื่องมือสำหรับการแสดงผลข้อมูล

1.3.1 Matplotlib เป็นไลบรารีพื้นฐานสำหรับการสร้างกราฟใน Python โดยให้ความยืดหยุ่นสูงและสามารถปรับแต่งกราฟได้หลากหลายรูปแบบ

1) ข้อดี มีความยืดหยุ่นสูง สามารถสร้างกราฟได้หลากหลาย

2) ข้อเสีย การสร้างกราฟที่ซับซ้อนอาจใช้โค้ดที่ยาวและซับซ้อน

1.3.2 Seaborn เป็นไลบรารีที่ถูกสร้างขึ้นบนพื้นฐานของ Matplotlib แต่เพิ่มความสามารถในการสร้างกราฟเชิงสถิติ และปรับแต่งการแสดงผลให้สวยงามมากขึ้น

1) ข้อดี ใช้งานง่าย สร้างกราฟสถิติได้อย่างสวยงามและรวดเร็ว

2) ข้อเสีย ไม่ยืดหยุ่นเท่า Matplotlib ในการปรับแต่งกราฟขั้นสูง

1.3.3 Plotly เป็นไลบรารีสำหรับสร้างกราฟเชิงโต้ตอบ (Interactive Graph) ที่สามารถแสดงผลในเว็บเบราว์เซอร์ได้

1) ข้อดี สร้างกราฟเชิงโต้ตอบได้ และสามารถใช้บนเว็บเบราว์เซอร์

2) ข้อเสีย ใช้งานยากขึ้นเล็กน้อยเมื่อเทียบกับ Matplotlib และ Seaborn

1.3.4 Tableau และ Power BI เป็นเครื่องมือแสดงผลข้อมูลที่เน้นการใช้งานง่ายและเป็นมิตรกับผู้ใช้ ซึ่งนิยมใช้ในงานธุรกิจและการวิเคราะห์ข้อมูลขนาดใหญ่

1) ข้อดี สร้างกราฟได้อย่างง่ายดาย และรวดเร็ว ไม่ต้องใช้การเขียนโค้ด
2) ข้อเสีย ต้องใช้ซอฟต์แวร์เฉพาะและอาจมีค่าใช้จ่าย

1.4 การเลือกกราฟที่เหมาะสมกับข้อมูล

การเลือกกราฟที่เหมาะสมกับข้อมูลเป็นสิ่งสำคัญเพื่อให้การแสดงผลข้อมูลมีประสิทธิภาพ กราฟแต่ละประเภทเหมาะกับการนำเสนอข้อมูลที่แตกต่างกัน เช่น

1.4.1 Bar Chart เหมาะสำหรับการเปรียบเทียบข้อมูลระหว่างหมวดหมู่
1.4.2 Pie Chart เหมาะสำหรับแสดงส่วนแบ่งหรือสัดส่วน
1.4.3 Line Chart เหมาะสำหรับแสดงแนวโน้มของข้อมูลที่เปลี่ยนแปลงตามเวลา
1.4.4 Scatter Plot เหมาะสำหรับการแสดงความสัมพันธ์ระหว่างตัวแปรสองตัว
1.4.5 Heatmap เหมาะสำหรับแสดงค่าหลายตัวแปรในรูปแบบตาราง

2. การสร้างกราฟขั้นสูง

การสร้างกราฟขั้นสูง (Advanced Data Visualization) คือ กระบวนการที่ซับซ้อนกว่าการสร้างกราฟพื้นฐาน เนื่องจากต้องการนำเสนอข้อมูลหลายตัวแปรหรือการแสดงข้อมูลที่มีความซับซ้อนมากขึ้น โดยมีการปรับแต่งรูปแบบต่าง ๆ เพื่อทำให้การแสดงผลข้อมูลมีประสิทธิภาพมากขึ้น กราฟขั้นสูงช่วยให้ผู้ใช้สามารถวิเคราะห์และตีความข้อมูลได้ง่ายขึ้นและครบถ้วนยิ่งขึ้น

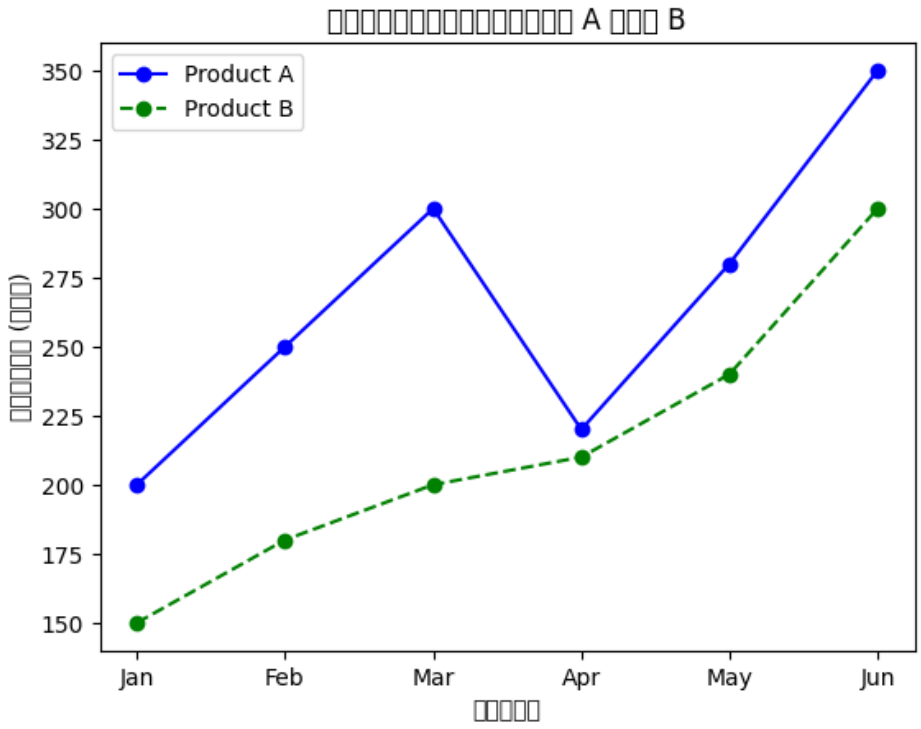
กราฟขั้นสูงมีหลายประเภท เช่น การแสดงกราฟหลายเส้น การแสดงกราฟที่มีแกนสองด้าน (Dual-Axis), การใช้ Subplots (หลายกราฟในหนึ่งเดียว), การสร้าง Heatmap สำหรับแสดงความสัมพันธ์ระหว่างตัวแปร และการสร้างกราฟเชิงโต้ตอบ (Interactive Visualization) ด้วยไลบรารีขั้นสูง โดยมีรายละเอียดดังนี้

2.1 กราฟหลายเส้น

กราฟหลายเส้น (Multiple Line Charts) ใช้ในการแสดงข้อมูลหลายชุดบนกราฟเดียวกันเพื่อทำการเปรียบเทียบ เช่น การเปรียบเทียบยอดขายของสินค้าหลายชนิดในช่วงเวลาเดียวกัน

โค้ด	import matplotlib.pyplot as plt
	
	# ข้อมูลตัวอย่าง
	months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
	sales_A = [200, 250, 300, 220, 280, 350]

	sales_B = [150, 180, 200, 210, 240, 300]
	
	# สร้างกราฟหลายเส้น
	plt.plot(months, sales_A, marker='o', linestyle='-', color='b', label='Product A')
	plt.plot(months, sales_B, marker='o', linestyle='--', color='g', label='Product B')
	plt.title('ยอดขายผลิตภัณฑ์ A และ B')
	plt.xlabel('เดือน')
	plt.ylabel('ยอดขาย (บาท)')
	plt.legend()
	plt.show()

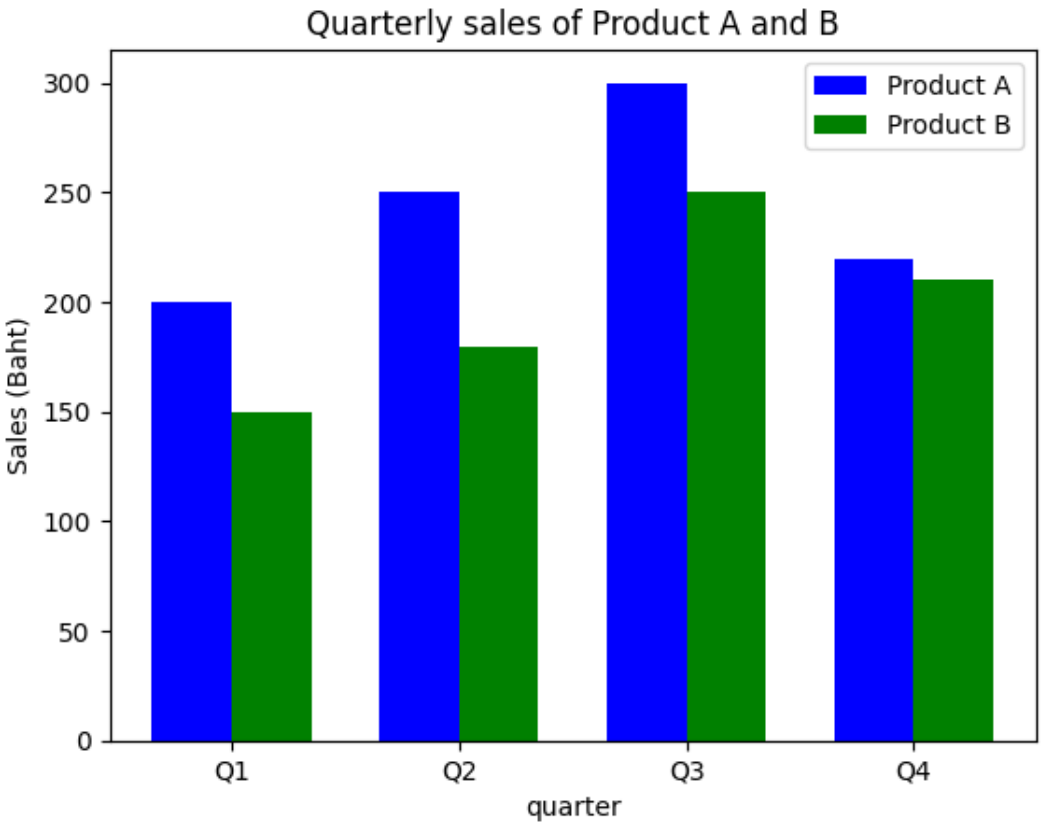
ผลลัพธ์	

จากตัวอย่าง เส้นสองเส้นแสดงยอดขายของ Product A และ B ในช่วงเวลาเดียวกัน เพื่อเปรียบเทียบข้อมูลสองชุดในกราฟเดียวกัน

2.2 กราฟแท่งแบบกลุ่ม (Grouped Bar Chart)

กราฟแท่งแบบกลุ่มใช้ในการแสดงข้อมูลหลายกลุ่มพร้อมกัน โดยแบ่งแท่งในแต่ละกลุ่มเพื่อเปรียบเทียบค่าของแต่ละกลุ่มได้ชัดเจน

โค้ด	import numpy as np
	
	# ข้อมูลตัวอย่าง
	quarters = ['Q1', 'Q2', 'Q3', 'Q4']
	sales_A = [200, 250, 300, 220]
	sales_B = [150, 180, 250, 210]
	
	# สร้างตำแหน่งแท่ง
	bar_width = 0.35
	index = np.arange(len(quarters))
	
	# สร้าง Grouped Bar Chart
	plt.bar(index, sales_A, bar_width, label='Product A', color='blue')
	plt.bar(index + bar_width, sales_B, bar_width, label='Product B', color='green')
	
	plt.title('Quarterly sales of Product A and B')
	plt.xlabel('quarter')
	plt.ylabel('Sales (Baht)')
	plt.xticks(index + bar_width / 2, quarters)
	plt.legend()
	plt.show()

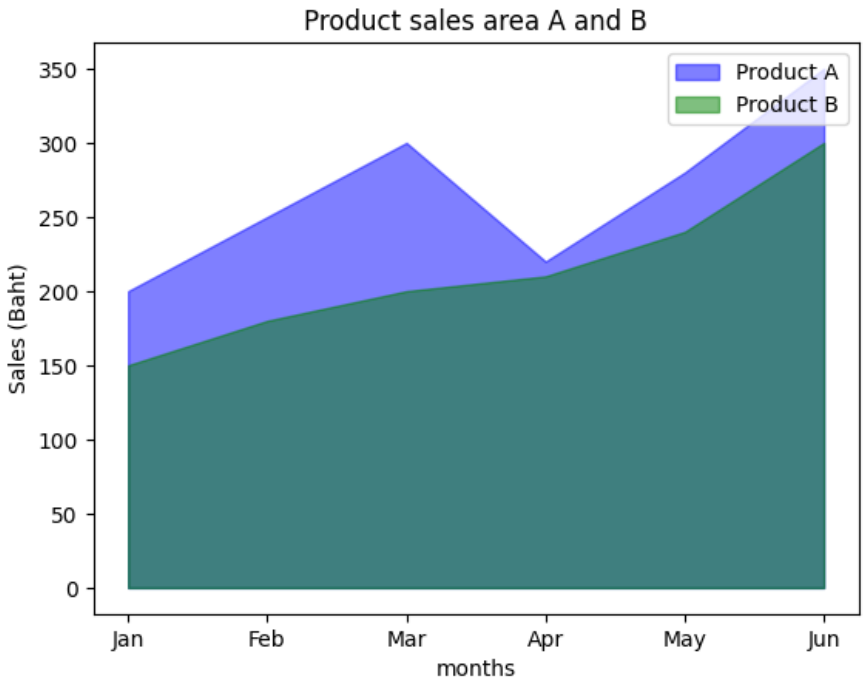
ผลลัพธ์	

ตัวอย่างที่ การเปรียบเทียบยอดขายของสินค้าสองชนิดในแต่ละไตรมาส โดยจัดแท่งให้อยู่ใกล้กันในแต่ละกลุ่มเพื่อการเปรียบเทียบ

2.3 กราฟพื้นที่

กราฟพื้นที่ (Area Chart) คือการแสดงข้อมูลเชิงเวลาเหมือนกราฟเชิงเส้น แต่เน้นการเติมพื้นที่ใต้เส้นเพื่อแสดงถึงปริมาณหรือจำนวนข้อมูล

โค้ด	Months = ['Jan','Feb','Mar','Apr','May','Jun']
	sales_A = [200, 250, 300, 220, 280,350]
	sales_B = [150, 180, 200, 210, 240,300]
	
	# สร้าง Area Chart
	plt.fill_between(months, sales_A, color='blue', alpha=0.5, label='Product A')
	plt.fill_between(months, sales_B, color='green', alpha=0.5, label='Product B')
	
	plt.title('Product sales area A and B')

	plt.xlabel('months') plt.ylabel('Sales (Baht)') plt.legend() plt.show()
ผลลัพธ์	

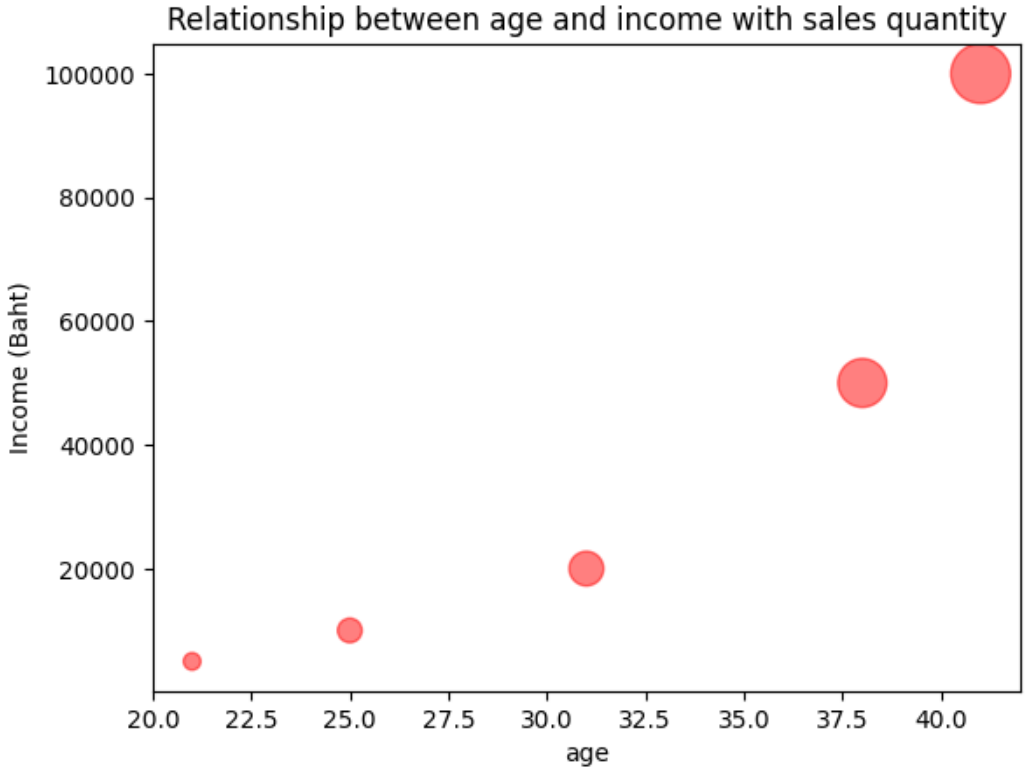
จากตัวอย่าง การเติมสีพื้นที่ใต้เส้นเพื่อแสดงปริมาณของยอดขายผลิตภัณฑ์ A และ B ในแต่ละเดือน

2.4 กราฟการกระจายแบบ Bubble Plot

กราฟการกระจายแบบ Bubble Plot เป็นการขยายจาก Scatter Plot โดยใช้ขนาดของฟอง (Bubble) แทนค่าตัวแปรที่สาม

โค้ด	import matplotlib.pyplot as plt
	# ข้อมูลตัวอย่าง
	age = [21, 25, 31, 38, 41]
	income = [5000, 10000, 20000, 50000, 100000]
	sizes = [50, 100, 200, 400, 600]

	# สร้าง Bubble Plot
	plt.scatter(age, income, s=sizes, color='red', alpha=0.5)
	plt.title('Relationship between age and income with sales quantity')
	plt.xlabel('age')
	plt.ylabel('Income (Baht)')
	plt.show()

ผลลัพธ์	

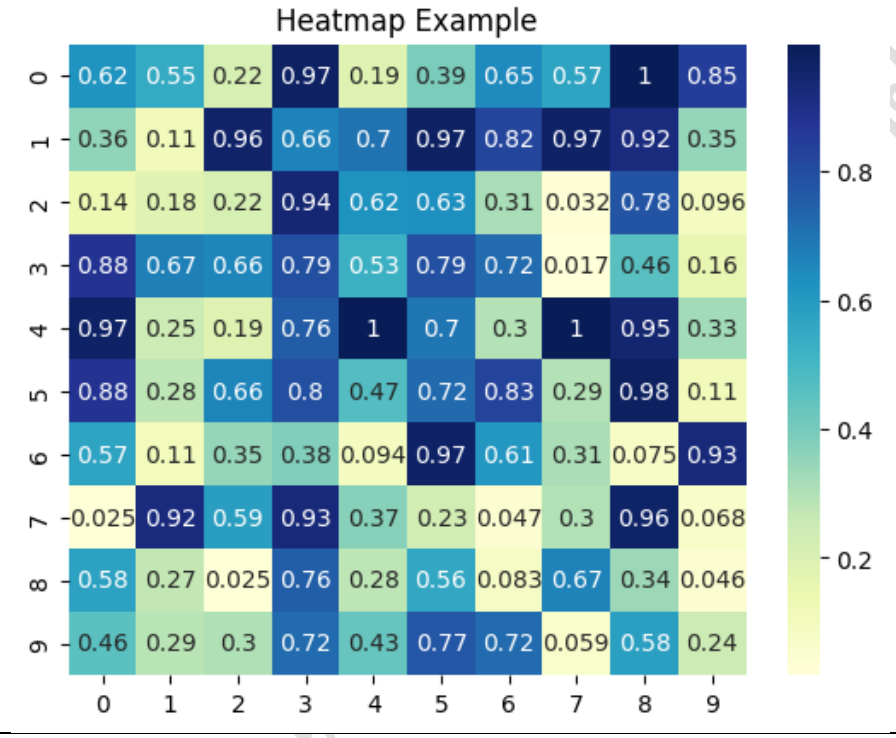
จากตัวอย่าง ขนาดของฟองแสดงถึงตัวแปรที่สาม เช่น จำนวนขาย แสดงถึงความสัมพันธ์ระหว่างตัวแปรสองตัวและปริมาณของตัวแปรที่สาม

2.5 ฮีทแมพ

ฮีทแมพ (Heatmap) ใช้ในการแสดงข้อมูลในรูปแบบตารางโดยใช้สีแทนค่าต่าง ๆ ทำให้สามารถแสดงความสัมพันธ์ระหว่างตัวแปรได้ชัดเจน

โค้ด	import seaborn as sns # ข้อมูลตัวอย่าง

	data = np.random.rand(10, 10)
	
	# สร้าง Heatmap
	sns.heatmap(data, annot=True, cmap='YlGnBu')
	plt.title('Heatmap Example')
	plt.show()

ผลลัพธ์	

ตัวอย่าง ตารางของข้อมูลที่มีสีแสดงถึงค่าต่าง ๆ โดยค่าที่สูงจะแสดงด้วยสีหนึ่ง และค่าที่ต่ำจะแสดงด้วยอีกสี

2.6 กราฟที่มีแกนสองด้าน

กราฟที่มีแกนสองด้าน (Dual-Axis Plot) ใช้ในการแสดงข้อมูลสองชุดที่มีหน่วยต่างกันบนกราฟเดียวกัน

โค้ด	# ข้อมูลตัวอย่าง
	temperature = [30, 32, 33, 35, 36, 37]
	rainfall = [100, 95, 80, 70, 50, 30]

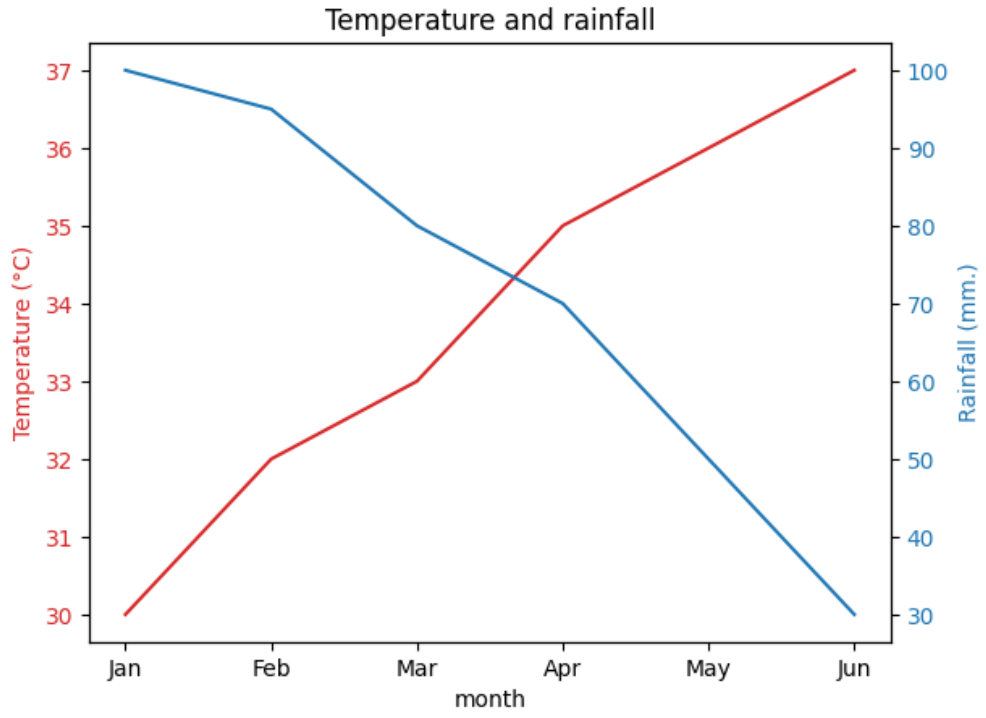
```
# สร้างกราฟแกนซ้าย
fig, ax1 = plt.subplots()

ax1.set_xlabel('month')
ax1.set_ylabel('Temperature (°C)', color='tab:red')
ax1.plot(months, temperature, color='tab:red')
ax1.tick_params(axis='y', labelcolor='tab:red')

# สร้างแกนขวา
ax2 = ax1.twinx()
ax2.set_ylabel('Rainfall (mm.)', color='tab:blue')
ax2.plot(months, rainfall, color='tab:blue')
ax2.tick_params(axis='y', labelcolor='tab:blue')

plt.title('Temperature and rainfall')
plt.show()
```

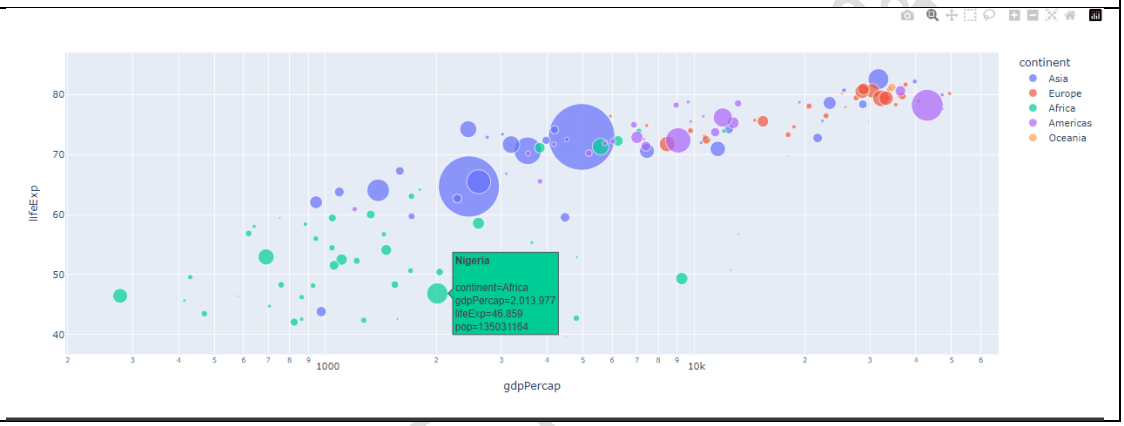
ผลลัพธ์



ตัวอย่าง แกน Y สองด้านแสดงข้อมูลสองประเภทที่มีหน่วยต่างกัน เช่น อุณหภูมิและปริมาณน้ำฝน

2.7 กราฟโต้ตอบ

กราฟโต้ตอบ (Interactive Plot) สามารถเลื่อนดู ซูมเข้า-ออก และทำการวิเคราะห์เชิงลึกได้ โดยใช้ไลบรารี Plotly หรือ Bokeh

โค้ด	
ผลลัพธ์	

ตัวอย่าง กราฟนี้สามารถเลื่อน ซูม และโต้ตอบได้ ซึ่งช่วยในการทำงานกับข้อมูลขนาดใหญ่

2.8 กราฟแบบ Subplots

การสร้างกราฟแบบ Subplots ใช้สำหรับการวางกราฟหลายกราฟในกรอบเดียวกันเพื่อให้สามารถเปรียบเทียบข้อมูลได้อย่างมีประสิทธิภาพ

โค้ด	import matplotlib.pyplot as plt
	
	# ข้อมูลตัวอย่าง
	months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
	sales_A = [200, 250, 300, 220, 280, 350]
	sales_B = [150, 180, 200, 210, 240, 300]
	
	# สร้าง Subplots

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))  # 1 แถว 2 กราฟ

# กราฟเชิงเส้น (Line Chart) ใน Subplot แรก
ax1.plot(months, sales_A, marker='o', color='blue', label='Product A')
ax1.plot(months, sales_B, marker='o', color='green', label='Product B')
ax1.set_title('linear sales')
ax1.set_xlabel('month')
ax1.set_ylabel('Sales (Baht)')
ax1.legend()

# กราฟแท่ง (Bar Chart) ใน Subplot ที่สอง
ax2.bar(months, sales_A, color='blue', label='Product A')
ax2.bar(months, sales_B, bottom=sales_A, color='green', label='Product B')  # ซ้อนแท่ง
ax2.set_title('Bar chart sales')
ax2.set_xlabel('month')
ax2.set_ylabel('Sales (Baht)')
ax2.legend()

# แสดงกราฟ
plt.tight_layout()
plt.show()
```

ผลลัพธ์