วิชาเหมืองข้อมูล (Data Mining)

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

Asst. Prof. Dr. Nattapong Songneam

Department of Computer Science,
Faculty of Science and Technology,
Phranakhon Rajabhat University

http://www.siam2dev.com

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

การจำแนกข้อมูล (Classification) เป็นกระบวนการที่ใช้เทคนิคต่าง ๆ เพื่อจัดกลุ่มข้อมูลให้อยู่ในหมวดหมู่หรือกลุ่มที่ถูกต้องตามลักษณะหรือลักษณะที่กำหนดไว้ล่วงหน้า ซึ่งการจำแนกข้อมูลมีบทบาทสำคัญในการทำนายผลลัพธ์หรือการตัดสินใจในหลาย ๆ ด้านของวิทยาการคณิตศาสตร์, วิทยาศาสตร์ข้อมูล, และปัญญาประดิษฐ์ ดังนั้น, การใช้เทคนิคการจำแนกข้อมูลที่เหมาะสมจึงเป็นสิ่งสำคัญ เพื่อให้ได้ผลลัพธ์ที่แม่นยำและมีประสิทธิภาพสูงสุด

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

ทบทวน

- ข้อมูล
- การเตรียมข้อมูล

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

เทคนิคเชิงสถิติ (Statistical Techniques) ในงานเหมืองข้อมูลคือ เทคนิคเชิงสถิติในงานเหมืองข้อมูลมีบทบาทสำคัญในการวิเคราะห์และอธิบายข้อมูล โดยเฉพาะในกระบวนการตรวจสอบความสัมพันธ์, การทำนาย, และการทำนายที่ทำในทางทฤษฎีสถิติ. นี่คือเทคนิคเชิงสถิติบางประการที่มักถูกใช้ในงานเหมืองข้อมูล:

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

ในบทนี้ เราจะกล่าวถึงเทคนิคเชิงสถิติที่ใช้ในการสร้างโมเดลเหมืองข้อมูล โดยเทคนิคเหล่านี้จะใช้ในการวิเคราะห์ข้อมูลและค้นหารูปแบบความสัมพันธ์ที่ซ่อนอยู่ในข้อมูล ซึ่งรูปแบบความสัมพันธ์เหล่านี้สามารถนำไปใช้ในการทำนายหรือการตัดสินใจได้

เทคนิคเชิงสถิติที่นิยมใช้ในการสร้างโมเดลเหมืองข้อมูล

1. การวิเคราะห์ความถี่ (Frequency Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาค่าความถี่ของข้อมูลแต่ละค่า โดยค่าความถี่ของข้อมูลแต่ละค่า หมายถึง จำนวนครั้งที่ข้อมูลนั้นปรากฏในข้อมูลทั้งหมด

2. การวิเคราะห์การกระจาย (Distribution Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหารูปแบบการกระจายของข้อมูล โดยรูปแบบการกระจายของข้อมูล หมายถึง ลักษณะการกระจายของข้อมูล เช่น การกระจายแบบปกติ การกระจายแบบเชิงเส้น การกระจายแบบทวินาม เป็นต้น

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

เทคนิคเชิงสถิติที่นิยมใช้ในการสร้างโมเดลเหมืองข้อมูล

3. การวิเคราะห์ความสัมพันธ์ (Relational Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาความสัมพันธ์ระหว่างข้อมูลสองชุดหรือมากกว่า โดยความสัมพันธ์ระหว่างข้อมูลสองชุดหรือมากกว่า หมายถึง ลักษณะความสัมพันธ์ระหว่างข้อมูลทั้งสองชุดหรือมากกว่า เช่น ความสัมพันธ์แบบเชิงเส้น ความสัมพันธ์แบบเชิงพหุ เป็นต้น

4. การวิเคราะห์การแปรปรวน (Variation Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาความแปรปรวนของข้อมูล โดยความแปรปรวนของข้อมูล หมายถึง ระดับการเปลี่ยนแปลงของข้อมูล

5. การวิเคราะห์แนวโน้ม (Trend Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาแนวโน้มของข้อมูล โดยแนวโน้มของข้อมูล หมายถึง ทิศทางการเปลี่ยนแปลงของข้อมูล

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1 การวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์ความถี่ (Frequency Analysis) คือกระบวนการที่ใช้สถิติเพื่อวิเคราะห์และแสดงข้อมูลตามความถี่ของค่าต่างๆ ที่ปรากฏในชุดข้อมูล. การวิเคราะห์ความถี่มีประโยชน์ในการทำความเข้าใจและสรุปลักษณะของข้อมูลที่มีอยู่. นี่คือขั้นตอนพื้นฐานของการวิเคราะห์ความถี่:

การวิเคราะห์ความถี่ (Frequency Analysis) เป็นเทคนิคการวิเคราะห์ข้อมูลเพื่อหาค่าความถี่ของข้อมูลแต่ละค่า โดยค่าความถี่ของข้อมูลแต่ละค่า หมายถึง จำนวนครั้งที่ข้อมูลนั้นปรากฏในข้อมูลทั้งหมด

การวิเคราะห์ความถี่เป็นเทคนิคการวิเคราะห์ข้อมูลขั้นพื้นฐานที่ใช้ในการอธิบายลักษณะทั่วไปของข้อมูล โดยค่าความถี่ของข้อมูลแต่ละค่าสามารถช่วยให้เราเข้าใจว่าข้อมูลส่วนใหญ่มีค่าอยู่ที่ใด มีค่าที่ผิดปกติหรือไม่ และนอกจากนี้ยังเป็นการจัดเตรียมข้อมูลเพื่อความสะดวกในการวิเคราะห์ข้อมูลขั้นต่อไป

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1 การวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์ความถี่ (Frequency Analysis) คือกระบวนการที่ใช้สถิติเพื่อวิเคราะห์และแสดงข้อมูลตามความถี่ของค่าต่างๆ ที่ปรากฏในชุดข้อมูล. การวิเคราะห์ความถี่มีประโยชน์ในการทำความเข้าใจและสรุปลักษณะของข้อมูลที่มีอยู่. นี่คือขั้นตอนพื้นฐานของการวิเคราะห์ความถี่:

การวิเคราะห์ความถี่สามารถใช้ได้กับข้อมูลประเภทตัวเลขหรือข้อมูลประเภทข้อความ โดยข้อมูลประเภทตัวเลขสามารถหาค่าความถี่ได้โดยแบ่งข้อมูลออกเป็นช่วงๆ แล้วนับจำนวนข้อมูลในแต่ละช่วง ตัวอย่างเช่น ข้อมูลคะแนนสอบสามารถแบ่งออกเป็นช่วงๆ เช่น คะแนน 0-10, คะแนน 11-20, คะแนน 21-30 เป็นต้น จากนั้นนับจำนวนนักเรียนที่มีคะแนนในแต่ละช่วง

ข้อมูลประเภทข้อความสามารถหาค่าความถี่ได้โดยแบ่งข้อมูลออกเป็นกลุ่มๆ แล้วนับจำนวนข้อมูลในแต่ละกลุ่ม ตัวอย่างเช่น ข้อมูลประเภทเพศสามารถแบ่งออกเป็นกลุ่มชายและหญิง จากนั้นนับจำนวนข้อมูลในแต่ละกลุ่ม

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.1 ตัวอย่างการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่างการใช้การวิเคราะห์ความถี่ ได้แก่

1) การวิเคราะห์พฤติกรรมการซื้อของผู้บริโภค เช่น สินค้าใดที่ผู้บริโภคนิยมซื้อมากที่สุด สินค้าใดที่ผู้บริโภคซื้อบ่อยที่สุด เป็นต้น

2) การวิเคราะห์รูปแบบการกระจายของคะแนนสอบ เช่น คะแนนสอบส่วนใหญ่อยู่ในช่วงใด คะแนนสอบกระจายตัวมากน้อยเพียงใด เป็นต้น

3) การวิเคราะห์ความแปรปรวนของราคาสินค้า เช่น ราคาสินค้าแปรปรวนมากน้อยเพียงใด เป็นต้น

การวิเคราะห์ความถี่เป็นเทคนิคการวิเคราะห์ข้อมูลพื้นฐานที่มีความสำคัญและมีประโยชน์ในการวิเคราะห์ข้อมูลต่างๆ

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

1) การสร้างตารางความถี่ (Frequency Table):
สร้างตารางที่แสดงค่าทั้งหมดที่ปรากฏในชุดข้อมูลและจำนวนครั้งที่เกิดขึ้นสำหรับแต่ละค่า.

2) การสร้างกราฟความถี่ (Frequency Graph):
สร้างกราฟที่แสดงความถี่ของแต่ละค่า. กราฟที่บ่งบอกความถี่ส่วนแบ่งของข้อมูลได้แก่ Histogram, Bar Chart, Pie Chart เป็นต้น.

3) การคำนวณค่าสถิติเบื้องต้น:
คำนวณค่าเฉลี่ย (mean), ค่ามัธยฐาน (median), ค่าฐานนิยม (mode), ค่าความแปรปรวน (variance), และค่าเบี่ยงเบนมาตรฐาน (standard deviation) เพื่อให้ได้ข้อมูลเพิ่มเติมเกี่ยวกับการกระจายของข้อมูล.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

4) การทำนายและการวิเคราะห์ทางสถิติ:
หากข้อมูลมีลักษณะการกระจายที่เป็นรูปแบบ (distribution) บางแบบ, การวิเคราะห์ความถี่สามารถช่วยในการทำนายและวิเคราะห์ข้อมูลเพิ่มเติม.

5) การวิเคราะห์และเปรียบเทียบความถี่:
เปรียบเทียบความถี่ของค่าต่างๆ ในกลุ่มหรือสองชุดข้อมูลเพื่อดูความแตกต่าง.

6) การแปลผลลัพธ์:
ให้ข้อมูลที่สามารถใช้ในการตีความหรือสรุปสิ่งที่เราได้วิเคราะห์.

การวิเคราะห์ความถี่มีประโยชน์มากในการทำความเข้าใจลักษณะของข้อมูล, การตรวจสอบความสมมติฐาน, และการสร้างรายงานสรุปข้อมูล

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

เพื่อสร้างตารางความถี่ (Frequency Table), ลองพิจารณาตัวอย่างดังนี้:

สมมุติว่าเรามีชุดข้อมูลต่อไปนี้ซึ่งแสดงคะแนนการทดสอบของนักเรียนในรายวิชาวิทยาศาสตร์:

คะแนน: [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]

เราสามารถสร้างตารางความถี่ได้ดังนี้:

1. นับความถี่ของแต่ละคะแนน:
   - 75: จำนวน 3 ครั้ง
   - 82: จำนวน 4 ครั้ง
   - 88: จำนวน 4 ครั้ง
   - 92: จำนวน 3 ครั้ง
   - 95: จำนวน 1 ครั้ง

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

2. สร้างตารางความถี่:

คะแนน	จำนวนครั้ง
75	3
82	4
88	4
92	3
95	1

ในตารางนี้, เราแสดงคะแนนและจำนวนครั้งที่แต่ละคะแนนปรากฏในชุดข้อมูล. นอกจากนี้, สามารถเพิ่มคอลัมน์เพื่อแสดงความถี่สะสม (cumulative frequency) หรือความถี่สัมพันธ์ (relative frequency) ได้ตามความต้องการของการวิเคราะห์.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่าง การสร้างตารางความถี่ (Frequency Table): ในภาษา python : ในภาษา Python, คุณสามารถใช้ไลบรารี Pandas เพื่อสร้างตารางความถี่ได้ง่ายดาย. นี่คือตัวอย่างการสร้างตารางความถี่โดยใช้ Pandas:

```
import pandas as pd

# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้าง DataFrame จากคะแนน
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']
# แสดงตารางความถี่
print(frequency_table)
```

ตัวอย่างที่ 4.1 ชื่อไฟล์ : dm_frequency_table.py

ในตัวอย่างนี้, เราใช้ pd.DataFrame เพื่อสร้าง DataFrame จากข้อมูลคะแนน. จากนั้น, เราใช้ value_counts() เพื่อนับความถี่ของแต่ละคะแนนและ reset_index() เพื่อเปลี่ยน index เป็นคอลัมน์. ในท้ายที่สุด, เรากำหนดชื่อคอลัมน์ใหม่และแสดงตารางความถี่.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่าง การสร้างตารางความถี่ (Frequency Table): ในภาษา python : ในภาษา Python, คุณสามารถใช้ไลบรารี Pandas เพื่อสร้างตารางความถี่ได้ง่ายดาย. นี่คือตัวอย่างการสร้างตารางความถี่โดยใช้ Pandas:

ตัวอย่างที่ 4.1 ชื่อไฟล์ : dm_frequency_table.py

```
import pandas as pd

# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้าง DataFrame จากคะแนน
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']
# แสดงตารางความถี่
print(frequency_table)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\AppServ\www\Python_DataMining> & C:/Us
   คะแนน  จำนวนครั้ง
0     82          4
1     75          3
2     88          3
3     92          3
4     95          1
PS C:\AppServ\www\Python_DataMining>
```

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่าง การสร้างตารางความถี่ (Frequency Table): ในภาษา python : ในภาษา Python, คุณสามารถใช้ไลบรารี Pandas เพื่อสร้างตารางความถี่ได้ง่ายดาย. นี่คือตัวอย่างการสร้างตารางความถี่โดยใช้ Pandas:

ตัวอย่างที่ 4.2 ชื่อไฟล์ : dm_frequency_table_graph.py

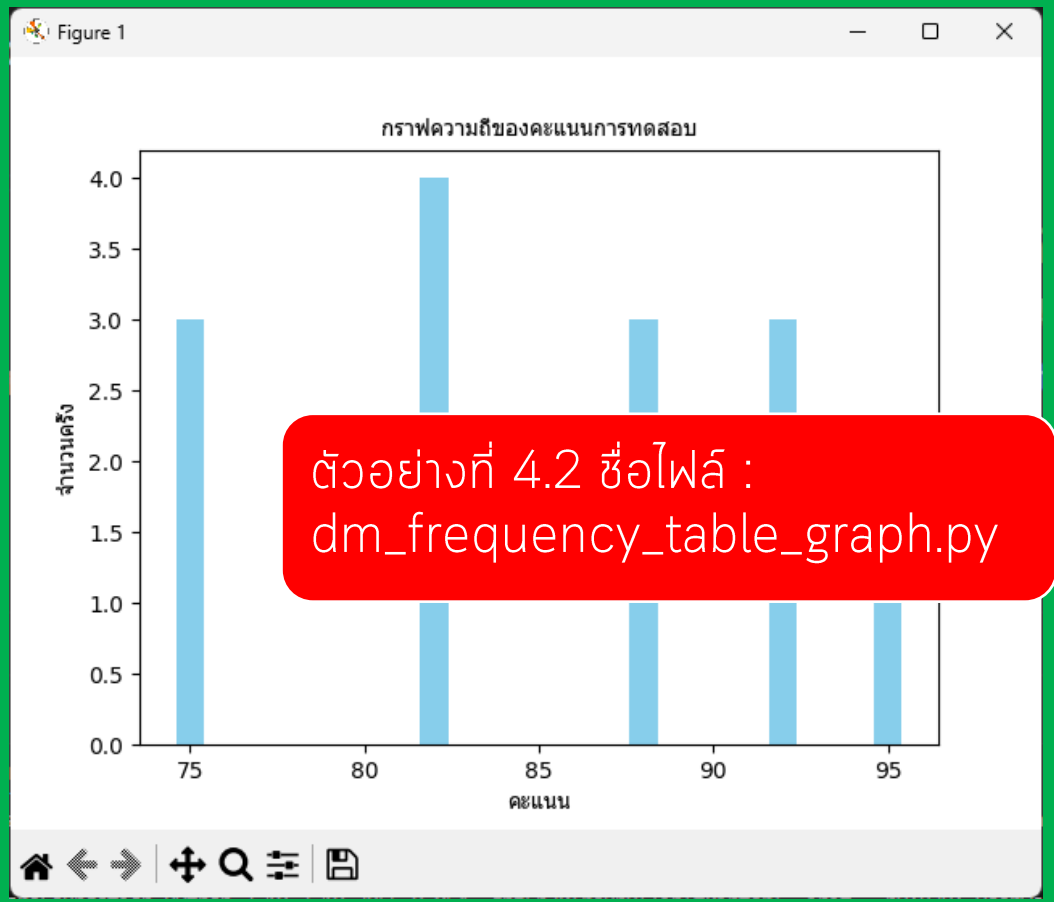
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager
# กำหนดฟอนต์ภาษาไทย (ในที่นี้ใช้ Tahoma)
font_thai = font_manager.FontProperties(fname="C:\\Windows\\Fonts\\tahoma.ttf")
# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้าง DataFrame จากคะแนน
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']
# สร้างกราฟความถี่
plt.bar(frequency_table['คะแนน'], frequency_table['จำนวนครั้ง'], color='skyblue')
# กำหนดรายละเอียดกราฟ
plt.xlabel('คะแนน', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.ylabel('จำนวนครั้ง', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.title('กราฟความถี่ของคะแนนการทดสอบ', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
# แสดงกราฟ
plt.show()
```

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่าง การสร้างตารางความถี่ (Frequency Table): ในภาษา python : ในภาษา Python, คุณสามารถใช้ไลบรารี Pandas เพื่อสร้างตารางความถี่ได้ง่ายดาย. นี่คือตัวอย่างการสร้างตารางความถี่โดยใช้ Pandas:

ในที่นี้, ให้เปลี่ยน font_thai เป็น "C:\Windows\Fonts\tahoma.ttf" หรือที่มีอยู่ในระบบของคุณ. ถ้าการใช้ "Tahoma" ทำงานถูกต้อง, แสดงว่าปัญหามีต้นเหตุที่พาธของฟอนต์.

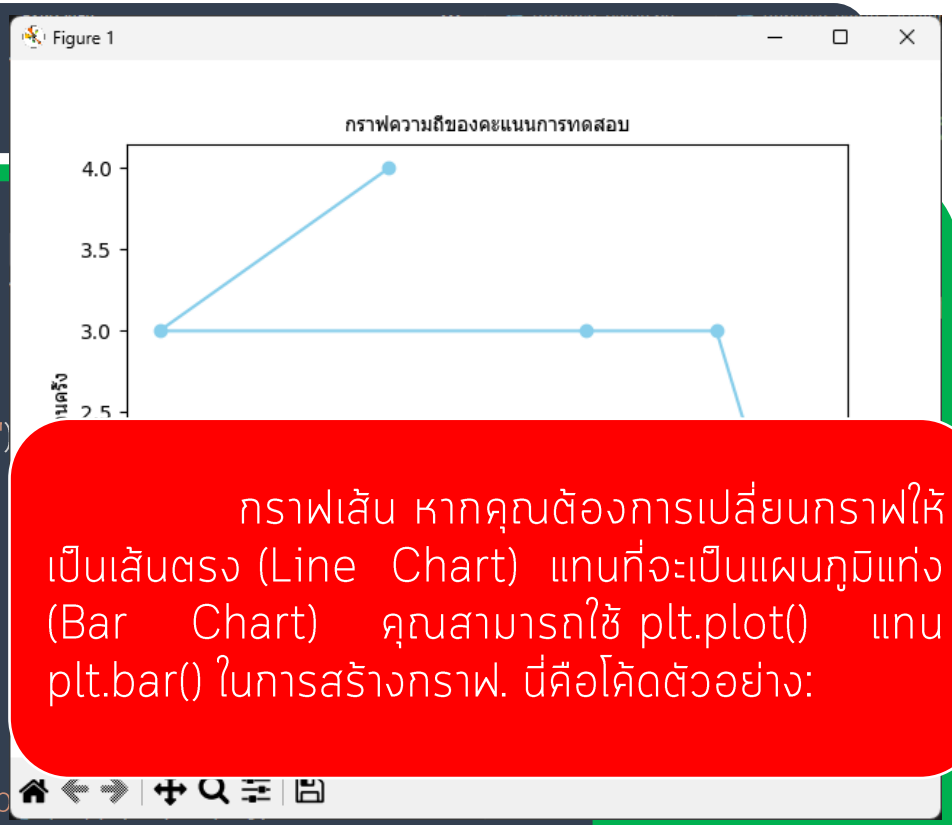


ตัวอย่างที่ 4.2 ชื่อไฟล์ : dm_frequency_table_graph.py

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager

# กำหนดฟอนต์ภาษาไทย (ในที่นี้ใช้ Tahoma)
font_thai = font_manager.FontProperties(fname="C:\\Windows\\Fonts\\tahoma.ttf")
# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้าง DataFrame จากคะแนน
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']
# สร้างกราฟเส้นตรง
plt.plot(frequency_table['คะแนน'], frequency_table['จำนวนครั้ง'], marker='o', color='skyb
# กำหนดรายละเอียดกราฟ
plt.xlabel('คะแนน', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.ylabel('จำนวนครั้ง', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.title('กราฟความถี่ของคะแนนการทดสอบ', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบ
# แสดงกราฟ
plt.show()
```



กราฟเส้น หากคุณต้องการเปลี่ยนกราฟให้เป็นเส้นตรง (Line Chart) แทนที่จะเป็นแผนภูมิแท่ง (Bar Chart) คุณสามารถใช้ plt.plot() แทน plt.bar() ในการสร้างกราฟ. นี่คือโค้ดตัวอย่าง:

ตัวอย่างที่ 4.3 ชื่อไฟล์ : dm_frequency_table_LineGraph.py

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

การคำนวณค่าสถิติเบื้องต้น:
คำนวณค่าเฉลี่ย (mean), ค่ามัธยฐาน (median), ค่าฐานนิยม (mode), ค่าความแปรปรวน (variance), และค่าเบี่ยงเบนมาตรฐาน (standard deviation) เพื่อให้ได้ข้อมูลเพิ่มเติมเกี่ยวกับการกระจายของข้อมูล.

การคำนวณค่าสถิติเบื้องต้น, มักจะพูดถึงค่าเฉลี่ย (mean), ค่ามัธยฐาน (median), และค่าฐานนิยม (mode) ซึ่งเป็นค่าสถิติที่ใช้บ่อย. นี่คือตัวอย่างการคำนวณค่าสถิติเบื้องต้นโดยใช้ภาษา Python และไลบรารี NumPy:

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

```python
import numpy as np

# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]

# คำนวณค่าเฉลี่ย (mean)
mean_score = np.mean(scores)
print(f'ค่าเฉลี่ย: {mean_score}')

# คำนวณค่ามัธยฐาน (median)
median_score = np.median(scores)
print(f'ค่ามัธยฐาน: {median_score}')

# คำนวณค่าฐานนิยม (mode)
mode_score = np.argmax(np.bincount(scores))
print(f'ค่าฐานนิยม: {mode_score}')
```

ตัวอย่างที่ 4.4 ชื่อไฟล์ : dm_basic_statistic_numpy.py

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์และเปรียบเทียบความถี่ :

การวิเคราะห์และเปรียบเทียบความถี่ (Frequency Analysis) เป็นกระบวนการที่ใช้เพื่อทำความเข้าใจและแสดงข้อมูลที่มีการกระจายตัวต่ำสูง, ค่าที่ปรากฏบ่อย, และลักษณะทางสถิติของข้อมูล. นี่คือบางขั้นตอนและเทคนิคที่ใช้ในการวิเคราะห์และเปรียบเทียบความถี่:

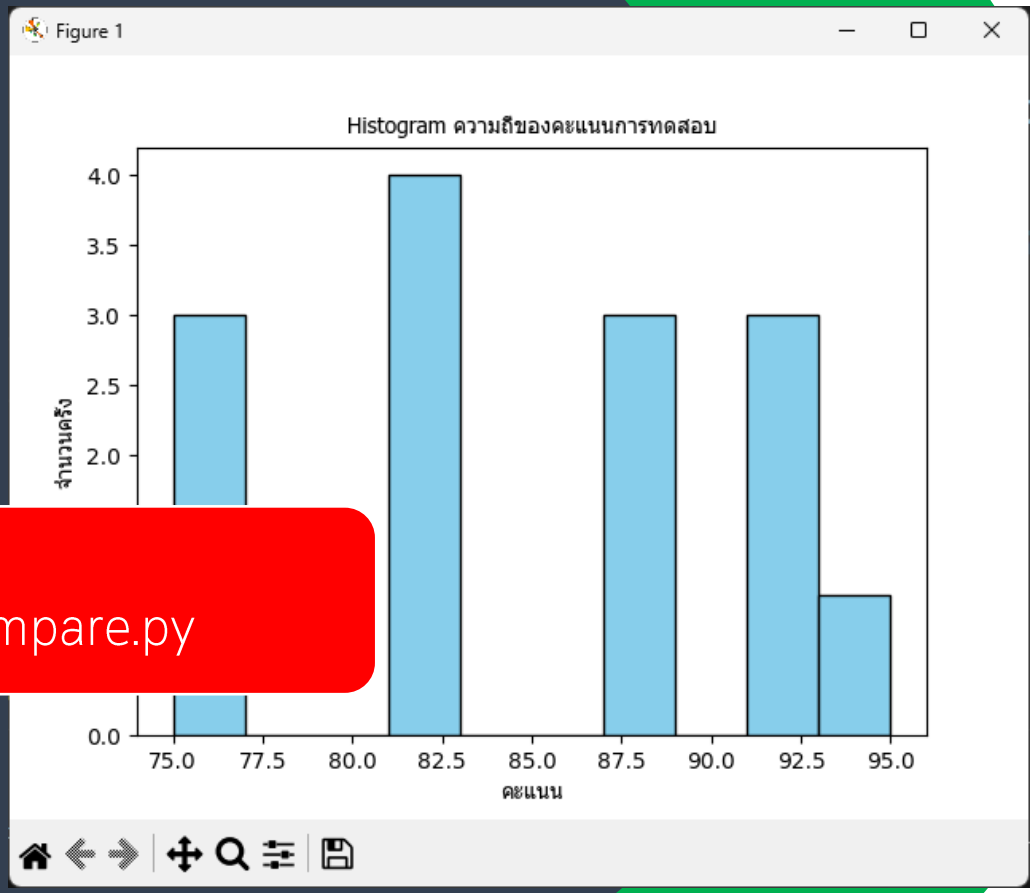
1. สร้างตารางความถี่ (Frequency Table)
2. สร้างกราฟความถี่ (Histogram)
3. วิเคราะห์และเปรียบเทียบ
   - -ค่าเฉลี่ย (Mean):
   - -ค่ามัธยฐาน (Median):
   - -ค่าฐานนิยม (Mode):
   - -ค่าความแปรปรวน (Variance):
   - -ค่าส่วนเบี่ยงมาตรฐาน (Standard Deviation):

การวิเคราะห์และเปรียบเทียบความถี่ช่วยให้เราทราบถึงแนวโน้มของข้อมูล, ค่าสถิติทางสถิติของข้อมูล, และการกระจายตัวของข้อมูล.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

```python
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import font_manager
# กำหนดฟอนต์ภาษาไทย (ในที่นี้ใช้ Tahoma)
font_thai = font_manager.FontProperties(fname="C:\\Windows\\Fonts\\tahoma.ttf")
# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้างกราฟความถี่
plt.hist(scores, bins=10, color='skyblue', edgecolor='black')
# กำหนดรายละเอียดกราฟ
plt.xlabel('คะแนน', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.ylabel('จำนวนครั้ง', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
plt.title('Histogram ความถี่ของคะแนนการทดสอบ', fontproperties=font_thai)  # กำหนดฟอนต์ในรูปแบบภาษาไทย
# แสดงกราฟ
plt.show()
# คำนวณค่าสถิติ
mean_score = pd.Series(scores).mean()
median_score = pd.Series(scores).median()
mode_score = pd.Series(scores).mode().iloc[0]
variance_score = pd.Series(scores).var()
std_dev_score = pd.Series(scores).std()

# แสดงค่าสถิติ
print(f'ค่าเฉลี่ย: {mean_score}')
print(f'ค่ามัธยฐาน: {median_score}')
print(f'ค่าฐานนิยม: {mode_score}')
print(f'ค่าความแปรปรวน: {variance_score}')
print(f'ค่าส่วนเบี่ยงมาตรฐาน: {std_dev_score}')
```



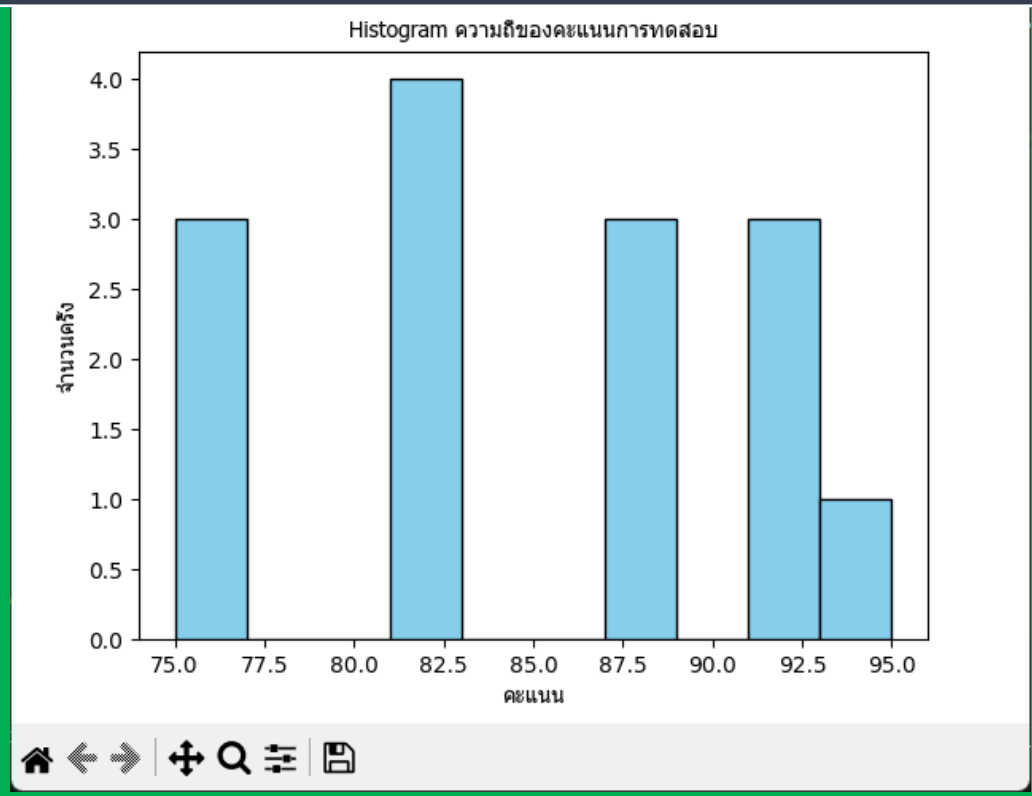
ตัวอย่างที่ 4.5 ชื่อไฟล์ : dm_statistic_numpy_compare.py

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

โปรแกรมนี้จะแสดงกราฟความถี่และค่าสถิติของคะแนนการทดสอบ. ในบางกรณี, คำสั่ง pd.Series(scores) ถูกเพิ่มเข้าไปเพื่อให้ Pandas รู้จักกับข้อมูลของคุณ

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINA

PS C:\AppServ\www\Python_DataMining> & C:/
ค่าเฉลี่ย: 84.85714285714286
ค่ามัธยฐาน: 85.0
ค่าฐานนิยม: 82
ค่าความแปรปรวน: 46.90109890109989
ค่าส่วนเบี่ยงมาตรฐาน: 6.848437697832908
PS C:\AppServ\www\Python_DataMining>
```



ตัวอย่างที่ 4.5 ชื่อไฟล์ : dm_statistic_numpy_compare.py

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์และการแปลผลลัพธ์ในแง่มุมของ Skewness, Kurtosis, และ Outliers มีความสำคัญในการเข้าใจลักษณะการกระจายของข้อมูล:

1. Skewness (การดูทิศทางของการกระจาย):
   - Positive Skewness (เบ้หวาน): หาก Skewness เป็นบวก, นั้นหมายถึงข้อมูลมีการกระจายที่มีค่ามากไปทางขวามือของกราฟ. ค่าเฉลี่ยมีแนวโน้มที่มากกว่าค่ามัธยฐาน.
   - Negative Skewness (เบ้สมมติ): ถ้า Skewness เป็นลบ, นั้นหมายถึงข้อมูลมีการกระจายที่มีค่ามากไปทางซ้ายมือของกราฟ. ค่าเฉลี่ยมีแนวโน้มที่น้อยกว่าค่ามัธยฐาน.

2. Kurtosis (การวัดการกระจายตัว):
   - **Leptokurtic (Kurtosis > 3):** มีการกระจายตัวมากขึ้นที่หลังจากมีค่าสูง (ความกระชับของกราฟมาก).
   - **Mesokurtic (Kurtosis = 3):** มีการกระจายตัวเท่ากับกราฟปกติ (ความกระชับของกราฟเท่ากับกราฟปกติ).
   - **Platykurtic (Kurtosis < 3):** มีการกระจายตัวน้อยลงที่หลังจากมีค่าสูง (ความกระชับของกราฟน้อย).

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์และการแปลผลลัพธ์ในแง่มุมของ Skewness, Kurtosis, และ Outliers มีความสำคัญในการเข้าใจลักษณะการกระจายของข้อมูล:

การแปลผลลัพธ์:

- หาก Skewness เป็นบวก: ข้อมูลมีแนวโน้มไปทางขวามือของกราฟ.
- หาก Skewness เป็นลบ: ข้อมูลมีแนวโน้มไปทางซ้ายมือของกราฟ.
- หาก Kurtosis > 3: ข้อมูลมีการกระจายตัวมากขึ้นหลังจากมีค่าสูง.
- หาก Kurtosis < 3: ข้อมูลมีการกระจายตัวน้อยลงที่หลังจากมีค่าสูง.
- การพิจารณา Outliers ช่วยในการตรวจสอบข้อมูลที่อาจต้องการการตรวจสอบเพิ่มเติม.

การวิเคราะห์เหล่านี้ช่วยให้ผู้วิเคราะห์เข้าใจลักษณะของข้อมูลและการกระจายตัวของข้อมูล, ซึ่งเป็นประโยชน์ในการตัดสินใจและการวางแผน.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่างการวิเคราะห์และการแปลผลลัพธ์ในแง่มุมของ Skewness, Kurtosis, และ Outliers, เราจะใช้ไลบรารี Pandas และ Seaborn ใน Python. ตัวอย่างนี้จะใช้ข้อมูลที่สร้างจากการสุ่มแบบปกติ (normal distribution) และการเพิ่ม Outliers:

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ในตัวอย่างนี้:

เราสร้างข้อมูลที่สุ่มแบบปกติด้วย numpy.random.normal และเพิ่ม Outliers.
พล็อต Histogram เพื่อแสดงการกระจายของข้อมูล.
คำนวณและแสดงผล Skewness และ Kurtosis.
พล็อต Boxplot เพื่อแสดง Outliers ที่อาจจะปรากฏ.

ผลลัพธ์:

Skewness บวกแสดงถึงการเบ้หวานของข้อมูล.
Kurtosis มากกว่า 3 แสดงถึงลักษณะ Leptokurtic ที่มีการกระจายตัวมากขึ้นหลังจากมีค่าสูง.
จาก Boxplot และ Histogram แสดงให้เห็น Outliers ที่มีค่าที่ต่างจากค่าปกติมาก.

การวิเคราะห์และการแปลผลลัพธ์ช่วยให้เราเข้าใจลักษณะของข้อมูลและปรับตัวแบบการวิเคราะห์ต่อไป.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

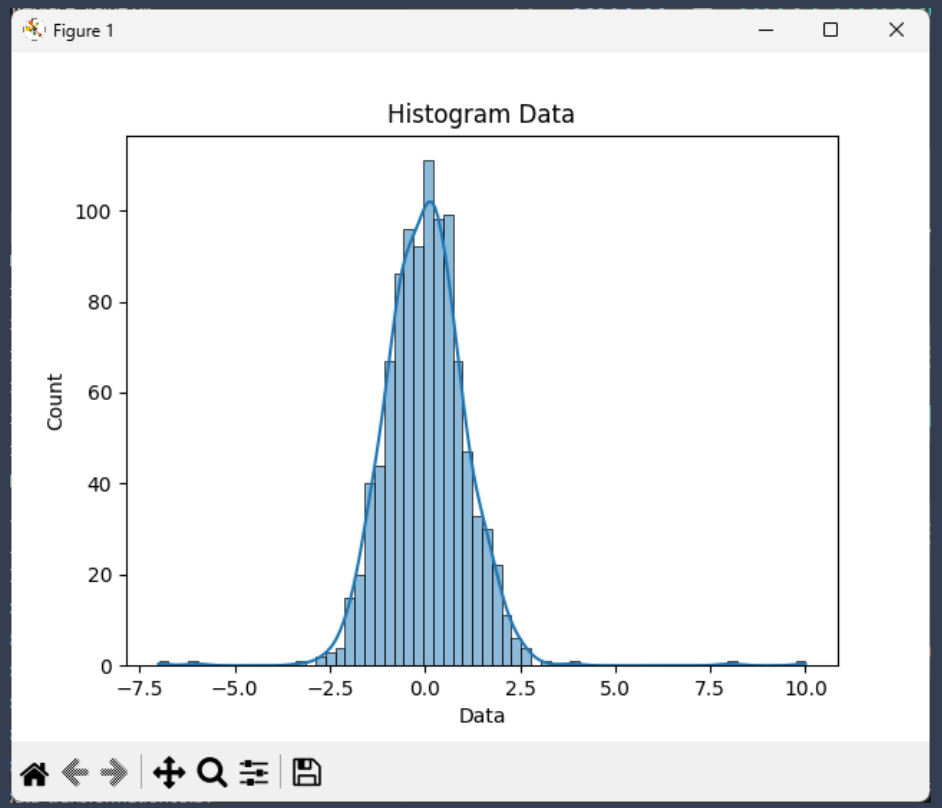
ตัวอย่างที่ 4.6 ชื่อไฟล์ : dm_statistic_numpy_compare.py

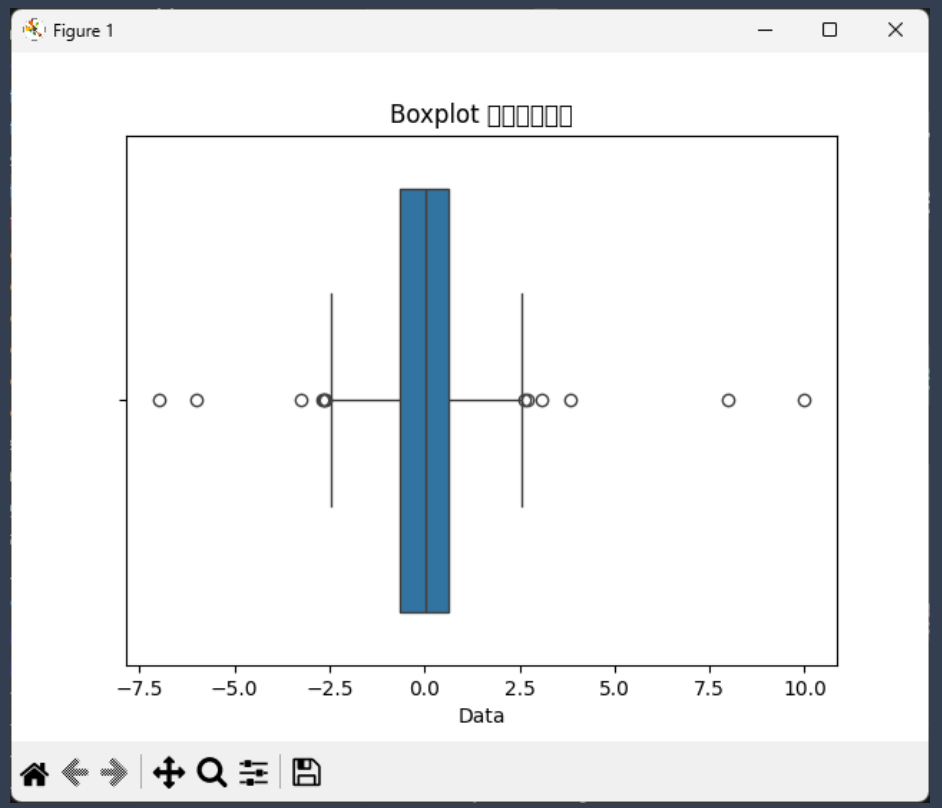
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# สร้างข้อมูลที่สุ่มแบบปกติ
np.random.seed(42)
data = np.random.normal(loc=0, scale=1, size=1000)
# เพิ่ม Outliers
outliers = np.array([8, 10, -7, -6])
data = np.concatenate([data, outliers])
# สร้าง DataFrame
df = pd.DataFrame({'Data': data})
# พล็อต Histogram
sns.histplot(df['Data'], kde=True)
plt.title('Histogram Data')
# แสดงกราฟ
plt.show()
```

```
# คำนวณ Skewness และ Kurtosis
skewness = df['Data'].skew()
kurtosis = df['Data'].kurtosis()
# แสดงผล Skewness และ Kurtosis
print(f'Skewness: {skewness}')
print(f'Kurtosis: {kurtosis}')
# แสดงตารางความถี่
frequency_table = pd.cut(df['Data'],
bins=20).value_counts().sort_index().reset_index()
frequency_table.columns = ['ช่วง', 'จำนวน']
print('\nตารางความถี่:')
print(frequency_table)
# พล็อต Boxplot เพื่อแสดง Outliers
sns.boxplot(x=df['Data'])
plt.title('Boxplot ข้อมูล')
# แสดงกราฟ
plt.show()
```

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)





บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

ในตัวอย่างนี้:

เราสร้างข้อมูลที่สุ่มแบบปกติด้วย numpy.random.normal และเพิ่ม Outliers.
พล็อต Histogram เพื่อแสดงการกระจายของข้อมูล.
คำนวณและแสดงผล Skewness และ Kurtosis.
พล็อต Boxplot เพื่อแสดง Outliers ที่อาจจะปรากฏ.

ผลลัพธ์:

Skewness บวกแสดงถึงการเบ้หวานของข้อมูล.
Kurtosis มากกว่า 3 แสดงถึงลักษณะ Leptokurtic ที่มีการกระจายตัวมากขึ้นหลังจากมีค่าสูง.
จาก Boxplot และ Histogram แสดงให้เห็น Outliers ที่มีค่าที่ต่างจากค่าปกติมาก.

การวิเคราะห์และการแปลผลลัพธ์ช่วยให้เราเข้าใจลักษณะของข้อมูลและปรับตัวแบบการวิเคราะห์ต่อไป.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

Seaborn

Seaborn เป็นไลบรารีสำหรับการทำงานกับการแสดงผลข้อมูลทางสถิติ (statistical data visualization) ในภาษา Python. Seaborn ถูกพัฒนาขึ้นบน Matplotlib (ไลบรารีการทำงานกับกราฟและแผนภูมิใน Python) เพื่อช่วยในการสร้างกราฟที่สวยงามและมีสไตล์สำหรับการแสดงผลข้อมูลทางสถิติ.

คุณสมบัติที่ Seaborn มีครบถ้วนและทำให้การแสดงผลข้อมูลเป็นเรื่องง่ายมากขึ้น รวมถึง:

- การจัดรูปแบบกราฟ: Seaborn มีรูปแบบกราฟที่สวยงามและมีสไตล์, ทำให้ง่ายต่อการสร้างกราฟที่มีความน่าสนใจ.
- การจัดการข้อมูลที่ซับซ้อน: สามารถจัดการข้อมูลที่มีความซับซ้อนได้ง่ายด้วยฟังก์ชันที่ให้มาพร้อมกับ Seaborn.

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

Seaborn

- การสนับสนุนตัวแปรสถิติ: Seaborn มีฟังก์ชันที่ช่วยในการทำงานกับตัวแปรสถิติเช่นการคำนวณและแสดงกราฟการกระจาย (distribution plots), แผนภูมิกลุ่ม (categorical plots), แผนภูมิความสัมพันธ์ (relational plots), และอื่น ๆ.
- การรองรับ Pandas DataFrame: Seaborn สามารถใช้งานร่วมกับ Pandas DataFrame ได้อย่างสมบูรณ์, ทำให้ง่ายต่อการทำงานกับข้อมูลที่มีโครงสร้าง.

ตลอดจน, Seaborn เป็นเครื่องมือที่มีประสิทธิภาพสำหรับการทำข้อมูลทางสถิติที่ซับซ้อนและกราฟใน Python

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.1.2 ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

pip install seaborn

pip install seaborn คือคำสั่งที่ใช้ในการติดตั้งไลบรารี Seaborn ผ่านทาง pip (Python package installer). pip เป็นเครื่องมือที่ใช้ในการจัดการและติดตั้งไลบรารี (libraries) หรือแพ็คเกจ (packages) ใน Python

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from
matplotlib!=3.6.1,>=3.3->seaborn) (4.39.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from m
atplotlib!=3.6.1,>=3.3->seaborn) (1.0.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from matp
lotlib!=3.6.1,>=3.3->seaborn) (9.5.0)
Requirement already satisfied: cycler>=0.10 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from matpl
otlib!=3.6.1,>=3.3->seaborn) (0.11.0)
Requirement already satisfied: tzdata>=2022.1 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pan
das>=1.2->seaborn) (2023.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from panda
s>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from python-da
teutil>=2.7->matplotlib!=3.6.1,>=3.3->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.0

[notice] A new release of pip is available: 23.0.1 -> 23.3.2
[notice] To update, run: C:\Users\User\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
PS C:\AppServ\www\Python_DataMining>
```

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.1 ความหมายของการวิเคราะห์การกระจาย (Distribution Analysis)

การวิเคราะห์การกระจาย (Distribution Analysis) เป็นเทคนิคการวิเคราะห์ข้อมูลเพื่อหารูปแบบการกระจายของข้อมูล โดยรูปแบบการกระจายของข้อมูล หมายถึง ลักษณะการกระจายของข้อมูล เช่น การกระจายแบบปกติ การกระจายแบบเชิงเส้น การกระจายแบบทวินาม เป็นต้น

การวิเคราะห์การกระจายเป็นเทคนิคการวิเคราะห์ข้อมูลขั้นพื้นฐานที่ใช้ในการอธิบายลักษณะทั่วไปของข้อมูล โดยรูปแบบการกระจายของข้อมูลสามารถช่วยให้เราเข้าใจว่าข้อมูลส่วนใหญ่มีค่าอยู่ที่ใด มีค่าที่ผิดปกติหรือไม่ และนอกจากนี้ยังเป็นการจัดเตรียมข้อมูลเพื่อความสะดวกในการวิเคราะห์ข้อมูลขั้นต่อไป

การวิเคราะห์การกระจายสามารถใช้ได้กับข้อมูลประเภทตัวเลขหรือข้อมูลประเภทข้อความ โดยข้อมูลประเภทตัวเลขสามารถหารูปแบบการกระจายได้โดยสร้างแผนภูมิการกระจายของข้อมูล เช่น แผนภูมิแท่ง แผนภูมิเส้น แผนภูมิความถี่สะสม เป็นต้น

ข้อมูลประเภทข้อความสามารถหารูปแบบการกระจายได้โดยสร้างแผนภูมิการกระจายของข้อมูล เช่น แผนภูมิวงกลม แผนภูมิแท่ง เป็นต้น

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.2 ตัวอย่างการใช้การวิเคราะห์การกระจาย

ตัวอย่างการใช้การวิเคราะห์การกระจาย

- การวิเคราะห์รูปแบบการกระจายของคะแนนสอบ เช่น คะแนนสอบส่วนใหญ่อยู่ในช่วงใด คะแนนสอบกระจายตัวมากน้อยเพียงใด เป็นต้น
- การวิเคราะห์รูปแบบการกระจายของราคาสินค้า เช่น ราคาสินค้าแปรปรวนมากน้อยเพียงใด เป็นต้น
- การวิเคราะห์รูปแบบการกระจายของข้อมูลประชากร เช่น ประชากรส่วนใหญ่มีอายุประมาณเท่าใด ประชากรกระจายตัวมากน้อยเพียงใด เป็นต้น

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

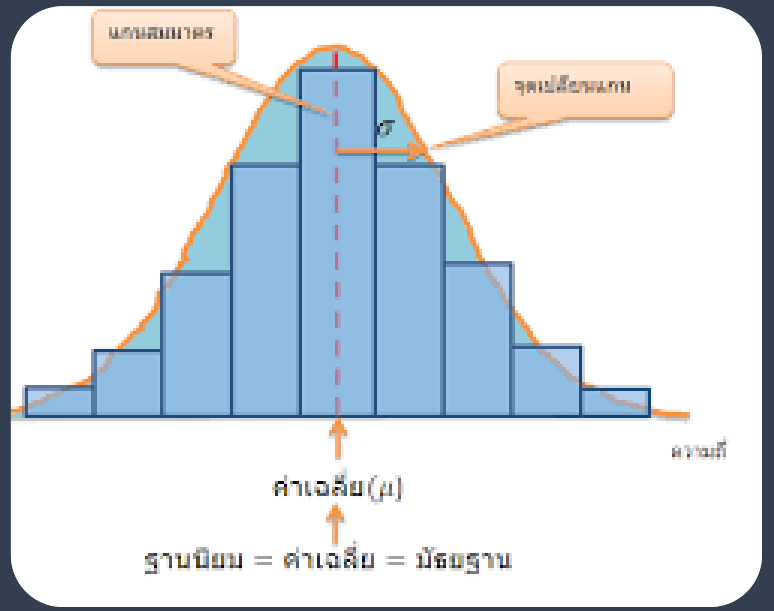
4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.3 ประเภทของการวิเคราะห์การกระจาย

การวิเคราะห์การกระจายสามารถแบ่งออกเป็นประเภทต่างๆ ได้ดังนี้

การกระจายแบบปกติ (Normal Distribution) เป็นรูปแบบการกระจายของข้อมูลทั่วไป โดยข้อมูลจะกระจายตัวเป็นรูปทรงระฆังคว่ำ โดยข้อมูลส่วนใหญ่จะอยู่ในช่วงกลางของข้อมูล และข้อมูลจะกระจายตัวน้อยลงเมื่ออยู่ห่างจากส่วนกลางของข้อมูล

รูปทรงของการแจกแจงจะมีลักษณะเป็นรูประฆังคว่ำ มีความสมมาตรกันทั้ง 2 ด้าน ซึ่งมีค่าเฉลี่ยอยู่ในตำแหน่งแกนสมมาตร(ตำแหน่งตรงกลาง) และมีส่วนเบี่ยงเบนมาตรฐานเป็นค่าแสดงการกระจายของข้อมูล และอยู่ที่ตำแหน่งจุดเปลี่ยนแกนของเส้นโค้ง



การกระจายแบบปกติ

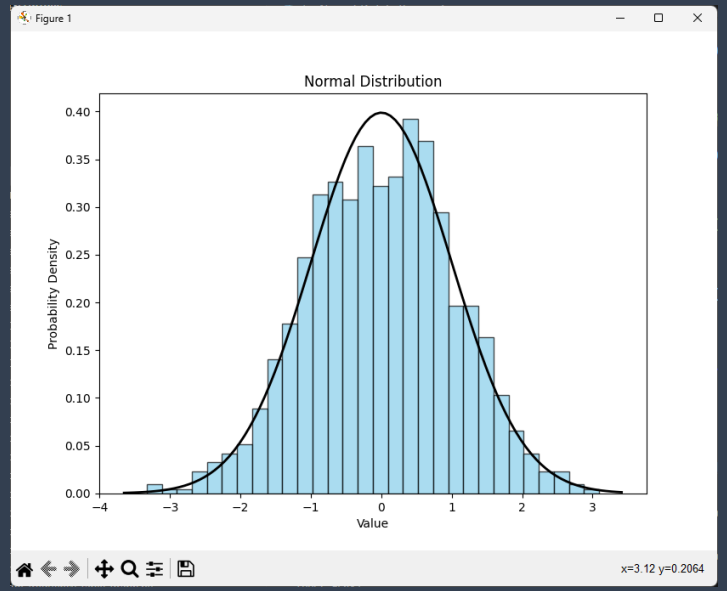
บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.3 ประเภทของการวิเคราะห์การกระจาย

การกระจายแบบปกติ (Normal Distribution)

เพื่อสร้างตัวอย่างข้อมูลที่มีการกระจายแบบปกติในภาษา Python, เราสามารถใช้ NumPy และ Matplotlib มาช่วย. ต่อไปนี้คือตัวอย่างโค้ดที่สร้างข้อมูลที่มีการกระจายแบบปกติและแสดง Histogram:



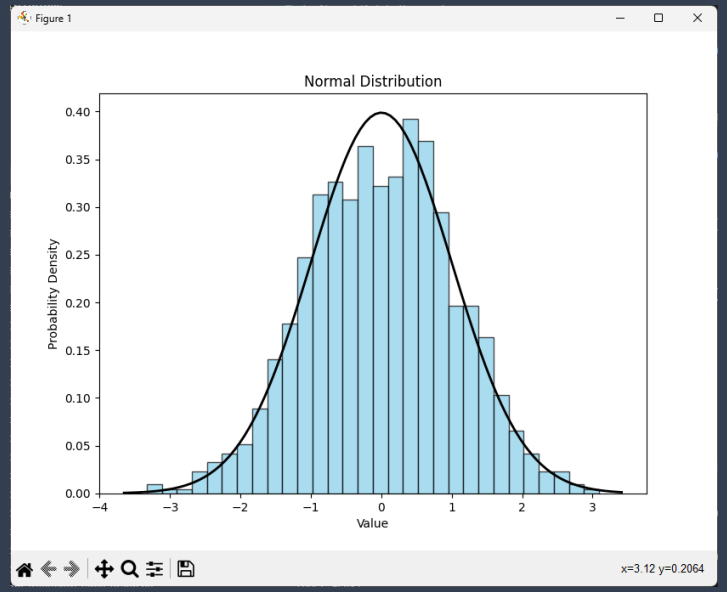
การกระจายแบบปกติ

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

ตัวอย่าง 4.7 : dm_Normal_Distributtion.py

```
import numpy as np
import matplotlib.pyplot as plt
# สร้างข้อมูลที่มีการกระจายแบบปกติ
mean = 0  # ค่าเฉลี่ย
std_dev = 1  # ส่วนเบี่ยงมาตรฐาน
sample_size = 1000  # ขนาดตัวอย่าง
# สุ่มข้อมูลจากการกระจายแบบปกติ
data = np.random.normal(mean, std_dev, sample_size)
# พล็อต Histogram เพื่อแสดงการกระจาย
plt.hist(data, bins=30, density=True, alpha=0.7, color='skyblue', edgecolor='black')
# พล็อต เส้นกราฟ Probability Density Function (PDF) ของการกระจายปกติ
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = (1/(std_dev * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mean)/std_dev)**2)
plt.plot(x, p, 'k', linewidth=2)
# กำหนดรายละเอียดกราฟ
plt.title('Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Probability Density')
# แสดงกราฟ
plt.show()
```



การกระจายแบบปกติ

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

ในตัวอย่างนี้:

ตัวอย่าง 4.7 : dm_Normal_Distributtion.py

เราใช้ numpy.random.normal เพื่อสร้างข้อมูลที่สุ่มมาจากการกระจายแบบปกติ.
พล็อต Histogram เพื่อแสดงการกระจายของข้อมูล.
เราพล็อต Probability Density Function (PDF) ของการกระจายปกติเพื่อแสดงรูปแบบการกระจาย.

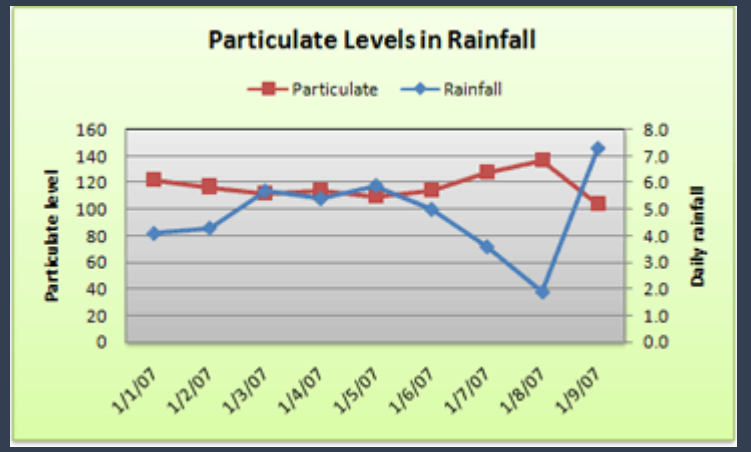
ค่า mean และ std_dev สามารถถูกปรับเปลี่ยนเพื่อสร้างการกระจายที่มีค่าเฉลี่ยและส่วนเบี่ยงมาตรฐานที่ต่างกัน.

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.3 ประเภทของการวิเคราะห์การกระจาย

การกระจายแบบเชิงเส้น (Linear Distribution) เป็นรูปแบบการกระจายของข้อมูล โดยข้อมูลจะกระจายตัวเป็นรูปเส้นตรง โดยข้อมูลจะเพิ่มขึ้นหรือลดลงอย่างสม่ำเสมอ



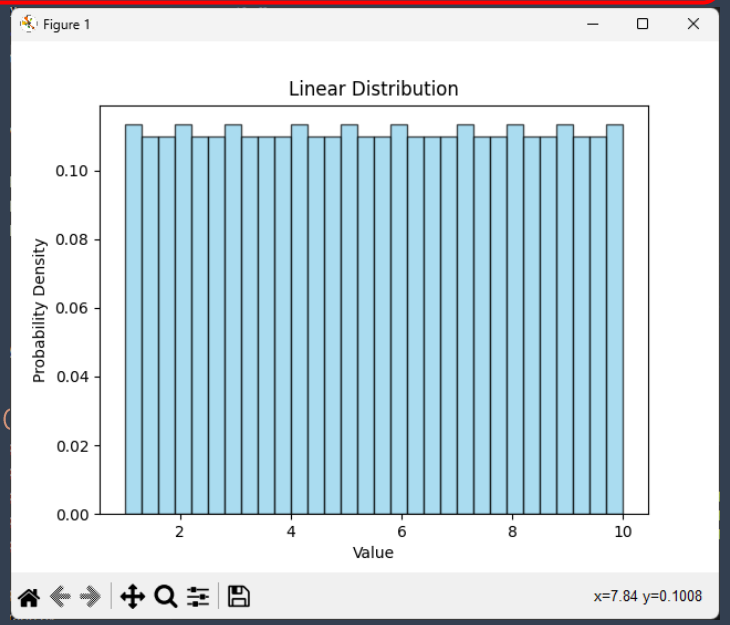
การกระจายแบบเชิงเส้น

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

ตัวอย่าง 4.8 : dm_Linear_Distributtion.py

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

# สร้างข้อมูลที่กระจายแบบเชิงเส้น
start = 1
end = 10
sample_size = 1000
# สร้างข้อมูลตามสมการเชิงเส้น
data = np.linspace(start, end, sample_size)
# พล็อต Histogram เพื่อแสดงการกระจาย
plt.hist(data, bins=30, density=True, alpha=0.7, color='skyblue', edgecolor='bla
# กำหนดรายละเอียดกราฟ
plt.title('Linear Distribution')
plt.xlabel('Value')
plt.ylabel('Probability Density')
# แสดงกราฟ
plt.show()
```



การกระจายแบบปกติ

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

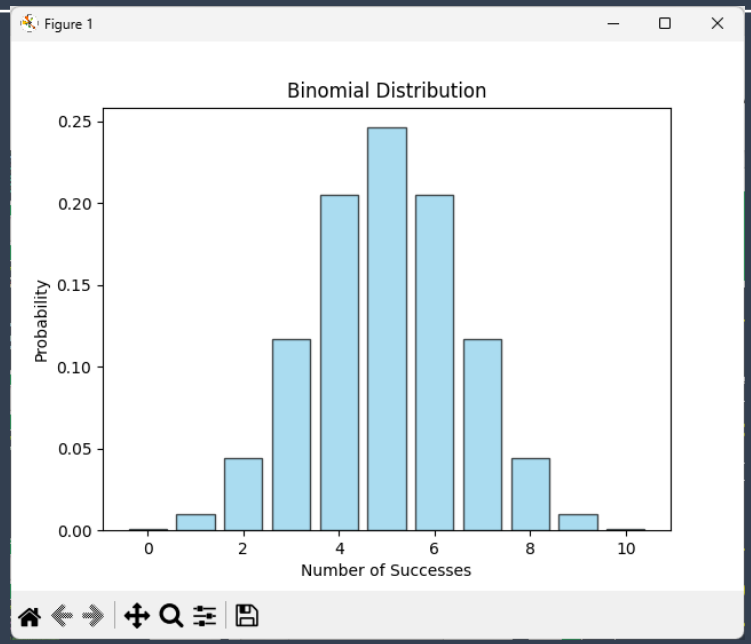
4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

4.2.3 ประเภทของการวิเคราะห์การกระจาย

การกระจายแบบทวินาม (Binomial Distribution) เป็นรูปแบบการกระจายของข้อมูล โดยข้อมูลจะกระจายตัวเป็นสองกลุ่มเท่าๆ กัน เช่น ผลลัพธ์ของการแข่งขันกีฬา เป็นต้น

สรุป

การวิเคราะห์การกระจายเป็นเทคนิคการวิเคราะห์ข้อมูลพื้นฐานที่มีความสำคัญและมีประโยชน์ในการวิเคราะห์ข้อมูลต่างๆ โดยรูปแบบการกระจายของข้อมูลสามารถช่วยให้เราเข้าใจลักษณะทั่วไปของข้อมูล และสามารถนำไปใช้ทำการวิเคราะห์ข้อมูลขั้นต่อไป



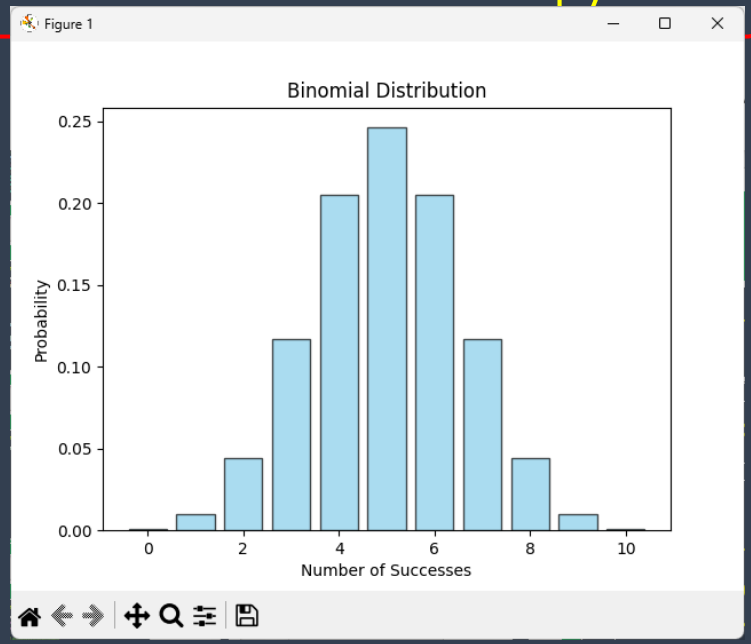
การกระจายแบบทวินาม

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

4.2 การวิเคราะห์การกระจาย (Distribution Analysis)

ตัวอย่าง 4.8 : dm_Linear_Distributtion.py

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom
# พารามิเตอร์ของการกระจายแบบทวินาม
n = 10  # จำนวนทดลอง
p = 0.5  # ความน่าจะเป็นของการสำเร็จในแต่ละทดลอง
# สร้างข้อมูล
data = np.arange(0, n+1)
probabilities = binom.pmf(data, n, p)
# พล็อตการกระจายแบบทวินาม
plt.bar(data, probabilities, color='skyblue', edgecolor='black', alpha=0.7)
# กำหนดรายละเอียดกราฟ
plt.title('Binomial Distribution')
plt.xlabel('Number of Successes')
plt.ylabel('Probability')
# แสดงกราฟ
plt.show()
```



การกระจายแบบทวินาม

อ้างอิง

- http://dataminingtrend.com/2014/association-rules/
- http://www.autosoft.in.th/data-science/forecasting-with-the-microsoft-time-series-data-mining-algorithm/