วิชาเหมืองข้อมูล Data Mining (4124305)

บทที่ 4

2/2568

การวิเคราะห์ข้อมูลเชิงสถิติในงานเหมืองข้อมูล (Statistical Data Analysis for Data Mining)

Asst. Prof. Dr. Nattapong Songneam

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

แก้ไขล่าสุด : 04.01.2569

บทที่ 4
การวิเคราะห์ข้อมูลเชิงสถิติในงานเหมืองข้อมูล (Statistical Data Analysis for Data Mining)

เทคนิคเชิงสถิติสำหรับเหมืองข้อมูล (Statistical Techniques for Data Mining) คือกระบวนการใช้เครื่องมือและเทคนิคทางสถิติเพื่อวิเคราะห์ข้อมูลขนาดใหญ่ (Big Data) เพื่อค้นหาความรู้หรือรูปแบบที่ซ่อนอยู่ในข้อมูล ซึ่งมีความสำคัญอย่างมากในหลากหลายสาขา เช่น การตลาด การเงิน การแพทย์ และการวิจัยทางวิทยาศาสตร์

บทที่ 4 เทคนิคเชิงสถิติ (Statistical Techniques)

Agenda

ทบทวน

ข้อมูล

การเตรียมข้อมูล

4.1 การวิเคราะห์ความถี่ (Frequency Analysis)
4.2 การวิเคราะห์การกระจาย (Distribution Analysis)
4.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)
4.4 การวิเคราะห์การแปรปรวน (Variation Analysis)
4.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

ทบทวน

0 CRISP-DM 6 ขั้นตอน

- **CRISP-DM** ย่อมาจาก Cross-industry standard process for data mining
- ซึ่งหมายถึง กระบวนการมาตรฐานที่ใช้สำหรับการทำเหมืองข้อมูล เพื่อทำการวิเคราะห์และนำไปใช้ประโยชน์ทางธุรกิจ ประกอบไปด้วย 6 ขั้นตอน ดังภาพ

- พัฒนาโดย 3 บริษัท
  - บริษัท SPSS
  - บริษัท Daimler Chrysler
  - บริษัท NCR

ทบทวน

0 CRISP-DM 6 ขั้นตอน

1. **เข้าใจธุรกิจ(Business Understanding)** ... เป็นขั้นตอนแรกมุ่งไปที่การทำความเข้าใจธุรกิจ ปัญหาและวัตถุประสงค์ของโครงการจากมุมมองทางธุรกิจ จากนั้นแปลงปัญหาให้อยู่ในรูปของโจทย์สำหรับการวิเคราะห์ข้อมูล และวางแผนการดำเนินงานเบื้องต้น
    1. ทำความเข้าใจปัญหา หรือโอกาสเชิงธุรกิจ
    2. ระบุ output หรือเป้าหมายที่ต้องการได้จากการวิเคราะห์ข้อมูลด้วย Data Mining

- ตัวอย่างเช่น
    - ทำอย่างไรถึงเพิ่มยอดขายให้กับสินค้าชนิดต่าง ๆ ได้
    - ต้องการแบ่งกลุ่มนักศึกษาออกตามความสนใจ / ถนัด
    - ทำอย่างไรให้ลูกค้ากลับมาซื้อสินค้าอีก
    - อยากทำนายปริมาณน้ำฝนที่ตกในอีก 2 วันข้างหน้า
    - อยากรู้ว่าลูกค้าคนใหนมีโอกาสป่วยเป็นโรคมะเร็ง

ทบทวน

0 CRISP-DM 6 ขั้นตอน

ตัวอย่างของการเข้าใจธุรกิจ

ตัวอย่างร้าน H&W CAFÉ ร้าน เบเกอรรี่ มีอาหารและเครื่องดื่ม มีระบบขายสินค้าด้วยคอมพิวเตอร์ บันทึกลงฐานในฐานข้อมูล ...สิ่งที่เราสนใจหรือโจทย์ก็คือ ผู้ซื้อสินค้าหรือลูกค้า แบ่งออกเป็นออกเป็น 3 ประเภท 1. ทานที่ร้าน 2. สั่งกลับบ้าน 3. เดริเวอรี่

เมื่อทราบลูกค้าแบ่งออกเป็นประเภท ร้านค้าก็สามารถสร้างโปรโมชันสินค้าให้ตรงกับความต้องการของลูกค้าแต่ละกลุ่ม

ทบทวน

0 CRISP-DM 6 ขั้นตอน

การจำแนกข้อมูล (Data Classification) ของความเสี่ยงในการเกิดโรคเบาหวานหรือระดับความรุนแรงของโรค

ขั้นที่ 1 : Business Understanding

สำหรับการทำ Business Understanding ในขั้นตอนที่ 1 ของการวิเคราะห์ข้อมูลทางธุรกิจสำหรับการพยากรณ์ความเสี่ยงในการเกิดโรคเบาหวานหรือระดับความรุนแรงของโรคเบาหวาน สามารถเริ่มต้นด้วยการทำความเข้าใจวัตถุประสงค์หลักที่ต้องการ เช่น การทำนายความเสี่ยงหรือการจำแนกระดับความรุนแรงของโรค เพื่อที่จะนำไปใช้ในการวางแผนการดูแลและการรักษาผู้ป่วยอย่างเหมาะสม

ทบทวน

0 CRISP-DM 6 ขั้นตอน

2. ความเข้าใจข้อมูล (Data Understanding) ขั้นตอนนี้เริ่มต้นด้วยการรวบรวมข้อมูล จากนั้นทำความเข้าใจ ตรวจสอบคุณภาพของข้อมูล และเลือกข้อมูลที่เก็บรวบรวมมาว่าจะใช้ข้อมูลใดบ้างในการวิเคราะห์

โดยในขั้นตอนนี้เป็นการ รวบรวมข้อมูลที่เกี่ยวข้อง ข้อมูลถูกต้องมีน่าเชื่อถือ ข้อมูลที่ได้ต้องมีปริมาณมากเพียงพอ ข้อมูลที่ได้ต้องมีความเหมาะสม และมีรายละเอียดเพียงพอต่อการนำไปวิเคราะห์ ตัวอย่างเช่น

1. ข้อมูลการซื้อสินค้าของแต่ละคนทั้งปี พ.ศ. 2566
2. ข้อมูลผลการเรียนและการลงทะเบียนเรียนของนักศึกษาตลอดหลักสูตร 4 ปี เป็นต้น

ขั้นตอนที่ 1 และ 2 สามารถทำกลับไปมาได้ เนื่องจากการทำความเข้าใจธุรกิจทำให้เราเข้าใจข้อมูลมากขึ้น และการเข้าใจข้อมูลก็ทำให้เราเข้าใจธุรกิจมากขึ้นเช่นกัน

ทบทวน

0 CRISP-DM 6 ขั้นตอน

ตัวอย่าง การทำความเข้าใจข้อมูล

จาก ตย. งานวิจัยหนึ่ง ได้มาจากฐานข้อมูลการสั่งอาหารของร้านตั้งแต่ ปี 2557-2564 (จำนวน หรือ ขนาดข้อมูลมีความสำคัญ) จากตัวอย่างงานวิจัย ใช้ข้อมูล 37020 รายการ แยกเป็น 3 ชุด คือ ชุดที่ 1 นั่งทานที่ร้าน ชุดที่ 2 สั่งกลับไปทานที่บ้าน และชุดที่ 3 สั่งผ่านเดลิเวอรี่

ขั้นตอนที่ 1 และ 2 สามารถทำกลับไปมาได้ เนื่องจากการทำความเข้าใจธุรกิจทำให้เราเข้าใจข้อมูลมากขึ้น และการเข้าใจข้อมูลก็ทำให้เราเข้าใจธุรกิจมากขึ้นเช่นกัน

ทบทวน

0 CRISP-DM 6 ขั้นตอน

3. การเตรียมข้อมูล (Data preparation)

ขั้นตอนการเตรียมข้อมูล หมายถึง ขั้นตอนทั้งหมดที่จะทำเพื่อให้ข้อมูลดิบที่เรารวบรวมมา กลายเป็นข้อมูลสมบูรณ์ที่พร้อมจะเข้าสู่โมเดลในขั้นตอนที่ 4 ขั้นตอนนี้จึงเป็นขั้นตอนที่ต้องใช้เวลานานที่สุด เนื่องจากโมเดลที่ได้จากการทำดาต้าไมนิ่งจะให้ผลลัพธ์ที่ถูกต้องหรือไม่นั้นขึ้นอยู่กับคุณภาพของข้อมูลที่ใช้ เช่น การสร้างตาราง การลบข้อมูลที่ไม่ต้องการออก การแปลงข้อมูลให้อยู่ในรูปแบบที่ต้องการ เป็นต้น โดยประกอบไปด้วย 3 ขั้นตอนดังต่อไปนี้

1) การคัดเลือกข้อมูล (Select Data) เลือกตาราง เลือกแอตทริบิวต์

จากตัวอย่างงานวิจัย เลือกข้อมูลที่เกี่ยวข้องกับการสั่งอาหาร ข้อมูลพนักงานไม่ได้เลือก

2) การทำความสะอาดข้อมูล (Data Cleaning)

3) การแปลงข้อมูล (Data Transformation)

ทบทวน

0 CRISP-DM 6 ขั้นตอน

4. การสร้างโมเดล (Modeling)

ในขั้นตอนนี้ เราจะเลือกและทดสองสร้างโมเดลหลาย ๆ แบบที่น่าจะสามารถแก้ไขปัญหาที่ต้องการได้ จากนั้นค่อย ๆ ปรับค่าพารามิเตอร์ในแต่ละโมเดล เพื่อให้ได้โมเดลที่เหมาะสมที่สุดมาใช้ในการแก้ไขปัญหา

หมายเหตุ หากยังไม่ได้โมเดลที่พอใจ เราสามารถกลับไปเตรียมข้อมูลให้พร้อมมากกว่านี้ได้ เนื่องจากข้อมูลที่ดี ก็จะทำให้ได้ผลลัพธ์ที่ดีเช่นกัน ดังคำกล่าวที่ว่า
“Garbage in, Garbage out” นั่นเอง

0 CRISP-DM 6 ขั้นตอน

ทบทวน

5. การวัดประสิทธิภาพของโมเดล (Evaluation)

เราจะทำการวัดประสิทธิภาพของโมเดลที่ได้จากขั้นตอนที่ 4 เพื่อวัดว่าโมเดลมีประสิทธิภาพเพียงพอต่อการนำไปใช้งานแล้วหรือไม่ ซึ่งโมเดลแต่ละประเภทก็จะมีตัววัดประสิทธิภาพที่แตกต่างกันออกไป

ทบทวน

0 CRISP-DM 6 ขั้นตอน

6. การนำโมเดลไปใช้งานจริง (Deployment)

เป็นการนำโมเดลที่เหมาะสมที่สุดไปใช้งานจริง เพื่อวิเคราะห์และแก้ปัญหาที่ต้องการ

*** ในขั้นตอนการสร้างโมเดล อาจจะต้องทำการสร้างโมเดลหลาย ๆ ตัว เพื่อที่จะเลือกเอาผลลัพธ์หรือโมเดลที่เหมาะสม(ค่าความถูกต้อง + ความเร็ว) ที่สุดต่อการนำไปใช้งาน

บทที่ 4

4.1 ความหมายของเทคนิคเชิงสถิติ (Statistical Techniques)

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

โดยในบทนี้ เราจะกล่าวถึงเทคนิคเชิงสถิติที่ใช้ในการสร้างโมเดลเหมืองข้อมูล โดยเทคนิคเหล่านี้จะใช้ในการวิเคราะห์ข้อมูลและค้นหารูปแบบความสัมพันธ์ที่ซ่อนอยู่ในข้อมูล ซึ่งรูปแบบความสัมพันธ์เหล่านี้สามารถนำไปใช้ในการทำนายหรือการตัดสินใจได้

4.2 เทคนิคเชิงสถิติที่ใช้สำหรับเหมืองข้อมูล

เทคนิคเชิงสถิติที่ใช้สำหรับเหมืองข้อมูล มีดังนี้

1. การวิเคราะห์ความถี่ (Frequency Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาค่าความถี่ของข้อมูลแต่ละค่า โดยค่าความถี่ของข้อมูลแต่ละค่า หมายถึง จำนวนครั้งที่ข้อมูลนั้นปรากฏในข้อมูลทั้งหมด

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

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

4.2 เทคนิคเชิงสถิติที่ใช้สำหรับเหมืองข้อมูล

เทคนิคเชิงสถิติที่ใช้สำหรับเหมืองข้อมูล มีดังนี้

4. การวิเคราะห์การแปรปรวน (Variation Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาความแปรปรวนของข้อมูล โดยความแปรปรวนของข้อมูล หมายถึง ระดับการเปลี่ยนแปลงของข้อมูล

5. การวิเคราะห์แนวโน้ม (Trend Analysis) เป็นการวิเคราะห์ข้อมูลเพื่อหาแนวโน้มของข้อมูล โดยแนวโน้มของข้อมูล หมายถึง ทิศทางการเปลี่ยนแปลงของข้อมูล

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

1. ความหมายของการวิเคราะห์ความถี่ (Frequency Analysis)

การวิเคราะห์ความถี่ (Frequency Analysis) หมายถึง กระบวนการที่ใช้สถิติเพื่อวิเคราะห์และแสดงข้อมูลตามความถี่ของค่าต่างๆ ที่ปรากฏในชุดข้อมูล การวิเคราะห์ความถี่มีประโยชน์ในการทำความเข้าใจและสรุปลักษณะของข้อมูลที่มีอยู่ นี่คือขั้นตอนพื้นฐานของการวิเคราะห์ความถี่:

การวิเคราะห์ความถี่ (Frequency Analysis) เป็นเทคนิคการวิเคราะห์ข้อมูลเพื่อหาค่าความถี่ของข้อมูลแต่ละค่า โดยค่าความถี่ของข้อมูลแต่ละค่า หมายถึง จำนวนครั้งที่ข้อมูลนั้นปรากฏในข้อมูลทั้งหมด

การวิเคราะห์ความถี่เป็นเทคนิคการวิเคราะห์ข้อมูลขั้นพื้นฐานที่ใช้ในการอธิบายลักษณะทั่วไปของข้อมูล โดยค่าความถี่ของข้อมูลแต่ละค่าสามารถช่วยให้เราเข้าใจว่าข้อมูลส่วนใหญ่มีค่าอยู่ที่ใด มีค่าที่ผิดปกติหรือไม่ และนอกจากนี้ยังเป็นการจัดเตรียมข้อมูลเพื่อความสะดวกในการวิเคราะห์ข้อมูลขั้นต่อไป

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

1. ความหมายของการวิเคราะห์ความถี่ (Frequency Analysis)

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

ข้อมูลประเภทข้อความสามารถหาค่าความถี่ได้โดยแบ่งข้อมูลออกเป็นกลุ่มๆ แล้วนับจำนวนข้อมูลในแต่ละกลุ่ม ตัวอย่างเช่น ข้อมูลประเภทเพศสามารถแบ่งออกเป็นกลุ่มชายและหญิง จากนั้นนับจำนวนข้อมูลในแต่ละกลุ่ม

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

2. ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

1) การสร้างตารางความถี่ (Frequency Table)

เป็นการสร้างตารางที่แสดงค่าทั้งหมดที่ปรากฏในชุดข้อมูลและจำนวนครั้งที่เกิดขึ้นสำหรับแต่ละค่า

2) การสร้างกราฟความถี่ (Frequency Graph):

เป็นการสร้างกราฟที่แสดงความถี่ของแต่ละค่า. กราฟที่บ่งบอกความถี่ส่วนแบ่งของข้อมูลได้แก่ Histogram, Bar Chart, Pie Chart เป็นต้น

3) การคำนวณค่าสถิติเบื้องต้น:

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

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

2. ขั้นตอนการวิเคราะห์ความถี่ (Frequency Analysis)

4) การทำนายและการวิเคราะห์ทางสถิติ:
หากข้อมูลมีลักษณะการกระจายที่เป็นรูปแบบ (distribution) บางแบบ, การวิเคราะห์ความถี่สามารถช่วยในการทำนายและวิเคราะห์ข้อมูลเพิ่มเติม

5) การวิเคราะห์และเปรียบเทียบความถี่:
เปรียบเทียบความถี่ของค่าต่าง ๆ ในกลุ่มหรือสองชุดข้อมูลเพื่อดูความแตกต่าง

6) การแปลผลลัพธ์:
ให้ข้อมูลที่สามารถใช้ในการตีความหรือสรุปสิ่งที่เราได้วิเคราะห์

การวิเคราะห์ความถี่มีประโยชน์มากในการทำความเข้าใจลักษณะของข้อมูล, การตรวจสอบความสมมติฐาน, และการสร้างรายงานสรุปข้อมูล

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างตารางความถี่ (Frequency Table)

1) การสร้างตารางความถี่ (Frequency Table)

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

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างตารางความถี่ (Frequency Table)

1) การสร้างตารางความถี่ (Frequency Table)

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

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

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างตารางความถี่ (Frequency Table)

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

ตัวอย่างที่ 4.1 ชื่อไฟล์ : Lec04_01_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)
```

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

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างตารางความถี่ (Frequency Table)

ตัวอย่างที่ 4.1 ชื่อไฟล์ : Lec04_01_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>
```

ตัวอย่างที่ 4.1 ชื่อไฟล์ : Lec04_01_frequency_table.py

Lec04_Frequency_table.py > ...

```python
#---Project Name : Lec04_01_frequency_table.py
#--- Developer : Asst.Prof.Dr. Nattapong Songneam
#--- Create Date : 15.12.2025
#--- Last Modified : 1.5.12.2025
#-----------------------------------------------------
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)

```

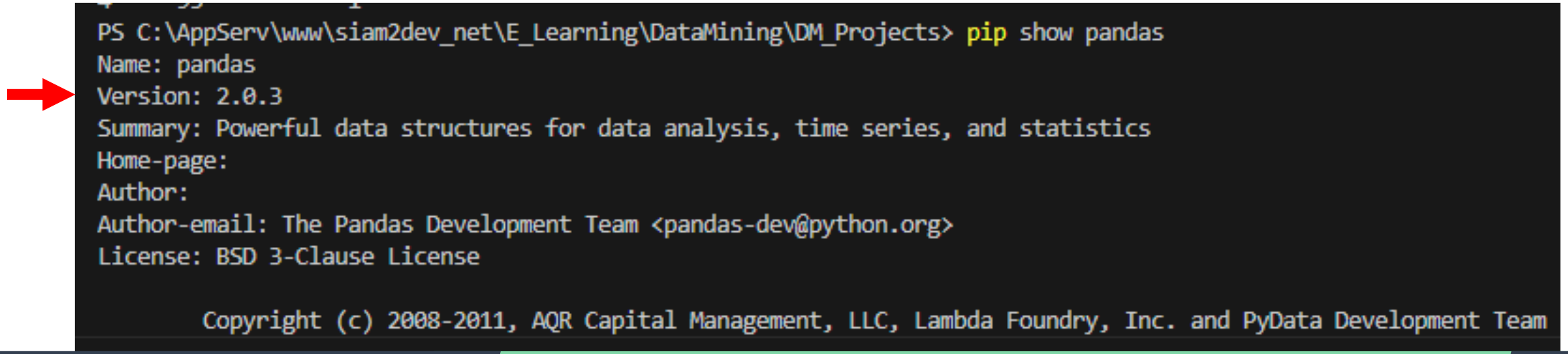
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Data_Mining_Project> & C:/Users/ASUS/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Data_Mining_Project/Lec04_Frequency_ta
PS D:\Data_Mining_Project> & C:/Users/ASUS/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Data_Mining_Project/Lec04_Frequency_ta
   คะแนน  จำนวนครั้ง
0     82          4
1     75          3
2     88          3
3     92          3
4     95          1
PS D:\Data_Mining_Project>
```

ใช้คำสั่ง `pip install pandas` เพื่อติดตั้ง Pandas จาก Terminal หรือ Command Prompt ใน VS Code.

pip install pandas

ตรวจสอบการติดตั้ง pandas ด้วยคำสั่ง pip show pandas



4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):

```python
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()
```

ตัวอย่างที่ 4.2 ชื่อไฟล์ : Lec04_02_frequency_graph.py

ใช้คำสั่ง `pip install pandas` เพื่อติดตั้ง Pandas จาก Terminal หรือ Command Prompt ใน VS Code.

```
pip install pandas
```

คำสั่งติดตั้งไลบรารี matplotlib

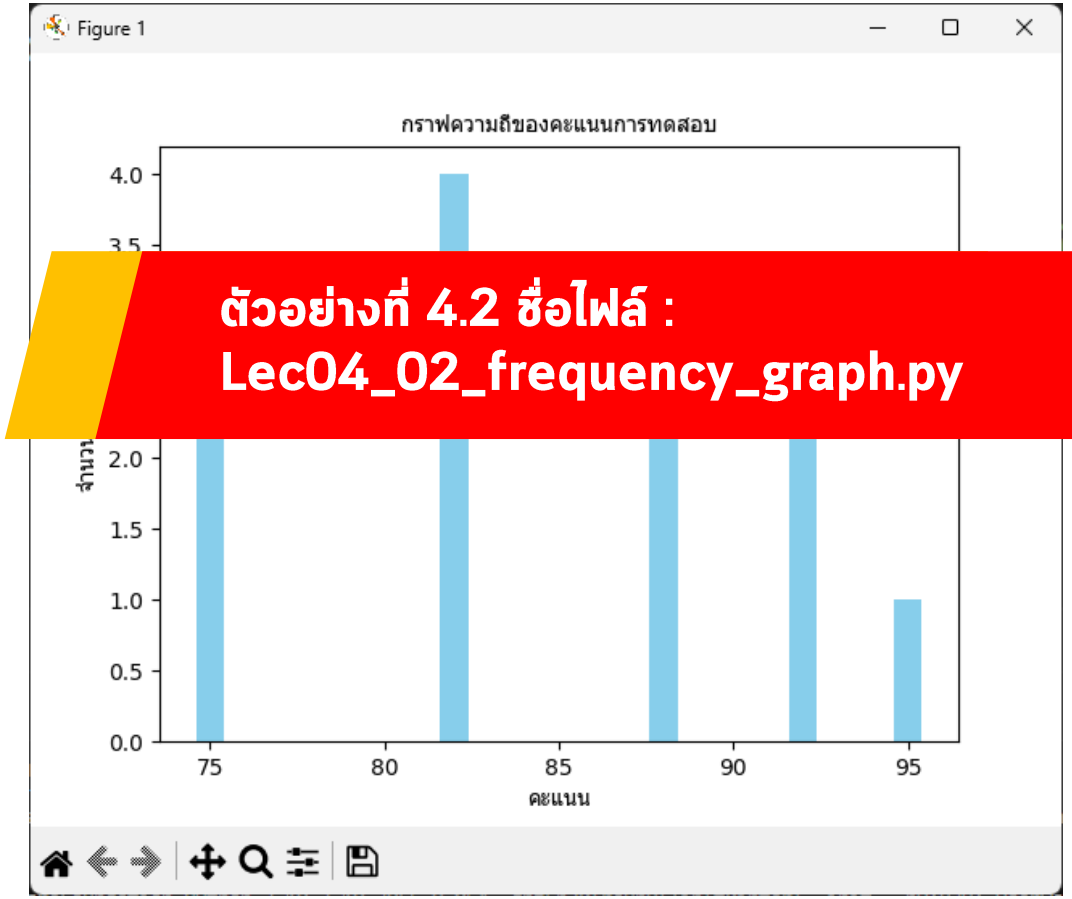
```
pip install matplotlib
```

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):

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 : Lec04_02_frequency_graph.py

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):

```python
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='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()
```

ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph.py

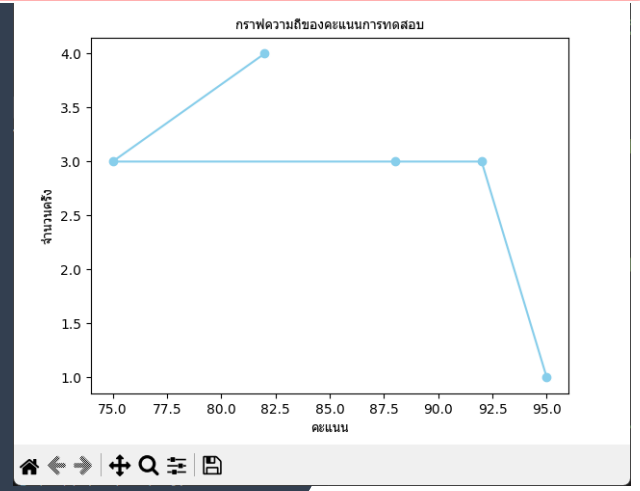
4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):

ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph.py

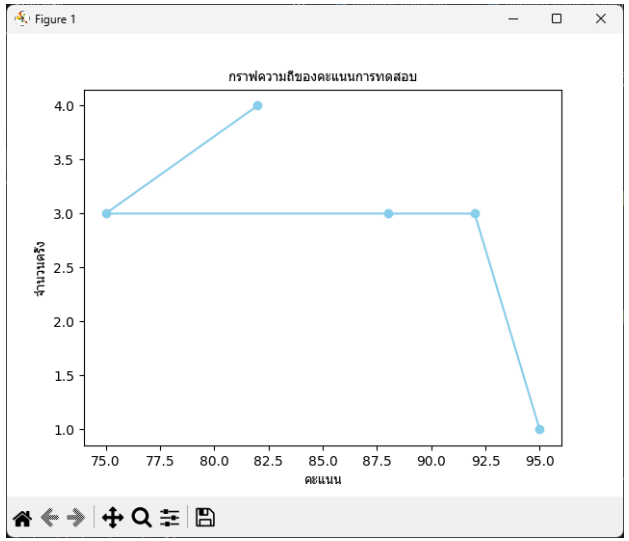
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager
# กำหนดฟอนต์ภาษาไทย (ในที่นี้ใช้ Tahoma)
font_thai = font_manager.FontProperties(fname="C:\\Windows\\Fonts\\tahoma.ttf")
# ข้อมูลตัวอย่าง: คะแนนการทดสอบของนักเรียน
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 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']
# สร้างกราฟเส้นตรง
plt.plot(frequency_table['คะแนน'], frequency_table['จำนวนครั้ง'], marker='o', 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()
```

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



4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):



ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph.py

สาเหตุที่ กราฟเส้นตรงไม่เรียงลำดับคะแนน แต่ดูเหมือนสลับไปมา เกิดจากขั้นตอนนี้ครับ

```
frequency_table = df['คะแนน'].value_counts().reset_index()
```

value_counts()

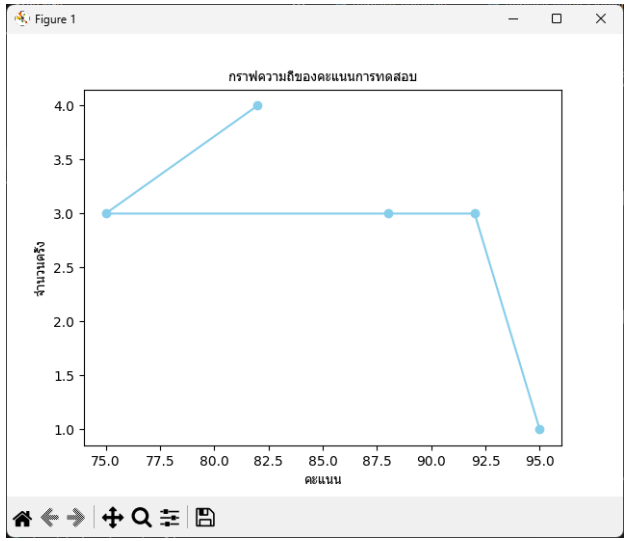
👉 จะเรียงข้อมูลตาม “จำนวนครั้ง (ความถี่)” จากมากไปน้อยโดยอัตโนมัติ

❌ ไม่ได้เรียงตามค่าคะแนน (75, 82, 88, 92, 95)

ดังนั้นลำดับคะแนนที่ได้จะเป็นประมาณนี้

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):



ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph.py

✅ วิธีแก้ (แนะนำ)
ให้ เรียงตามคะแนนก่อน **plot**

```
frequency_table = (
    df['คะแนน']
    .value_counts()
    .reset_index()
    .rename(columns={'index': 'คะแนน', 'คะแนน': 'จำนวนครั้ง'})
    .sort_values(by='คะแนน')   # ⭐ เรียงตามคะแนน
)
```

แล้วค่อย plot

```
plt.plot(
    frequency_table['คะแนน'],
    frequency_table['จำนวนครั้ง'],
    marker='o',
    color='skyblue'
)
```

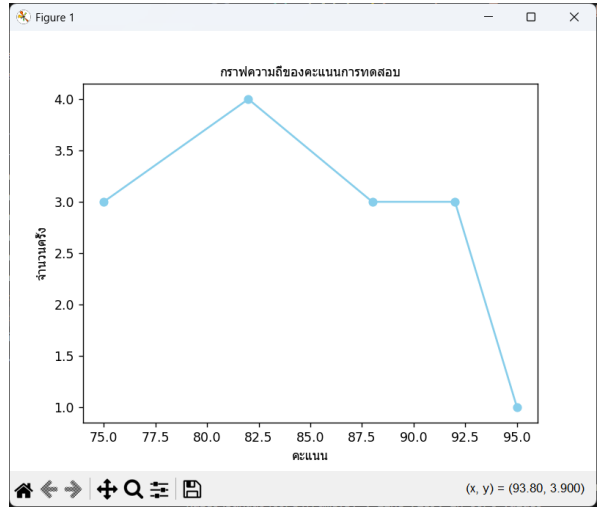
4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

3. การสร้างกราฟความถี่ (Frequency Graph):

ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph_01.py

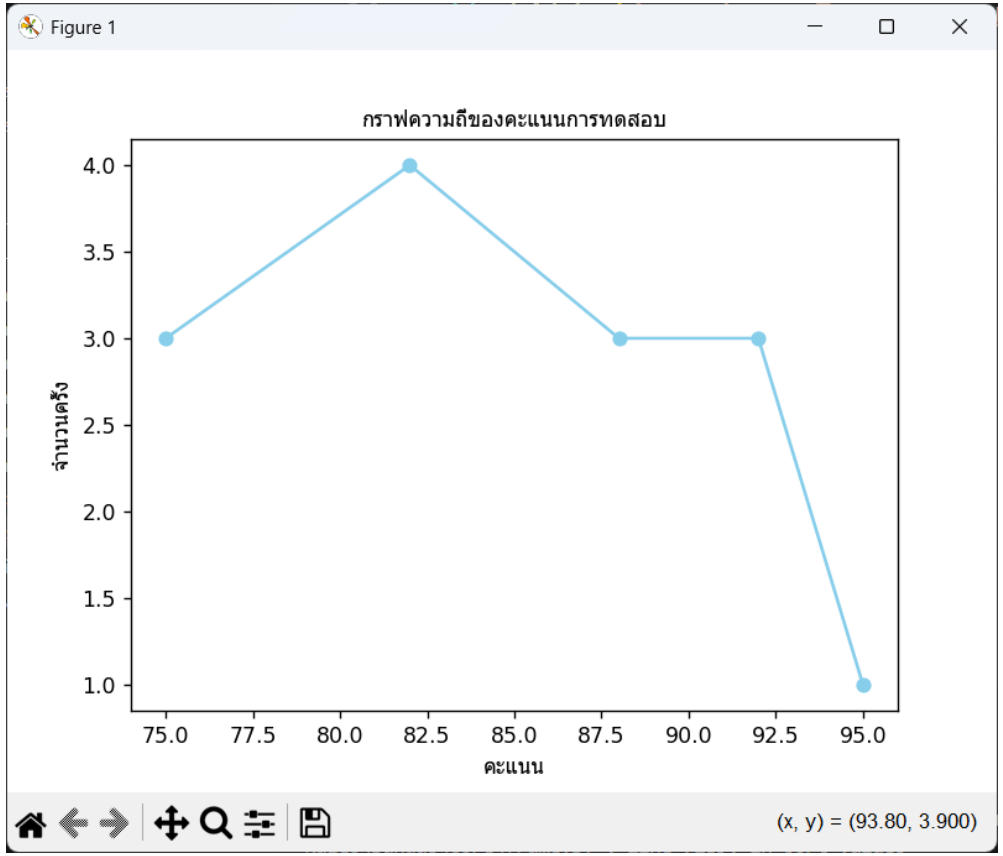
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager
# กำหนดฟอนต์ภาษาไทย
font_thai = font_manager.FontProperties(
    fname="C:\\Windows\\Fonts\\tahoma.ttf"
)
# ข้อมูลตัวอย่าง
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88,
75, 82]
# สร้าง DataFrame
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['
คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']
# สำคัญมาก
frequency_table =
frequency_table.sort_values(by='คะแนน')
```

```
# วาดกราฟเส้น
plt.plot(
    frequency_table['คะแนน'],
    frequency_table['จำนวนครั้ง'],
    marker='o',
    color='skyblue'
)
# ใส่ชื่อแกนภาษาไทย
plt.xlabel('คะแนน', fontproperties=font_thai)
plt.ylabel('จำนวนครั้ง',
fontproperties=font_thai)
plt.title('กราฟความถี่ของคะแนนการทดสอบ',
fontproperties=font_thai)
plt.show()
```



ตัวอย่างที่ 4.3 ชื่อไฟล์ : Lec04_03_frequency_LineGraph_01.py

```python
#---Project Name : Lec04_03_frequency_Linegraph01.py
#--- Developer : Asst.Prof.Dr. Nattapong Songneam
#--- Create Date : 15.12.2025
#--- Last Modified : 1.5.12.2025
#-------------------------------------------------------
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager
# กำหนดฟอนต์ภาษาไทย
font_thai = font_manager.FontProperties(
    fname="C:\\Windows\\Fonts\\tahoma.ttf"
)
# ข้อมูลตัวอย่าง
scores = [75, 82, 88, 92, 75, 82, 88, 95, 82, 92, 92, 88, 75, 82]
# สร้าง DataFrame
df = pd.DataFrame({'คะแนน': scores})
# สร้างตารางความถี่
frequency_table = df['คะแนน'].value_counts().reset_index()
frequency_table.columns = ['คะแนน', 'จำนวนครั้ง']   # ⭐ สำคัญมาก
frequency_table = frequency_table.sort_values(by='คะแนน')
# วาดกราฟเส้น
plt.plot(
    frequency_table['คะแนน'],
    frequency_table['จำนวนครั้ง'],
    marker='o',
    color='skyblue'
)
# ใส่ชื่อแกนภาษาไทย
plt.xlabel('คะแนน', fontproperties=font_thai)
plt.ylabel('จำนวนครั้ง', fontproperties=font_thai)
plt.title('กราฟความถี่ของคะแนนการทดสอบ', fontproperties=font_thai)
plt.show()

```



4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

4. การคำนวณค่าสถิติเบื้องต้น

3. 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:

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

5. การวิเคราะห์และเปรียบเทียบความถี่ :

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.

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

5. การวิเคราะห์และเปรียบเทียบความถี่ :

```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}')
```

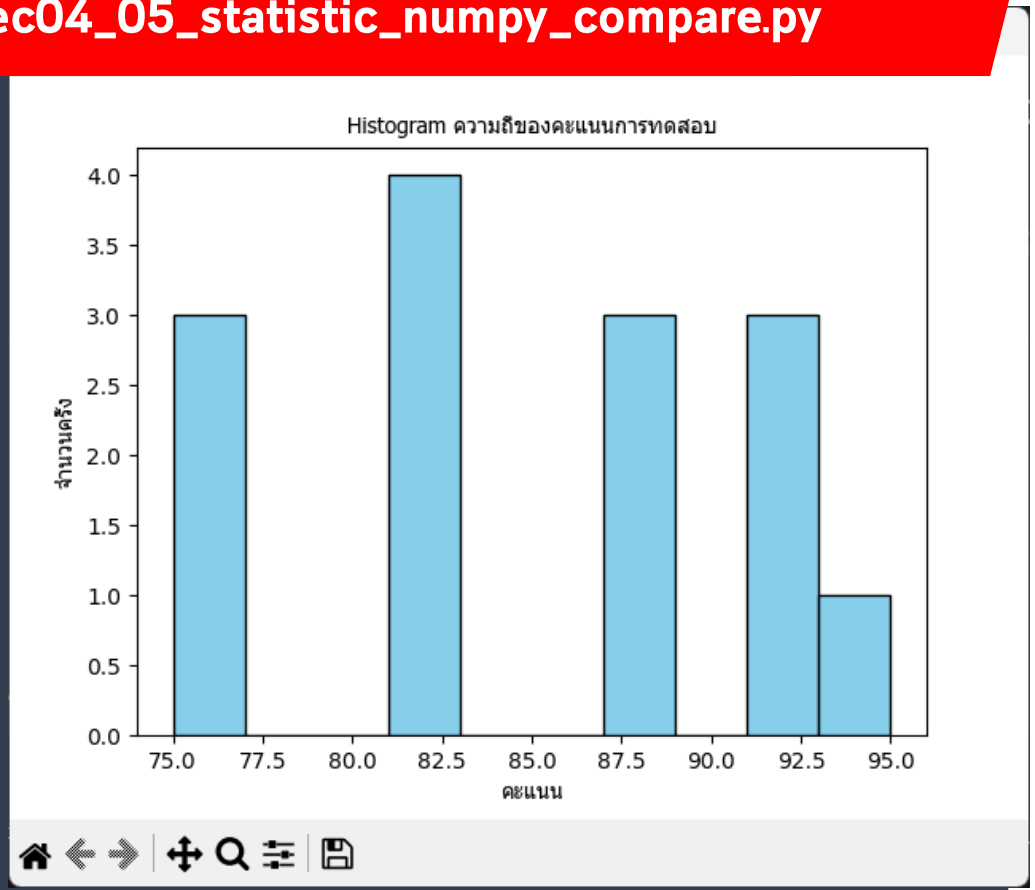
ตัวอย่างที่ 4.3 ชื่อไฟล์ :
Lec04_04_basic_Statistic_numpy.py

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

5. การวิเคราะห์และเปรียบเทียบความถี่ :

ตัวอย่างที่ 4.5 ชื่อไฟล์ : Lec04_05_statistic_numpy_compare.py

```
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}')
```



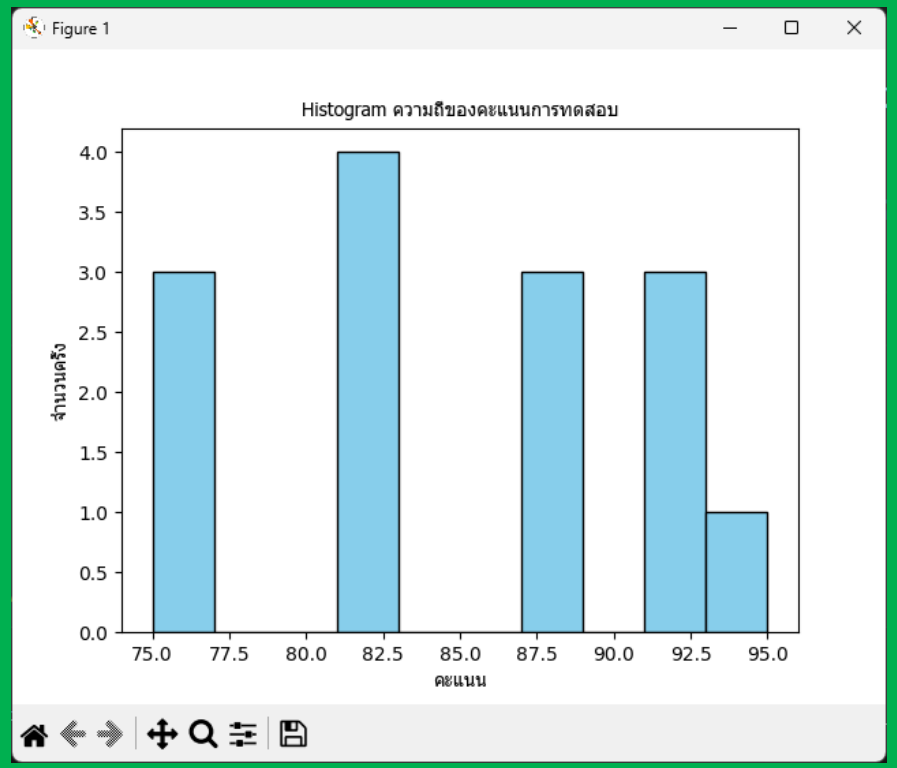
4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

5. การวิเคราะห์และเปรียบเทียบความถี่ :

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

ตัวอย่างที่ 4.5 ชื่อไฟล์ : Lec04_05_statistic_numpy_compare.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINA

PS C:\AppServ\www\Python_DataMining> & C:/
ค่าเฉลี่ย: 84.85714285714286
ค่ามัธยฐาน: 85.0
ค่าฐานนิยม: 82
ค่าความแปรปรวน: 46.9010989010989
ค่าส่วนเบี่ยงมาตรฐาน: 6.8484376978329O8
PS C:\AppServ\www\Python_DataMining>
```



4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

6. ตัวอย่างการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่างการใช้การวิเคราะห์ความถี่ ได้แก่

1) การวิเคราะห์พฤติกรรมการซื้อของผู้บริโภค เช่น สินค้าใดที่ผู้บริโภคนิยมซื้อมากที่สุด สินค้าใดที่ผู้บริโภคซื้อบ่อยที่สุด เป็นต้น

2) การวิเคราะห์รูปแบบการกระจายของคะแนนสอบ เช่น คะแนนสอบส่วนใหญ่อยู่ในช่วงใด คะแนนสอบกระจายตัวมากน้อยเพียงใด เป็นต้น

3) การวิเคราะห์ความแปรปรวนของราคาสินค้า เช่น ราคาสินค้าแปรปรวนมากน้อยเพียงใด เป็นต้น

การวิเคราะห์ความถี่เป็นเทคนิคการวิเคราะห์ข้อมูลพื้นฐานที่มีความสำคัญและมีประโยชน์ในการวิเคราะห์ข้อมูลต่างๆ

4.2 4.2.1 การวิเคราะห์ความถี่ (Frequency Analysis)

6. ตัวอย่างการวิเคราะห์ความถี่ (Frequency Analysis)

ตัวอย่างการใช้การวิเคราะห์ความถี่ได้แก่

1) การวิเคราะห์พฤติกรรมการซื้อของผู้บริโภค เช่น สินค้าใดที่ผู้บริโภคนิยมซื้อมากที่สุด สินค้าใดที่ผู้บริโภคซื้อบ่อยที่สุด เป็นต้น

ตัวอย่าง Data Set: พฤติกรรมการซื้อของผู้บริโภค

Transaction_ID	Customer_ID	เพศ	อายุ	สินค้า	หมวดสินค้า	ราคา	จำนวน	วันที่ซื้อ
T001	C001	ชาย	25	นม	อาหาร	35	2	1/1/2024
T002	C002	หญิง	32	ขนมปัง	อาหาร	25	1	1/1/2024
T003	C001	ชาย	25	น้ำอัดลม	เครื่องดื่ม	20	3	1/2/2024
T004	C003	หญิง	45	ผงซักฟอก	ของใช้	120	1	1/2/2024
T005	C004	ชาย	29	นม	อาหาร	35	1	1/3/2024
T006	C002	หญิง	32	นม	อาหาร	35	2	1/3/2024
T007	C005	หญิง	21	ขนมขบเคี้ยว	อาหาร	15	4	1/4/2024
T008	C003	หญิง	45	น้ำยาล้างจาน	ของใช้	55	2	1/4/2024

คำอธิบายแต่ละตัวแปร

- **Transaction_ID** : รหัสการซื้อ
- **Customer_ID** : รหัสลูกค้า
- **เพศ / อายุ** : ข้อมูลประชากรศาสตร์
- **สินค้า** : ชื่อสินค้า
- **หมวดสินค้า** : ใช้จัดกลุ่ม
- **ราคา** : ราคาต่อหน่วย
- **จำนวน** : จำนวนที่ซื้อ
- **วันที่ซื้อ** : วิเคราะห์พฤติกรรมตามเวลา

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

การวิเคราะห์การกระจาย (Distribution Analysis) คือ 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).

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

การวิเคราะห์การกระจาย (Distribution Analysis) คือ 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.

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

1. ความหมายของการวิเคราะห์การกระจาย (Distribution Analysis)

ความหมายของการวิเคราะห์การกระจาย (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

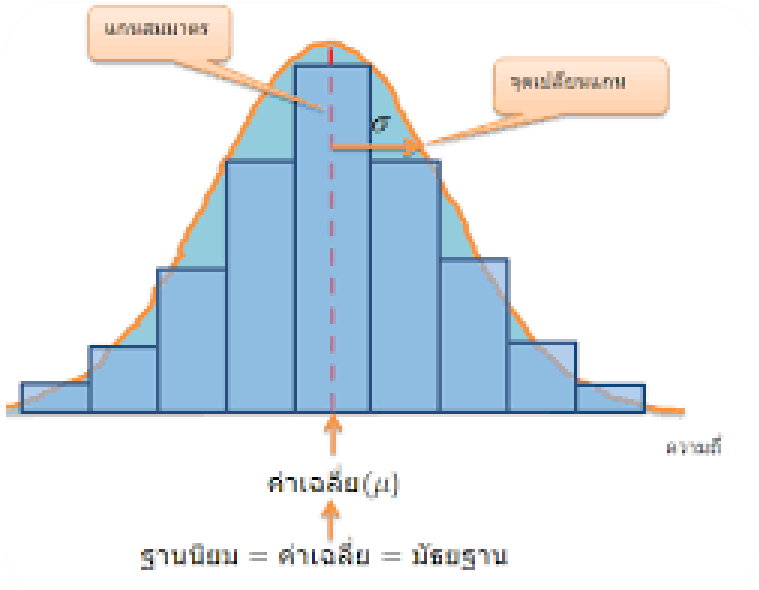
4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

2. ประเภทของการวิเคราะห์การกระจาย

การวิเคราะห์การกระจายสามารถแบ่งออกเป็นประเภทต่างๆ ได้ดังนี้

การกระจายแบบปกติ (Normal Distribution) เป็นรูปแบบการกระจายของข้อมูลทั่วไป โดยข้อมูลจะกระจายตัวเป็นรูปทรงระฆังคว่ำ โดยข้อมูลส่วนใหญ่จะอยู่ในช่วงกลางของข้อมูล และข้อมูลจะกระจายตัวน้อยลงเมื่ออยู่ห่างจากส่วนกลางของข้อมูล

รูปทรงของการแจกแจงจะมีลักษณะเป็นรูประฆังคว่ำ มีความสมมาตรกันทั้ง 2 ด้าน ซึ่งมีค่าเฉลี่ยอยู่ในตำแหน่งแกนสมมาตร(ตำแหน่งตรงกลาง) และมีส่วนเบี่ยงเบนมาตรฐานเป็นค่าแสดงการกระจายของข้อมูล และอยู่ที่ตำแหน่งจุดเปลี่ยนแกนของเส้นโค้ง



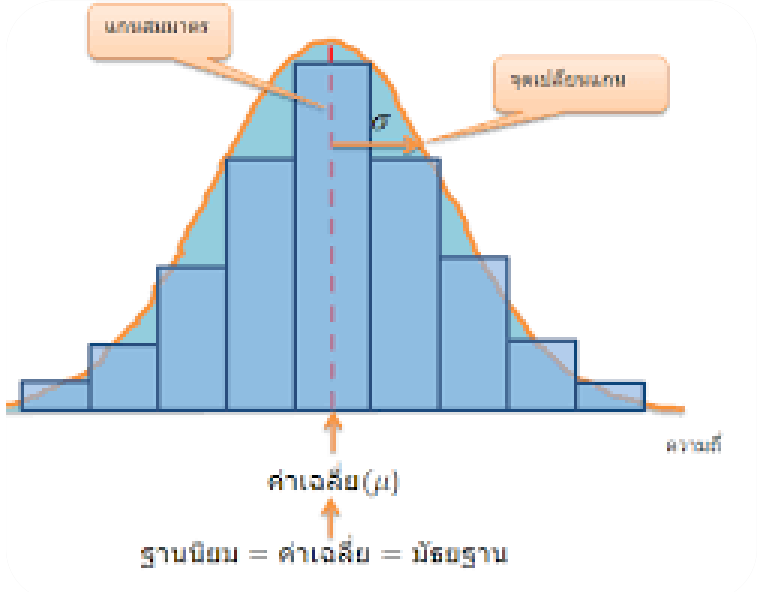
การกระจายแบบปกติ

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

2. ประเภทของการวิเคราะห์การกระจาย (ต่อ...)

การกระจายแบบปกติ (Normal Distribution)

เพื่อสร้างตัวอย่างข้อมูลที่มีการกระจายแบบปกติในภาษา Python, เราสามารถใช้ NumPy และ Matplotlib มาช่วย. ต่อไปนี้คือตัวอย่างโค้ดที่สร้างข้อมูลที่มีการกระจายแบบปกติและแสดง Histogram

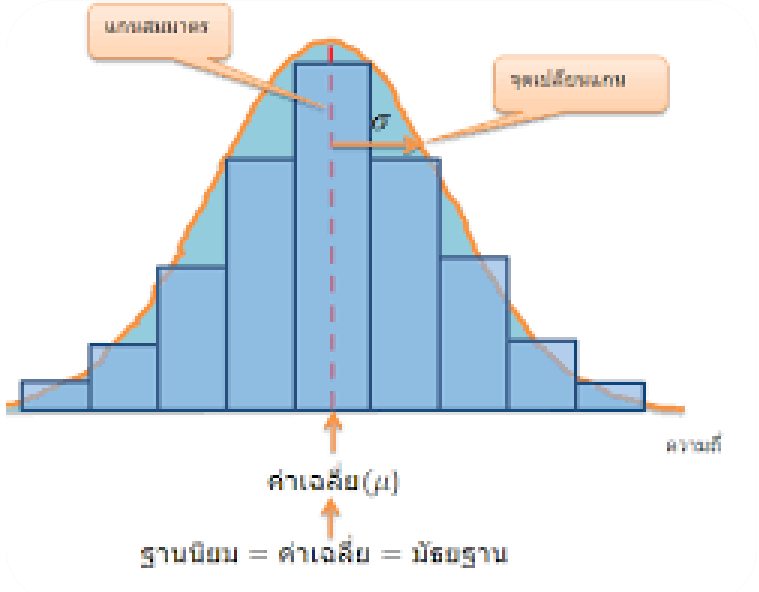


การกระจายแบบปกติ

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

2. ประเภทของการวิเคราะห์การกระจาย (ต่อ...)

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



การกระจายแบบปกติ

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

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

ตัวอย่างการใช้การวิเคราะห์การกระจาย

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

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

ตัวอย่างที่ 4.6 ชื่อไฟล์ : Lec04_06_statistic_numpy_compare.py

Part01 สร้างข้อมูล

```
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()
```

Part02 วิเคราะห์การกระจาย

```
# 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()
```

ตัวอย่างที่ 4.6 ชื่อไฟล์ : Lec04_06_statistic_numpy_compare.py

Part01 สร้างข้อมูล

```
1.  import numpy as np
2.  import pandas as pd
3.  import seaborn as sns
4.  import matplotlib.pyplot as plt
5.
    # สร้างข้อมูลที่สุ่มแบบปกติ
6.  np.random.seed(42)
7.  data = np.random.normal(loc=0, scale=1,
    size=1000)
    # เพิ่ม Outliers
8.  outliers = np.array([8, 10, -7, -6])
9.  data = np.concatenate([data, outliers])
    # สร้าง DataFrame
10. df = pd.DataFrame({'Data': data})
    # พล็อต Histogram
11. sns.histplot(df['Data'], kde=True)
12. plt.title('Histogram Data')
    # แสดงกราฟ
13. plt.show()
```

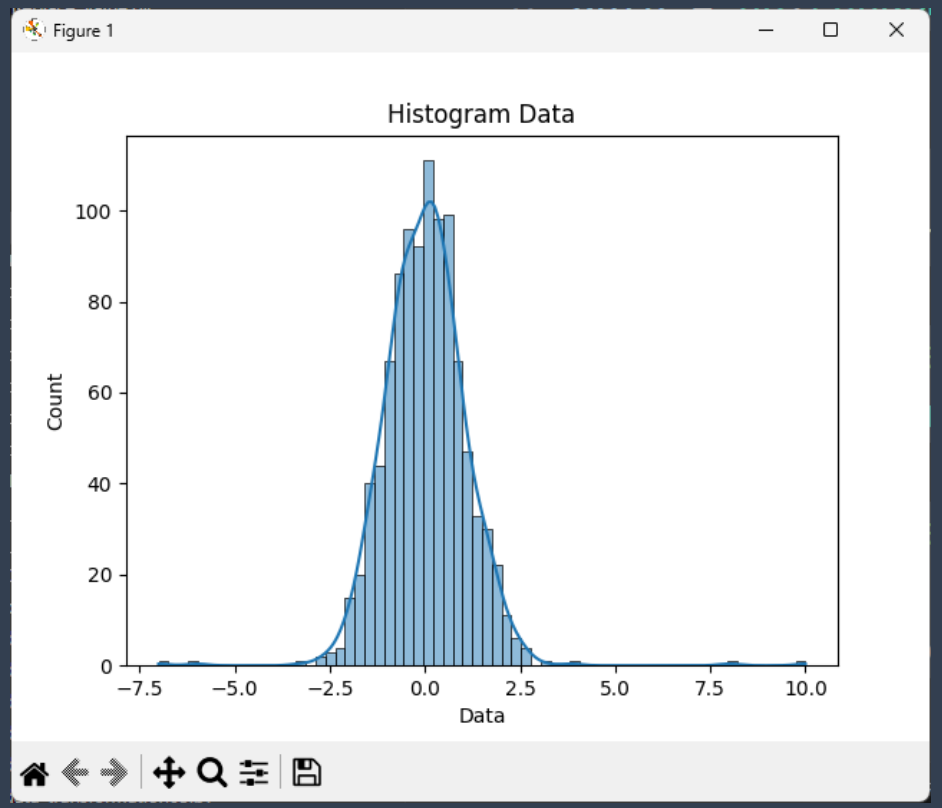
Part02 วิเคราะห์การกระจาย

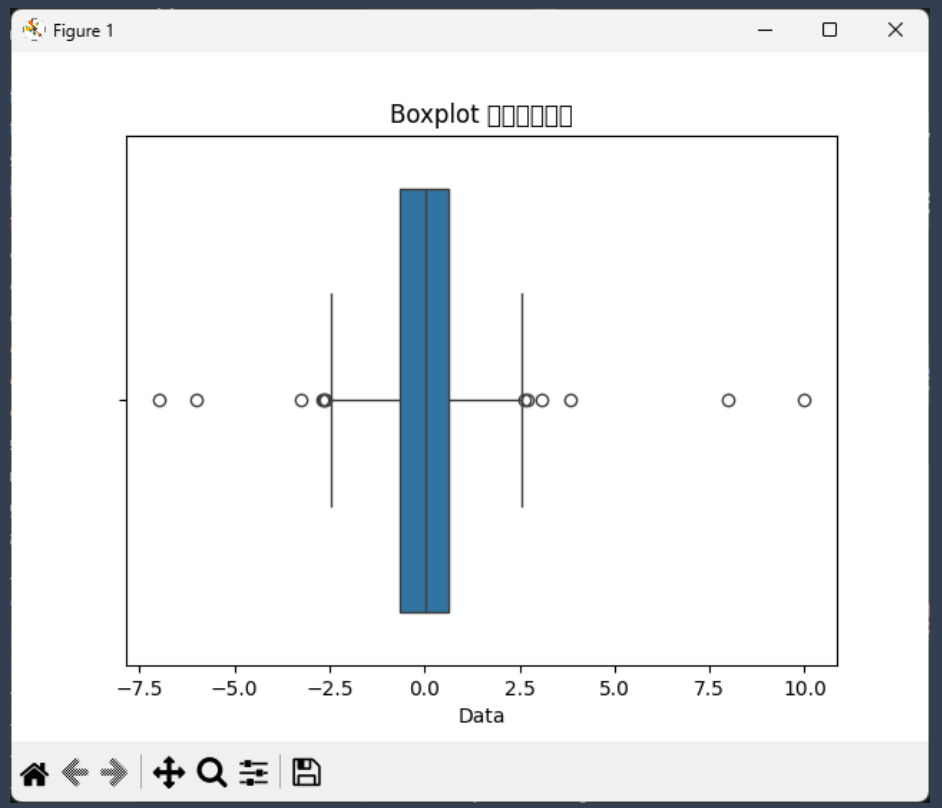
```
14. # คำนวณ Skewness และ Kurtosis
15. skewness = df['Data'].skew()
16. kurtosis = df['Data'].kurtosis()
    # แสดงผล Skewness และ Kurtosis
17. print(f'Skewness: {skewness}')
18. print(f'Kurtosis: {kurtosis}')
19. # แสดงตารางความถี่
20. frequency_table = pd.cut(df['Data'],
    bins=20).value_counts().sort_index().reset_inde
    x()
21. frequency_table.columns = ['ช่วง', 'จำนวน']
22. print('\nตารางความถี่:')
23. print(frequency_table)
    # พล็อต Boxplot เพื่อแสดง Outliers
24. sns.boxplot(x=df['Data'])
25. plt.title('Boxplot ข้อมูล')
    # แสดงกราฟ
26. plt.show()
```

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

ผลลัพธ์





4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

บทที่ 4

3. ตัวอย่างของการวิเคราะห์การกระจาย

การแปลผลลัพธ์

จากตัวอย่างที่ 4.6 นี้:

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

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

การติดตั้งไลบรารี Seaborn

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.

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

การติดตั้งไลบรารี Seaborn

Seaborn (ต่อ...)

- 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

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

การติดตั้งไลบรารี Seaborn

Seaborn (ต่อ...)

pip install seaborn

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

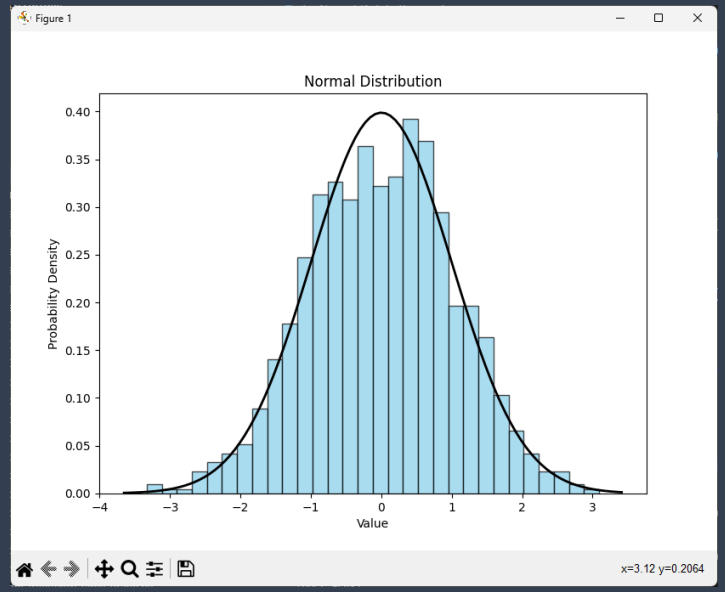
```
matplotlib!=3.6.1,>=3.3->seaborn) (4.39.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from m
atplotlib!=3.6.1,>=3.3->seaborn) (1.0.7)
Requirement already 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>
```

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

ตัวอย่าง 4.7 : Lec04_07_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

ตัวอย่างที่ 4.7 ชื่อไฟล์ :
Lec04_07_Normal_Distributtion.py

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

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

อภิปรายผล

ในตัวอย่างนี้:

ตัวอย่าง 4.7 : Lec04_07_Normal_Distributtion.py

- ใช้ numpy.random.normal เพื่อสร้างข้อมูลที่สุ่มมาจากการกระจายแบบปกติ.
- พล็อต Histogram เพื่อแสดงการกระจายของข้อมูล
- พล็อต Probability Density Function (PDF) ของการกระจายปกติเพื่อแสดงรูปแบบการกระจาย

ค่า mean และ std_dev สามารถถูกปรับเปลี่ยนเพื่อสร้างการกระจายที่มีค่าเฉลี่ยและส่วนเบี่ยงมาตรฐานที่ต่างกัน.

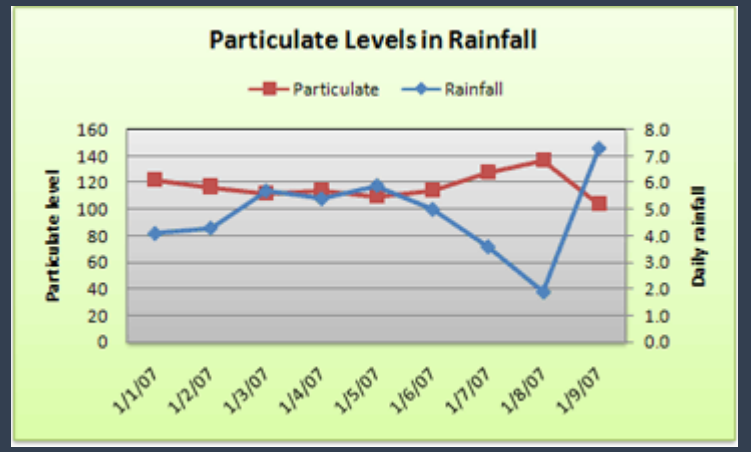
4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

อภิปรายผล

การกระจายแบบเชิงเส้น (Linear Distribution)

การกระจายแบบเชิงเส้น (Linear Distribution) เป็นรูปแบบการกระจายของข้อมูล โดยข้อมูลจะกระจายตัวเป็นรูปเส้นตรง โดยข้อมูลจะเพิ่มขึ้นหรือลดลงอย่างสม่ำเสมอ



การกระจายแบบเชิงเส้น

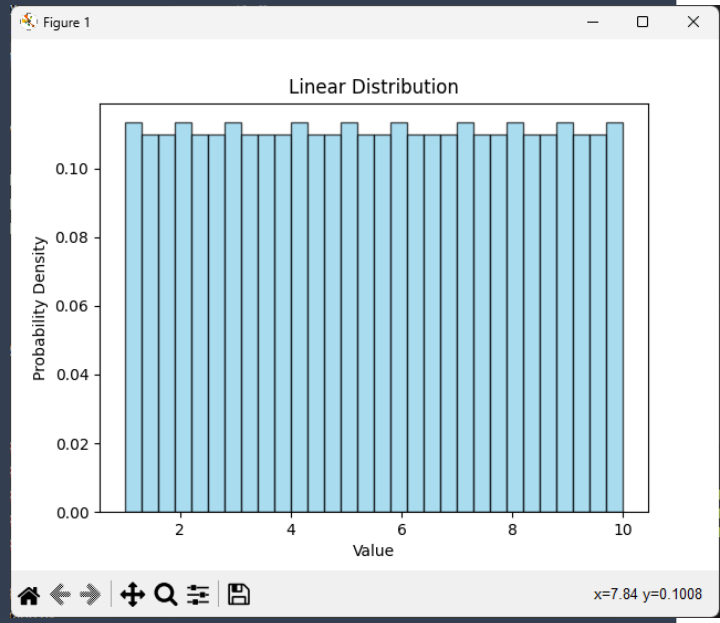
4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

กราฟแท่ง

ตัวอย่าง 4.8
Lec04_08_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='black')
# กำหนดรายละเอียดกราฟ
plt.title('Linear Distribution')
plt.xlabel('Value')
plt.ylabel('Probability Density')
# แสดงกราฟ
plt.show()
```



การกระจายแบบปกติ

ตัวอย่าง 4.8
Lec04_08_Linear_Distributtion.py

```
1. import numpy as np
2. import matplotlib.pyplot as plt
   # สร้างข้อมูลที่กระจายแบบเชิงเส้น
3. start = 1
4. end = 10
5. sample_size = 1000
6. # สร้างข้อมูลตามสมการเชิงเส้น
7. data = np.linspace(start, end, sample_size)
   # พล็อต Histogram เพื่อแสดงการกระจาย
8. plt.hist(data, bins=30, density=True, alpha=0.7, color='skyblue', edgecolor='black')
9. # กำหนดรายละเอียดกราฟ
10. plt.title('Linear Distribution')
11. plt.xlabel('Value')
12. plt.ylabel('Probability Density')
    # แสดงกราฟ
13. plt.show()
```

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

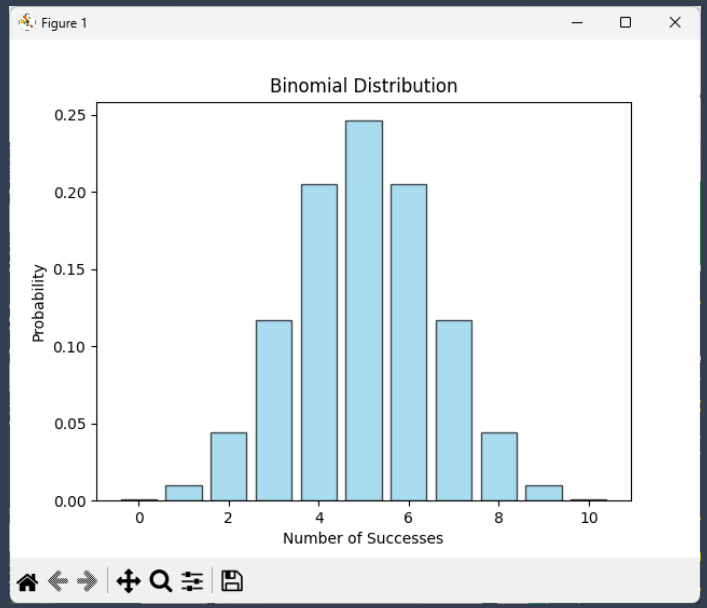
3. ตัวอย่างของการวิเคราะห์การกระจาย

การกระจายแบบทวินาม (Binomial Distribution)

การกระจายแบบทวินาม (Binomial Distribution) เป็นรูปแบบการกระจายของข้อมูล โดยข้อมูลจะกระจายตัวเป็นสองกลุ่มเท่าๆ กัน เช่น ผลลัพธ์ของการแข่งขันกีฬา เป็นต้น

สรุป

การวิเคราะห์การกระจายเป็นเทคนิคการวิเคราะห์ข้อมูลพื้นฐานที่มีความสำคัญและมีประโยชน์ในการวิเคราะห์ข้อมูลต่างๆ โดยรูปแบบการกระจายของข้อมูลสามารถช่วยให้เราเข้าใจลักษณะทั่วไปของข้อมูล และสามารถนำไปใช้ทำการวิเคราะห์ข้อมูลขั้นต่อไป



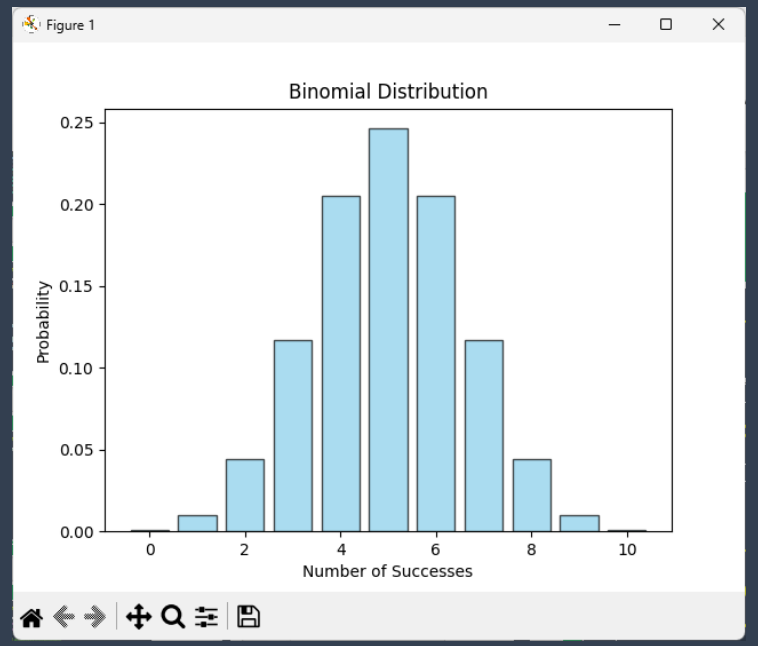
การกระจายแบบทวินาม

4.2 4.2.2 การวิเคราะห์การกระจาย (Distribution Analysis)

3. ตัวอย่างของการวิเคราะห์การกระจาย

ตัวอย่าง 4.8 : Lec04_08_Linear_Distributtion.py

```
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

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

1. ความหมายของการวิเคราะห์ความสัมพันธ์ (Relational Analysis)

- การวิเคราะห์ความสัมพันธ์ (Relational Analysis) เป็นหนึ่งในเทคนิคเชิงสถิติที่ใช้เพื่อศึกษาและวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรหรือตัวชี้วัดในชุดข้อมูล. ความสัมพันธ์นี้สามารถเป็นไปได้ในรูปแบบต่าง ๆ เช่น ความสัมพันธ์บวก, ความสัมพันธ์ลบ, หรือไม่มีความสัมพันธ์.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

1. ความหมายของการวิเคราะห์ความสัมพันธ์ (Relational Analysis) ต่อ...

การวิเคราะห์ความสัมพันธ์ (Relational Analysis) เป็นหนึ่งในเทคนิคเชิงสถิติที่ใช้เพื่อศึกษาและวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรหรือตัวชี้วัดในชุดข้อมูล. ความสัมพันธ์นี้สามารถเป็นไปได้ในรูปแบบต่าง ๆ เช่น ความสัมพันธ์บวก, ความสัมพันธ์ลบ, หรือไม่มีความสัมพันธ์.

นิยามของความสัมพันธ์ระหว่างตัวแปรสามารถทำได้โดยใช้ตัวชี้วัดทางสถิติบางประการ ซึ่งตัวชี้วัดที่มักจะใช้มีดังนี้:

1) สหสัมพันธ์ (Correlation):
2) การทดสอบสมมุติฐาน (Hypothesis Testing):
3) การวิเคราะห์เชิงเส้น (Linear Regression):
4) การวิเคราะห์ความแปรปรวน (Variance Analysis):

การวิเคราะห์ความสัมพันธ์มีวัตถุประสงค์หลายประการ เช่น การทำนาย, การจัดการและปรับปรุงกระบวนการทางธุรกิจ, หรือการค้นหาปัจจัยที่สามารถอธิบายความแปรปรวนของตัวแปรตามได้

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

2. ประเภทของการวิเคราะห์ความสัมพันธ์ (Relational Analysis)

1) สหสัมพันธ์ (Correlation): เป็นตัวชี้วัดที่ใช้วัดความสัมพันธ์ระหว่างตัวแปรที่มีการเปลี่ยนแปลงพร้อม ๆ กัน โดยใช้สหสัมพันธ์. ค่าสหสัมพันธ์อยู่ในช่วง -1 ถึง 1, โดย -1 แสดงถึงความสัมพันธ์ลบที่สมบูรณ์, 1 แสดงถึงความสัมพันธ์บวกที่สมบูรณ์, และ 0 แสดงถึงไม่มีความสัมพันธ์.

2) การทดสอบสมมุติฐาน (Hypothesis Testing): การทดสอบสมมุติฐานเป็นเทคนิคที่ใช้สถิติเพื่อทดสอบความสัมพันธ์ระหว่างตัวแปร ในบางกรณี, การทดสอบสมมุติฐานสามารถบอกถึงว่ามีหรือไม่มีความสัมพันธ์

3) การวิเคราะห์เชิงเส้น (Linear Regression): เป็นเทคนิคที่ใช้เพื่อวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรต้นและตัวแปรตาม โดยพยายามให้มีเส้นตรงที่เข้ากับข้อมูลให้ดีที่สุด.

4) การวิเคราะห์ความแปรปรวน (Variance Analysis): เป็นการวิเคราะห์การแปรปรวนของตัวแปรตามในบริบทของตัวแปรต้น ซึ่งช่วยในการทำความเข้าใจถึงการกระจายของข้อมูล.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

แปลเป็นภาษาไทยว่า "ความสัมพันธ์" หรือ "การเกี่ยวพัน" หมายถึง ความสัมพันธ์ระหว่างสองสิ่งหรือมากกว่านั้นว่ามีความเกี่ยวข้องกันหรือไม่อย่างไร โดยความสัมพันธ์ดังกล่าวอาจหมายถึงความสัมพันธ์เชิงปริมาณ (quantitative relationship) หรือความสัมพันธ์เชิงคุณภาพ (qualitative relationship)

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

ที่ X_i และ Y_i คือค่าของตัวแปรต้นและตัวแปรตามในตัวอย่างที่ i, $\bar{X}$ และ $\bar{Y}$ คือค่าเฉลี่ยของตัวแปรต้นและตัวแปรตาม, และ $\sum$ หมายถึงผลรวม.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

สหสัมพันธ์ (Correlation) เป็นตัวชี้วัดทางสถิติที่ใช้วัดความสัมพันธ์ระหว่างตัวแปรสองตัว วัดความเกี่ยวข้องและทิศทางของความสัมพันธ์ระหว่างตัวแปรนั้น ๆ วัดความสัมพันธ์ว่ามีแนวโน้มเปลี่ยนแปลงไปพร้อมกันหรือไม่ และในทิศทางใดบ้าง.
การวัดสหสัมพันธ์ใช้ค่าที่เรียกว่า "ค่าสหสัมพันธ์" (Correlation Coefficient) ซึ่งมักแทนด้วยสัญลักษณ์ "r". ค่าสหสัมพันธ์มีค่าระหว่าง -1 ถึง 1:

- ค่า r = 1 แสดงถึงความสัมพันธ์บวกที่สมบูรณ์ หรือความสัมพันธ์แบบเส้นตรงบวกที่สมบูรณ์ ค่าตัวแปรต้นเพิ่มขึ้น ตัวแปรตามก็เพิ่มขึ้น.
- ค่า r = -1 แสดงถึงความสัมพันธ์ลบที่สมบูรณ์ หรือความสัมพันธ์แบบเส้นตรงลบที่สมบูรณ์ ค่าตัวแปรต้นเพิ่มขึ้น ตัวแปรตามก็ลดลง.
- ค่า r = 0 แสดงถึงไม่มีความสัมพันธ์ทางเส้นตรงระหว่างตัวแปรสองตัว

ค่าสหสัมพันธ์สามารถคำนวณได้ตามสูตร Pearson Correlation Coefficient ดังนี้:

$$r=\Sigma(Xi-X^-)(Yi-Y^-)\Sigma(Xi-X^-)2\Sigma(Yi-Y^-)2r=\Sigma(Xi-X^-)2\Sigma(Yi-Y^-)2$$

$$\Sigma(Xi-X^-)(Yi-Y^-)$$

ที่ XiXi และ YiYi คือค่าของตัวแปรต้นและตัวแปรตามในตัวอย่างที่ i, X^-X^- และ Y^-Y^- คือค่าเฉลี่ยของตัวแปรต้นและตัวแปรตาม, และ ΣΣ หมายถึงผลรวม.

การทำวิเคราะห์สหสัมพันธ์ช่วยให้เราเข้าใจแนวโน้มและความสัมพันธ์ระหว่างข้อมูล ซึ่งมีประโยชน์ในหลายด้าน เช่น การวิเคราะห์ข้อมูล, การทำนาย, หรือการตรวจสอบสมมุติฐานในการวิจัย

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

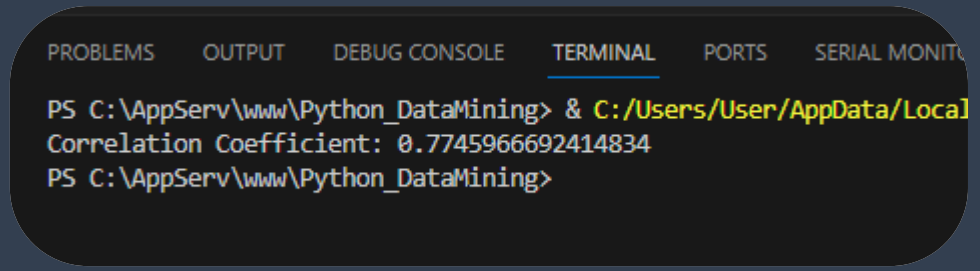
3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น:

```
import numpy as np

# สร้างข้อมูลตัวอย่าง
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 5, 4, 5])

# คำนวณ correlation coefficient
correlation_coefficient = np.corrcoef(x, y)[0, 1]

print(f"Correlation Coefficient: {correlation_coefficient}")
```

1 สหสัมพันธ์ (Correlation)
ตัวอย่าง 4.9 : Lec04_09_Correlation_numpy.py



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

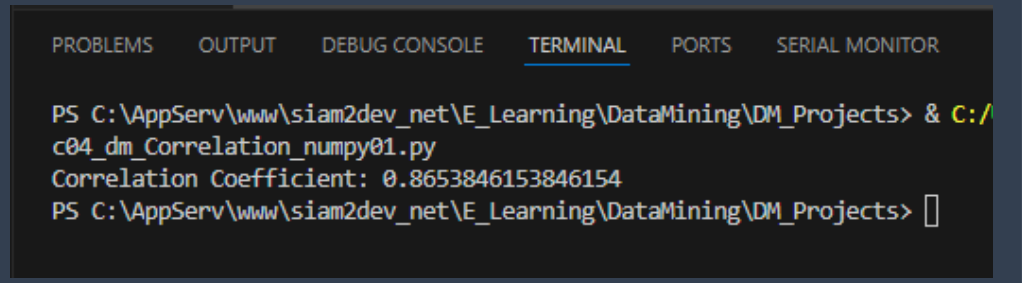
3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น:

1 สหสัมพันธ์ (Correlation)
ตัวอย่าง 4.9 : Lec04_09_Correlation_numpy.py

```
import numpy as np

# สร้างข้อมูลตัวอย่าง ชุดละ 20 ตัว
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25])
y = np.array([2, 4, 1, 5, 7, 3, 8, 10, 6, 12, 9, 15, 14, 11, 18, 16, 20, 22, 24, 21, 25, 23, 19, 17, 13])
# คำนวณ correlation coefficient
correlation_coefficient = np.corrcoef(x, y)[0, 1]

print(f"Correlation Coefficient: {correlation_coefficient}")
```



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น:

```
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25])
y = np.array([2, 4, 1, 5, 7, 3, 8, 10, 6, 12, 9, 15, 14, 11, 18, 16, 20, 22, 24, 21, 25, 23, 19, 17, 13])
```

dm_correlation_numpy01.py > x

```python
import numpy as np

# สร้างข้อมูลตัวอย่าง ชุดละ 20 ตัว
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
y = np.array([2, 4, 1, 5, 7, 3, 8, 10, 6, 12, 9, 15, 14, 11, 18, 16, 20, 22, 24, 21, 2
# คำนวณ correlation coefficient
correlation_coefficient = np.corrcoef(x, y)[0, 1]

print(f"Correlation Coefficient: {correlation_coefficient}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR

```
PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/AppServ/www/Python_DataMining/dm_correlation_numpy01.py
Correlation Coefficient: 0.8653846153846154
PS C:\AppServ\www\Python_DataMining>
```

การใช้งาน numpy
ตัวอย่าง 4.9 : Lec04_10_Correlation_numpy01.py
Correlation_Coefficient: 0.8653846153846154

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น:

การใช้งาน numpy

ตัวอย่าง 4.9 : Lec04_10_Correlation_numpy_noise.py

Correlation_Coefficient: 0.8653846153846154

```
import numpy as np

# สร้างข้อมูล x และ y ที่แตกต่างกัน
x_data = np.arange(1, 21)  # สร้างลำดับตัวเลข 1-20 สำหรับ x
y_data = x_data + np.random.normal(0, 3, 20)  # สร้าง y โดยเพิ่ม noise ที่แตกต่างกัน

# คำนวณ correlation coefficient
correlation_coefficient = np.corrcoef(x_data, y_data)[0, 1]

print(f"Correlation Coefficient: {correlation_coefficient}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects> & C:/
ec04_10_Correlation_numpy_noise.py
Correlation Coefficient: 0.8450668237596006
PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects> & C:/
ec04_10_Correlation_numpy_noise.py
Correlation Coefficient: 0.9554240240477998
PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects> & C:/
ec04_10_Correlation_numpy_noise.py
Correlation Coefficient: 0.9086136213948324
PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects>
```

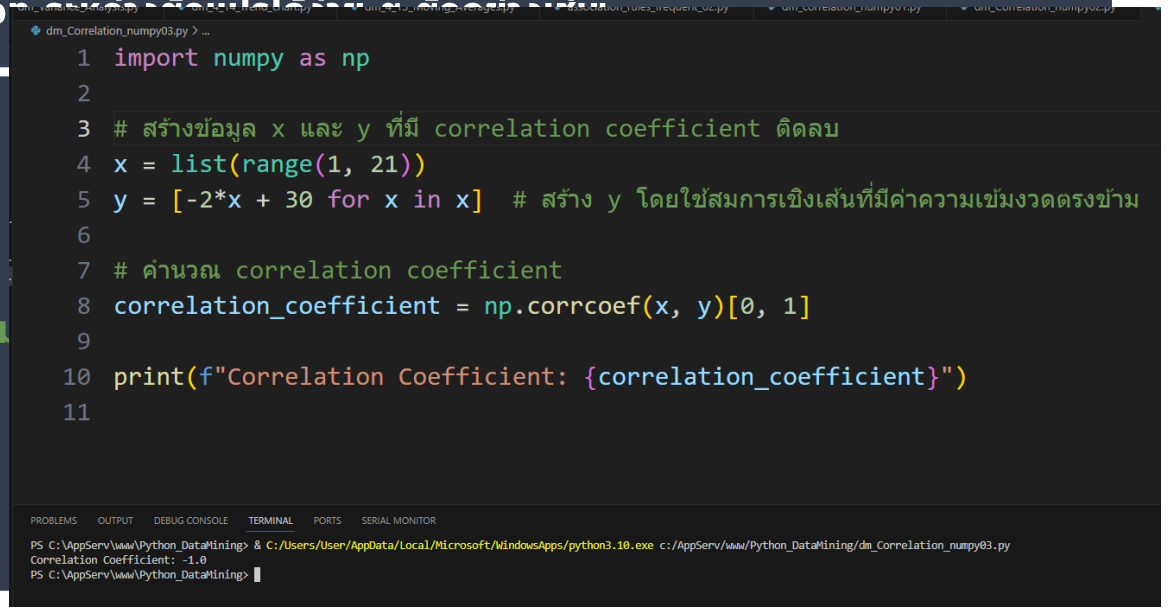
4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

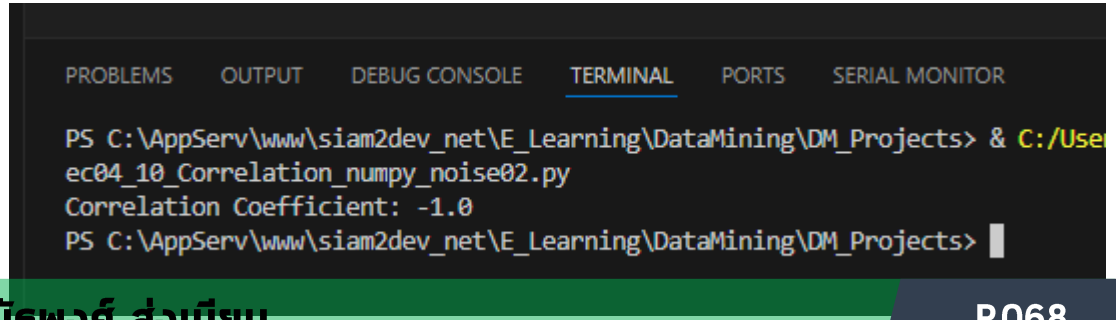
1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น

```python
import numpy as np

# สร้างข้อมูล x และ y ที่มี correlation coefficient ติดลบ
x_data = list(range(1, 21))
y_data = [-2*x + 30 for x in x_data]  # สร้าง y โดยใช้สมการเชิงเส้นที่มีค่าความเ
# คำนวณ correlation coefficient
correlation_coefficient = np.corrcoef(x_data, y_data)[0, 1]

print(f"Correlation Coefficient: {correlation_coefficient}")
```



การใช้งาน numpy
ตัวอย่าง 4.9 :
Lec04_10_Correlation_numpy_noise02.py



ยังไม่ได้นำเข้า numpy

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/AppSe
ec04_10_Correlation_numpy_noise02.py
Traceback (most recent call last):
  File "c:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects\Lec04_10_Correlation_numpy_noise02.py", line 6, in <module>
    correlation_coefficient = np.corrcoef(x_data, y_data)[0, 1]
NameError: name 'np' is not defined
PS C:\AppServ\www\siam2dev_net\E_Learning\DataMining\DM_Projects>
```

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ภาษา Python, คุณสามารถใช้ไลบรารี numpy หรือ pandas เพื่อคำนวณ correlation ระหว่างตัวแปรได้ง่าย ๆ ตัวอย่างเช่น:

Correlation coefficient ที่คุณได้รับคือ 0.7745966692414834 แสดงถึงความสัมพันธ์บวกที่มีความเข้มงวด (positive correlation) ระหว่างตัวแปรที่คำนวณ (อาจจะเป็น x และ y ในตัวอย่างข้างบน).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONIT
PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local
Correlation Coefficient: 0.7745966692414834
PS C:\AppServ\www\Python_DataMining>
```

ค่า correlation coefficient มีค่าระหว่าง -1 ถึง 1:

ถ้าค่าเป็นบวก, แสดงถึงความสัมพันธ์บวก ค่ามากขึ้นแสดงความเข้มงวดมากขึ้น.
ถ้าค่าเป็นลบ, แสดงถึงความสัมพันธ์ลบ ค่ามากขึ้นแสดงความเข้มงวดมากขึ้น
ถ้าค่าเป็น 0, แสดงไม่มีความสัมพันธ์.

ในที่นี้, ค่า correlation coefficient มีค่าบวก 0.77 ซึ่งแสดงถึงความสัมพันธ์บวกและมีความเข้มงวดมาก. นี้แปลว่าเมื่อค่า x เพิ่มขึ้น, ค่า y มีแนวโน้ม tend ที่จะเพิ่มขึ้นด้วยความน่าจะเป็นสูง.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

ค่าสัมประสิทธิ์สหสัมพันธ์ (correlation coefficient) มีค่าอยู่ระหว่าง -1 ถึง 1 โดยค่าสัมประสิทธิ์สหสัมพันธ์เป็นบวก หมายถึง ทั้งสองสิ่งมีความสัมพันธ์ในทิศทางเดียวกัน เช่น เมื่อระดับสติปัญญาสูงขึ้น คะแนนสอบก็จะสูงขึ้นด้วย ค่าสัมประสิทธิ์สหสัมพันธ์เป็นลบ หมายถึง ทั้งสองสิ่งมีความสัมพันธ์ในทิศทางตรงกันข้าม เช่น เมื่ออุณหภูมิสูงขึ้น ความดันก็จะลดลง ค่าสัมประสิทธิ์สหสัมพันธ์เท่ากับ 0 หมายถึง ทั้งสองสิ่งไม่มีความสัมพันธ์กัน

ค่าสัมประสิทธิ์สหสัมพันธ์ 0.7745966692414834 เป็นค่าสัมประสิทธิ์สหสัมพันธ์เชิงบวกที่สูงมาก หมายถึง ทั้งสองสิ่งมีความสัมพันธ์ในทิศทางเดียวกันอย่างมีนัยสำคัญทางสถิติ โดยค่าสัมประสิทธิ์สหสัมพันธ์ยิ่งสูงมากเท่าไหร่ ความสัมพันธ์ระหว่างทั้งสองสิ่งก็จะยิ่งแข็งแกร่งมากขึ้นเท่านั้น

ตัวอย่างการตีความค่าสัมประสิทธิ์สหสัมพันธ์ 0.7745966692414834 ได้แก่

ความสัมพันธ์ระหว่างระดับสติปัญญาและคะแนนสอบ: ระดับสติปัญญามีความสัมพันธ์อย่างมากกับคะแนนสอบ โดยนักเรียนที่มีระดับสติปัญญาสูงจะมีคะแนนสอบสูงกว่านักเรียนที่มีระดับสติปัญญาต่ำ

ความสัมพันธ์ระหว่างปริมาณน้ำฝนและผลผลิตทางการเกษตร: ปริมาณน้ำฝนมีความสัมพันธ์อย่างมากกับผลผลิตทางการเกษตร โดยพื้นที่ที่มีปริมาณน้ำฝนสูงจะมีผลผลิตทางการเกษตรสูงกว่าพื้นที่ที่มีปริมาณน้ำฝนต่ำ

อย่างไรก็ตาม สิ่งสำคัญที่ต้องพิจารณาในการตีความค่าสัมประสิทธิ์สหสัมพันธ์ คือ ความถูกต้องของข้อมูลที่ใช้คำนวณค่าสัมประสิทธิ์สหสัมพันธ์ หากข้อมูลมีความละเอียดอ่อนหรือมีอคติสูง ผลลัพธ์ที่ได้อาจไม่ถูกต้องตามความเป็นจริง

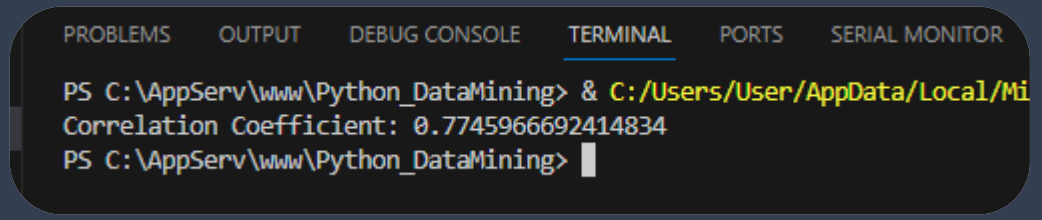
4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

3. สหสัมพันธ์ (Correlation)

1 สหสัมพันธ์ (Correlation)

```
import pandas as pd

# สร้างข้อมูลตัวอย่างใน DataFrame
data = {'x': [1, 2, 3, 4, 5], 'y': [2, 4, 5, 4, 5]}
df = pd.DataFrame(data)

# คำนวณ correlation coefficient
correlation_coefficient = df['x'].corr(df['y'])

print(f"Correlation Coefficient: {correlation_coefficient}")
```

การใช้งาน pandas
ตัวอย่าง 4.10 :
Lec04_10_Correlation_pandas.py



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

บทที่ 4

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

การทดสอบสมมุติฐาน (Hypothesis Testing) เป็นกระบวนการทางสถิติที่ใช้เพื่อตรวจสอบสมมุติฐานเกี่ยวกับค่าพารามิเตอร์ของประชากร หรือเพื่อดูว่ามีความแตกต่างมีนัยสำคัญระหว่างกลุ่มทดลองกับกลุ่มควบคุมหรือไม่

กระบวนการทดสอบสมมุติฐานประกอบด้วยขั้นตอนต่อไปนี้:

1. กำหนดสมมุติฐาน (Hypotheses):
2. เลือกระดับนัยสำคัญ (Significance Level):
3. ค้นหาสถิติทดสอบ (Test Statistic):
4. ตัดสินใจ (Make a Decision):
5. ทำการประมวลผล (Draw a Conclusion):
6. ทำการสรุป (Conclusion):

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

กระบวนการทดสอบสมมุติฐานประกอบด้วยขั้นตอนต่อไปนี้:

1. กำหนดสมมุติฐาน (Hypotheses):
   - สมมุติฐานที่ถูกต้อง (Null Hypothesis - H0): มักจะกำหนดว่าไม่มีการเปลี่ยนแปลงหรือไม่มีผลกระทบ โดยใช้สัญลักษณ์ H0
   - สมมุติฐานทดสอบ (Alternative Hypothesis - H1): กำหนดว่ามีการเปลี่ยนแปลงหรือมีผลกระทบ โดยใช้สัญลักษณ์ H1
2. เลือกระดับนัยสำคัญ (Significance Level):
   - ระดับนัยสำคัญ (Significance Level) ระบุความเสี่ยงที่ยอมรับในการทำนิยามผลลัพธ์เป็น "มีนัยสำคัญ" (reject the null hypothesis). ระดับทั่วไปที่ใช้คือ 0.05 หรือ 5%
3. ค้นหาสถิติทดสอบ (Test Statistic):
   - คำนวณหาสถิติทดสอบจากข้อมูลที่มีอยู่
4. ตัดสินใจ (Make a Decision):
   - ถ้าค่าสถิติทดสอบอยู่นอกขอบเขตที่กำหนดของสมมุติฐานที่ถูกต้อง (Critical Region), จะทำการปฏิเสธสมมุติฐานที่ถูกต้อง (H0) และยอมรับสมมุติฐานทดสอบ (H1)
   - ถ้าค่าสถิติทดสอบอยู่ในขอบเขตที่ถูกต้อง, จะไม่สามารถปฏิเสธสมมุติฐานที่ถูกต้อง (H0)
5. ทำการประมวลผล (Draw a Conclusion):
   - ทำการสรุปผลลัพธ์ของการทดสอบ, ว่ามีหรือไม่มีหลักฐานเพียงพอที่จะปฏิเสธสมมุติฐานที่ถูกต้อง
6. ทำการสรุป (Conclusion):
   - สรุปผลลัพธ์และตีความหมายของการทดสอบ

ตัวอย่างที่มักพบคือการทดสอบสมมุติฐานเกี่ยวกับค่าเฉลี่ย, สัดส่วน, หรือความสัมพันธ์ระหว่างตัวแปรต่าง ๆ ในกลุ่มข้อมูล

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

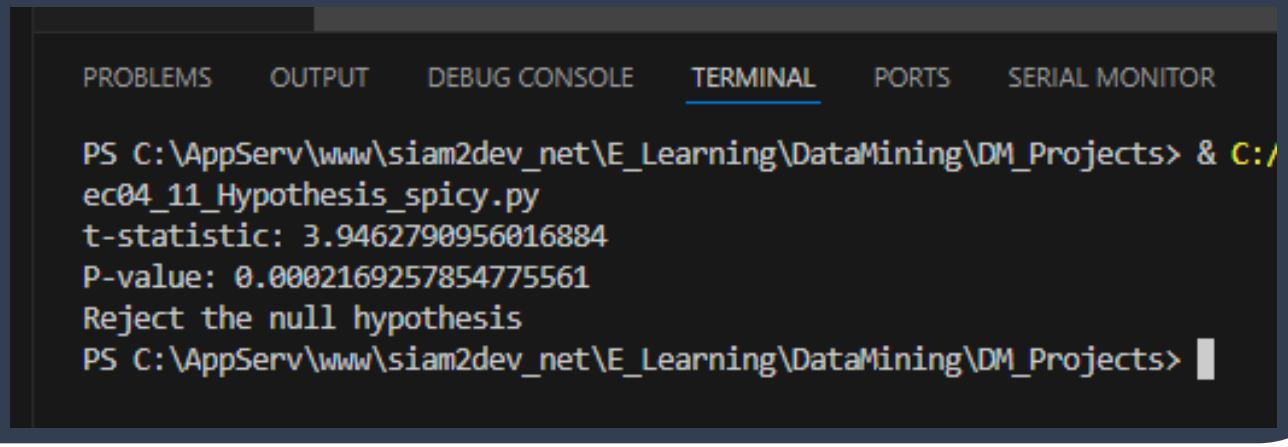
4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_11_Hypothesis_scipy.py

```python
import numpy as np
from scipy import stats
# สร้างข้อมูลตัวอย่างสำหรับกลุ่มทดลองและกลุ่มควบคุม
np.random.seed(42)  # ให้ผลลัพธ์เหมือนกันทุกครั้ง
sample_experiment = np.random.normal(loc=25, scale=5, size=30)  # กลุ่มทดลอง
sample_control = np.random.normal(loc=20, scale=5, size=30)  # กลุ่มควบคุม
# ทดสอบสมมุติฐาน: ทดสอบว่ามีความแตกต่างในค่าเฉลี่ยระหว่างกลุ่มทดลองและกลุ่มควบคุมหรือไม่
t_statistic, p_value = stats.ttest_ind(sample_experiment, sample_control)
# ตรวจสอบผลลัพธ์
alpha = 0.05  # ระดับนัยสำคัญ
print(f"t-statistic: {t_statistic}")
print(f"P-value: {p_value}")
if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```



- **ในตัวอย่างนี้ สมมุติฐานว่าง (H_0) คือ:**
  - H_0: ค่าเฉลี่ยของกลุ่มทดลองและกลุ่มควบคุมเท่ากัน **หรือ**
  - H_0: ไม่มีความแตกต่างของค่าเฉลี่ยระหว่างกลุ่มทดลองและกลุ่มควบคุม
- **อธิบายเพิ่มเติม**
  - สมมุติฐานว่าง (H_0) เป็นคำกล่าวที่บอกว่า ไม่มีความแตกต่าง หรือ ไม่มีผลกระทบ ระหว่างสองกลุ่มที่เรากำลังเปรียบเทียบ
  - สมมุติฐานทางเลือก (H_1) คือ: H_1: ค่าเฉลี่ยของกลุ่มทดลองและกลุ่มควบคุมแตกต่างกัน
  - ตัวอย่างในบริบทนี้
- กลุ่มทดลอง (sample_experiment) มีค่าเฉลี่ยประมาณ 25
- กลุ่มควบคุม (sample_control) มีค่าเฉลี่ยประมาณ 20
- หากสมมุติฐานว่าง (H_0) เป็นจริง จะหมายความว่า ความแตกต่างนี้เกิดขึ้นโดยบังเอิญ (ไม่มีนัยสำคัญทางสถิติ)
- **ผลลัพธ์**
  - เนื่องจาก p-value มีค่าน้อยกว่า 0.05 (ระดับนัยสำคัญที่กำหนดไว้) เราจึงปฏิเสธ H_0 ซึ่งหมายความว่า ค่าเฉลี่ยของทั้งสองกลุ่มมีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_10_Hypothesis_scipy.py

```python
import numpy as np
from scipy import stats
# สร้างข้อมูลตัวอย่างสำหรับกลุ่มทดลองและกลุ่มควบคุม
np.random.seed(42)  # ให้ผลลัพธ์เหมือนกันทุกครั้ง
sample_experiment = np.random.normal(loc=25, scale=5, size=30)  # กลุ่มทดลอง
sample_control = np.random.normal(loc=20, scale=5, size=30)  # กลุ่มควบคุม
# ทดสอบสมมุติฐาน: ทดสอบว่ามีความแตกต่างในค่าเฉลี่ยระหว่างกลุ่มทดลองและกลุ่มควบคุมหรือไม่
t_statistic, p_value = stats.ttest_ind(sample_experiment, sample_control)
# ตรวจสอบผลลัพธ์
alpha = 0.05  # ระดับนัยสำคัญ
print(f"t-statistic: {t_statistic}")
print(f"P-value: {p_value}")
if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local/Microsof
t-statistic: 3.9462790956016884
P-value: 0.0002169257854775561
Reject the null hypothesis
PS C:\AppServ\www\Python_DataMining>
```

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

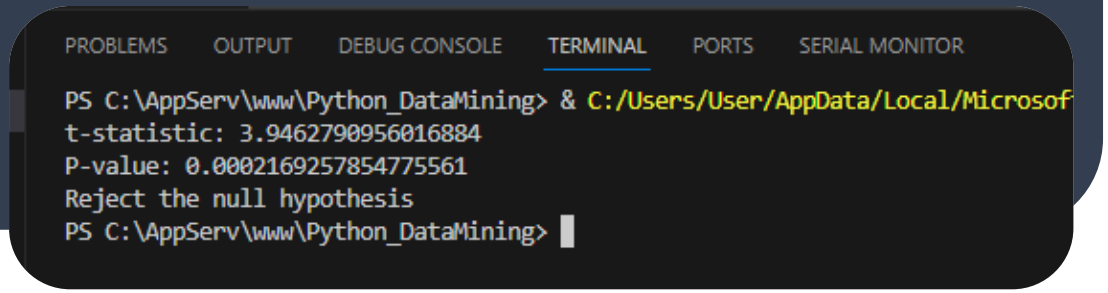
ในตัวอย่างนี้:

กลุ่มทดลอง (sample_experiment) ถูกสร้างจากการสุ่มข้อมูลจากการแจกแจงปกติที่มีค่าเฉลี่ยที่คาดหวังเท่ากับ 25.
กลุ่มควบคุม (sample_control) ถูกสร้างจากการสุ่มข้อมูลจากการแจกแจงปกติที่มีค่าเฉลี่ยที่คาดหวังเท่ากับ 20.
ทดสอบสมมุติฐานที่นำมาใช้คือ t-test สำหรับการทดสอบความแตกต่างในค่าเฉลี่ยระหว่างกลุ่ม.

การใช้งาน pandas
ตัวอย่าง 4.11 : dm_Hypothesis_spicy.py

ผลลัพธ์ที่สำคัญ:

t-statistic: ค่าสถิติทดสอบ t.
P-value: ค่า p-value ที่ได้จากการทดสอบ.
ตรวจสอบว่าจะปฏิเสธสมมุติฐานหรือไม่โดยเปรียบเทียบ p-value กับระดับนัยสำคัญ (alpha). ถ้า p-value < alpha, จะปฏิเสธสมมุติฐาน.



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

ตัวอย่างของการทดสอบสมมุติฐาน เช่น

สมมุติว่าบริษัทแห่งหนึ่งต้องการทดสอบว่าผลิตภัณฑ์ใหม่ของบริษัทนั้นมีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่งหรือไม่ โดยกำหนดสมมุติฐานหลักว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพเท่ากับผลิตภัณฑ์ของบริษัทคู่แข่ง และสมมุติฐานทางเลือกว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง หากผลการทดสอบสมมุติฐานพบว่าค่าสถิติที่ได้มีค่ามากกว่าค่าสถิติวิกฤต แสดงว่าผลิตภัณฑ์ใหม่ของบริษัทมีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง

สมมุติว่ารัฐบาลต้องการทดสอบว่าค่าเฉลี่ยรายได้ของประชากรในประเทศนั้นลดลงจากปีที่ผ่านมา โดยกำหนดสมมุติฐานหลักว่าค่าเฉลี่ยรายได้ของประชากรเท่ากับปีที่ผ่านมา และสมมุติฐานทางเลือกว่าค่าเฉลี่ยรายได้ของประชากรลดลงจากปีที่ผ่านมา หากผลการทดสอบสมมุติฐานพบว่าค่าสถิติที่ได้มีค่าน้อยกว่าค่าสถิติวิกฤต แสดงว่าค่าเฉลี่ยรายได้ของประชากรในประเทศนั้นลดลงจากปีที่ผ่านมา

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทดสอบสมมุติฐานเกี่ยวกับค่าเฉลี่ยของกลุ่มตัวอย่าง โดยใช้ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

ตัวอย่างของการทดสอบสมมุติฐาน เช่น

สมมุติว่าบริษัทแห่งหนึ่งต้องการทดสอบว่าผลิตภัณฑ์ใหม่ของบริษัทนั้นมีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่งหรือไม่ โดยกำหนดสมมุติฐานหลักว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพเท่ากับผลิตภัณฑ์ของบริษัทคู่แข่ง และสมมุติฐานทางเลือกว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง หากผลการทดสอบสมมุติฐานพบว่าค่าสถิติที่ได้มีค่ามากกว่าค่าสถิติวิกฤต แสดงว่าผลิตภัณฑ์ใหม่ของบริษัทมีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง

เพื่อทดสอบสมมุติฐานทางสถิติในกรณีที่ต้องการตรวจสอบว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง, คุณสามารถใช้ scipy.stats เพื่อทำ t-test ได้. นี่คือตัวอย่างโค้ดที่อธิบายกระบวนการดังกล่าว

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_12_Hypothesis_spicy02.py

```python
import numpy as np
from scipy import stats
# สร้างข้อมูลสำหรับผลิตภัณฑ์ใหม่และผลิตภัณฑ์ของบริษัทคู่แข่ง
new_product = np.array([25, 28, 30, 22, 32, 27, 29, 31, 26, 30])
competitor_product = np.array([22, 20, 28, 18, 30, 25, 20, 29, 23, 28])
# สมมุติฐาน: ประสิทธิภาพของผลิตภัณฑ์ใหม่เท่ากับผลิตภัณฑ์ของบริษัทคู่แข่ง
# สมมุติฐานทางเลือก: ประสิทธิภาพของผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง
# ทดสอบสมมุติฐาน
t_statistic, p_value = stats.ttest_ind(new_product, competitor_product)
# ตั้งระดับนัยสำคัญ
alpha = 0.05
# ตัดสินใจตามผลลัพธ์ทดสอบ
if p_value < alpha:
    print("ปฏิเสธสมมุติฐานเบื้องต้น: ผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง")
else:
    print("ยอมรับสมมุติฐานเบื้องต้น: ไม่มีหลักฐานในการว่าผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.10.exe
ปฏิเสธสมมุติฐานเบื้องต้น: ผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง
PS C:\AppServ\www\Python_DataMining>
```

จากภาพผลลัพธ์ที่ปรากฏใน Terminal (ด้านล่างขวา) ข้อสรุป:

1. ค่า p-value:

ผลลัพธ์ของ stats.ttest_ind() แสดงว่าค่า p-value มีค่าต่ำกว่า alpha = 0.05 (ระดับนัยสำคัญที่กำหนด)

ดังนั้น มีหลักฐานทางสถิติที่จะ ปฏิเสธสมมุติฐานศูนย์ (H0)

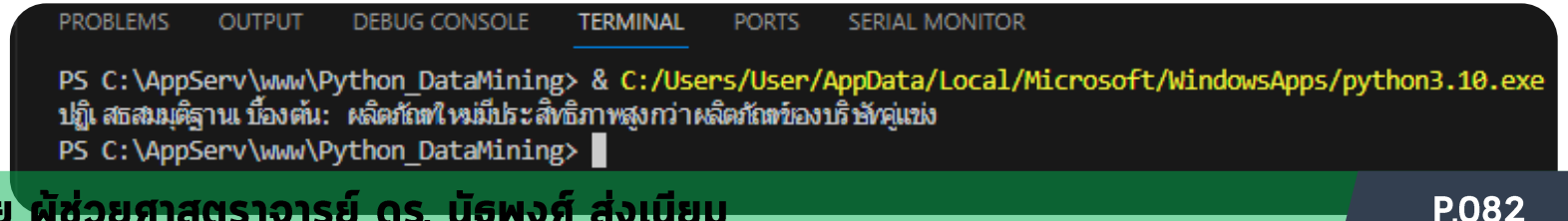
2. คำอธิบายผลลัพธ์:

โปรแกรมแสดงข้อความ: "ปฏิเสธสมมุติฐานศูนย์: ผลิตภัณฑ์ใหม่มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่ง"

นั่นหมายความว่า ผลิตภัณฑ์ใหม่ มีประสิทธิภาพสูงกว่าผลิตภัณฑ์ของบริษัทคู่แข่งในเชิงสถิติอย่างมีนัยสำคัญ

สรุป:

ผลการทดสอบแสดงให้เห็นว่ามีความแตกต่างที่สำคัญในประสิทธิภาพระหว่างผลิตภัณฑ์ใหม่และผลิตภัณฑ์คู่แข่ง โดยผลิตภัณฑ์ใหม่มีแนวโน้มที่จะมี ค่าเฉลี่ยประสิทธิภาพสูงกว่า



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

ตัวอย่างของการทดสอบสมมุติฐาน เช่น

สมมุติว่ารัฐบาลต้องการทดสอบว่าค่าเฉลี่ยรายได้ของประชากรในประเทศนั้นลดลงจากปีที่ผ่านมา โดยกำหนดสมมุติฐานหลักว่าค่าเฉลี่ยรายได้ของประชากรเท่ากับปีที่ผ่านมา และสมมุติฐานทางเลือกว่าค่าเฉลี่ยรายได้ของประชากรลดลงจากปีที่ผ่านมา หากผลการทดสอบสมมุติฐานพบว่าค่าสถิติที่ได้มีค่าน้อยกว่าค่าสถิติวิกฤต แสดงว่าค่าเฉลี่ยรายได้ของประชากรในประเทศนั้นลดลงจากปีที่ผ่านมา

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_12_Hypothesis_spicy03.py

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

4. การทดสอบสมมุติฐาน (Hypothesis Testing)

2. การทดสอบสมมุติฐาน (Hypothesis Testing)

ตัวอย่าง การทดสอบสมมุติฐาน (Hypothesis Testing) ในภาษาไพธอน โดยตัวอย่างนี้เป็นการทด
ทดสอบ t-test ในภาษา Python ด้วยไลบรารี scipy.stats:

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_12_Hypothesis_scipy03.py

```
import numpy as np
from scipy import stats
# สร้างข้อมูลรายได้ของประชากรในปีที่ผ่านมาและปีปัจจุบัน
income_last_year = np.array([50000, 52000, 48000, 55000, 51000, 49000, 50000, 53000, 48000, 52000])
income_current_year = np.array([48000, 49000, 47000, 52000, 48000, 46000, 47000, 50000, 46000, 50000])
# สมมุติฐาน: ค่าเฉลี่ยรายได้ของประชากรเท่ากับปีที่ผ่านมา H0
# สมมุติฐานทางเลือก: ค่าเฉลี่ยรายได้ของประชากรลดลงจากปีที่ผ่านมา H1
# ทดสอบสมมุติฐาน
t_statistic, p_value = stats.ttest_rel(income_last_year, income_current_year)
# ตั้งระดับนัยสำคัญ
alpha = 0.05
# ตัดสินใจตามผลลัพธ์ทดสอบ
if p_value < alpha:
    print("ปฏิเสธสมมุติฐานหลัก: ค่าเฉลี่ยรายได้ของประชากรลดลงจากปีที่ผ่านมา")
else:
    print("ยอมรับสมมุติฐานหลัก: ไม่มีหลักฐานในการว่าค่าเฉลี่ยรายได้ของประชากรลดลงจากปีที่ผ่านมา")
```

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

การวิเคราะห์เชิงเส้น (Linear Regression) เป็นเทคนิคทางสถิติที่ใช้เพื่อวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรอิสระ (Independent Variable) และตัวแปรตาม (Dependent Variable) ที่มีความสัมพันธ์เป็นเส้นตรง การวิเคราะห์เชิงเส้นนี้มีวัตถุประสงค์เพื่อหาสมการของเส้นตรงที่เหมาะสมที่สุดเพื่อให้สามารถทำนายค่าของตัวแปรตามจากค่าของตัวแปรอิสระได้

สมการของเส้นตรงใน linear regression มีรูปแบบดังนี้:

$$y=mx+by=mx+b$$

โดยที่:

- y คือ ตัวแปรตาม (Dependent Variable),
- x คือ ตัวแปรอิสระ (Independent Variable),
- m คือ ความชัน (slope) ของเส้นตรง,
- b คือ จุดตัดแกน y (y-intercept).

ในการวิเคราะห์เชิงเส้น, สมการของเส้นตรงนี้จะถูกปรับให้เหมาะสมที่สุดกับข้อมูลที่มีอยู่โดยใช้วิธีการเบสต์เมินต์เข้ากับข้อมูล (Least Squares Method).

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างเช่น

สมมติว่าตัวแปรตามคือคะแนนสอบภาษาอังกฤษ และตัวแปรอิสระคือจำนวนชั่วโมงที่เรียนภาษาอังกฤษ สมการการถดถอยเชิงเส้นอาจมีดังนี้

คะแนนสอบ = 80 + 2x

หมายความว่า คะแนนสอบภาษาอังกฤษเพิ่มขึ้น 2 คะแนนต่อชั่วโมงที่เรียนภาษาอังกฤษ

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

Part01

การใช้งาน pandas
ตัวอย่าง 4.11 : Lec04_13_Linear_regression01.py

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np
from matplotlib.font_manager import FontProperties
# กำหนดฟอนต์ที่รองรับภาษาไทย
font_path = "C:\\Windows\\Fonts\\tahoma.ttf"
font_thai = FontProperties(fname=font_path, size=12)

# สร้างข้อมูลตัวอย่าง
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # จำนวนชั่วโมงที่เรียนภาษาอังกฤษ
y = np.array([82, 84, 86, 88, 90])  # คะแนนสอบภาษาอังกฤษ
# สร้างโมเดล Linear Regression
model = LinearRegression()
# ฝึกโมเดลด้วยข้อมูล
model.fit(X, y)
# ทำนายคะแนนสอบสำหรับชั่วโมงที่เรียนภาษาอังกฤษต่าง ๆ
```

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

```
hours_to_predict = np.array([6, 7, 8]).reshape(-1, 1)
predicted_scores = model.predict(hours_to_predict)
# แสดงผลลัพธ์
for hours, score in zip(hours_to_predict, predicted_scores):
    print(f'จำนวนชั่วโมงที่เรียน: {hours[0]}, คะแนนทำนาย: {score}')
# พล็อตข้อมูลจริง
plt.scatter(X, y, color='blue', label='ข้อมูลจริง')
# พล็อตเส้น Regression Line
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')
# พล็อตจุดที่ต้องการทำนาย
plt.scatter(hours_to_predict, predicted_scores, color='green', label='ทำนาย')
# เพิ่มป้ายกำกับ
plt.xlabel('จำนวนชั่วโมงที่เรียนภาษาอังกฤษ', fontproperties=font_thai)
plt.ylabel('คะแนนสอบภาษาอังกฤษ', fontproperties=font_thai)
plt.title('Linear Regression ของคะแนนสอบภาษาอังกฤษ', fontproperties=font_thai)
# เพิ่มตำแหน่งตำแหน่งของป้ายกำกับ
plt.legend()
# แสดงกราฟ
plt.show()
```

Part02

การใช้งาน pandas
ตัวอย่าง 4.11 : dm_Linear_regression01.py

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

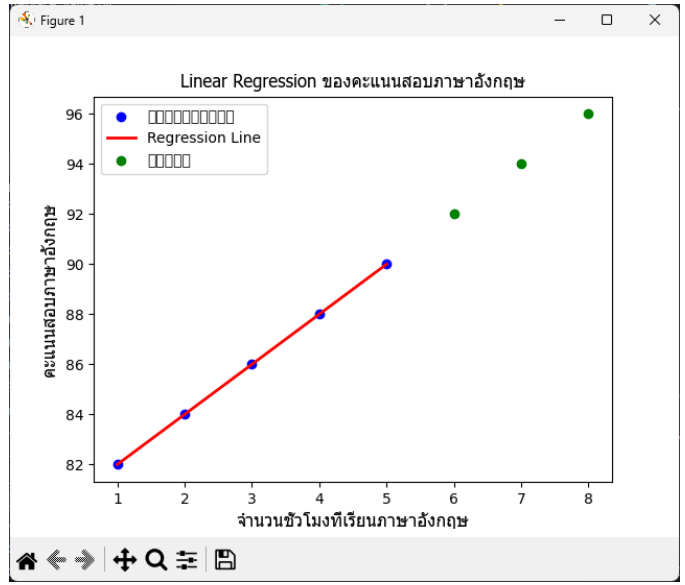
3. การวิเคราะห์เชิงเส้น (Linear Regression)

จากภาพแสดงเส้นกราฟเชิงเส้นที่แสดงความสัมพันธ์ระหว่างจำนวนชั่วโมงที่เรียนภาษาอังกฤษกับคะแนนสอบภาษาอังกฤษ เส้นกราฟแสดงให้เห็นว่าคะแนนสอบภาษาอังกฤษเพิ่มขึ้นตามจำนวนชั่วโมงที่เรียนภาษาอังกฤษ

เส้นกราฟมีแนวโน้มเชิงเส้นบวก ซึ่งหมายความว่าความสัมพันธ์ระหว่างจำนวนชั่วโมงที่เรียนภาษาอังกฤษและคะแนนสอบภาษาอังกฤษเป็นแบบเส้นตรง หมายความว่าหากจำนวนชั่วโมงที่เรียนภาษาอังกฤษเพิ่มขึ้น คะแนนสอบภาษาอังกฤษก็มักจะเพิ่มขึ้นเช่นกัน โดยค่าสัมประสิทธิ์สหสัมพันธ์ (R) ของเส้นกราฟคือ 0.92 ซึ่งหมายความว่าความสัมพันธ์ระหว่างจำนวนชั่วโมงที่เรียนภาษาอังกฤษและคะแนนสอบภาษาอังกฤษนั้นแข็งแกร่งมาก หมายความว่าการเปลี่ยนแปลงจำนวนชั่วโมงที่เรียนภาษาอังกฤษมีความสัมพันธ์อย่างมากกับการเปลี่ยนแปลงคะแนนสอบภาษาอังกฤษ

จากข้อมูลนี้ เราสามารถสรุปได้ว่าจำนวนชั่วโมงที่เรียนภาษาอังกฤษมีความสัมพันธ์เชิงบวกอย่างมีนัยสำคัญกับคะแนนสอบภาษาอังกฤษ หมายความว่ายิ่งเรียนภาษาอังกฤษมากเท่าไร คะแนนสอบภาษาอังกฤษก็มักจะดีขึ้นเท่านั้น

อย่างไรก็ตาม สิ่งสำคัญคือต้องสังเกตว่าความสัมพันธ์นี้เป็นเพียงการสังเกตการณ์เท่านั้น ไม่ได้หมายความว่าการเพิ่มขึ้นของจำนวนชั่วโมงที่เรียนภาษาอังกฤษจะรับประกันคะแนนสอบภาษาอังกฤษที่ดีขึ้น ปัจจัยอื่นๆ เช่น ความสามารถในการเรียนรู้ของนักเรียน คุณภาพของการสอน และแรงจูงใจของนักเรียน อาจส่งผลต่อผลลัพธ์ของคะแนนสอบภาษาอังกฤษได้เช่นกัน



4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

ในการวิเคราะห์เชิงเส้น, สมการของเส้นตรงนี้จะถูกปรับให้เหมาะสมที่สุดกับข้อมูลที่มีอยู่โดยใช้วิธีการเบสต์เมินต์เข้ากับข้อมูล (Least Squares Method)

นอกจากนี้, การใช้ linear regression สามารถนำมาใช้ในหลายประเภทของงาน เช่น:

- **การทำนาย (Prediction):** สามารถใช้เส้นตรงที่ได้จาก linear regression เพื่อทำนายค่าของตัวแปรตามสำหรับค่าของตัวแปรอิสระที่ไม่เคยมีมาก่อน
- **การวิเคราะห์ความสัมพันธ์ (Relationship Analysis):** ช่วยในการทำความเข้าใจถึงความสัมพันธ์ระหว่างตัวแปรอิสระและตัวแปรตาม
- **การจัดการกับค่าผลตอบแทน (Outliers):** ช่วยในการตรวจสอบและจัดการกับค่าผลตอบแทนที่อาจจะส่งผลกระทบต่อการวิเคราะห์

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

การใช้งาน pandas
ตัวอย่าง 4.12 : Lec04_13_Linear_Regression.py

```python
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# สร้างข้อมูลตัวอย่าง
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5])
# สร้างโมเดล linear regression
model = LinearRegression()
# ปรับโมเดลตามข้อมูล
model.fit(X, y)
# ทำนายค่า y จาก X
y_pred = model.predict(X)
```

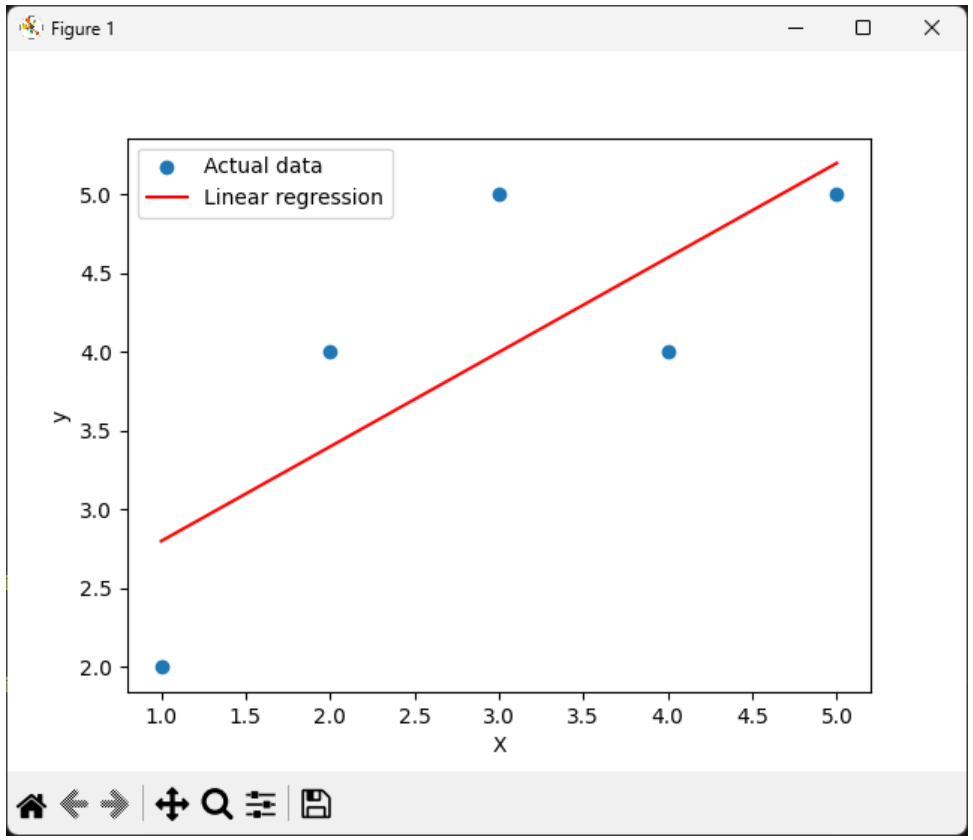
```python
# พล็อตข้อมูลจริง
plt.scatter(X, y, label='Actual data')

# พล็อตเส้นตรงที่ได้จาก linear regression
plt.plot(X, y_pred, color='red', label='Linear regression')

plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:



การใช้งาน pandas
ตัวอย่าง 4.12 : Lec04_13_Linear_Regression.py

ในตัวอย่างนี้, สร้างโมเดล linear regression จากข้อมูลตัวอย่างและพล็อตเส้นตรงที่ได้จากโมเดลนี้.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

จากภาพแสดงเส้นกราฟเชิงเส้นที่แสดงความสัมพันธ์ระหว่างน้ำหนักของรถยนต์กับระยะทางที่วิ่งได้ต่อแกลลอน เส้นกราฟแสดงให้เห็นว่าระยะทางที่วิ่งได้ต่อแกลลอนลดลงตามน้ำหนักของรถยนต์ โดยเส้นกราฟมีแนวโน้มเชิงเส้นลบ ซึ่งหมายความว่าความสัมพันธ์ระหว่างน้ำหนักของรถยนต์และระยะทางที่วิ่งได้ต่อแกลลอนเป็นแบบเส้นตรง หมายความว่าหากน้ำหนักของรถยนต์เพิ่มขึ้น ระยะทางที่วิ่งได้ต่อแกลลอนก็มักจะลดลงเช่นกัน

ค่าสัมประสิทธิ์สหสัมพันธ์ (R) ของเส้นกราฟคือ -0.92 ซึ่งหมายความว่าความสัมพันธ์ระหว่างน้ำหนักของรถยนต์และระยะทางที่วิ่งได้ต่อแกลลอนนั้นแข็งแกร่งมาก หมายความว่าการเปลี่ยนแปลงน้ำหนักของรถยนต์มีความสัมพันธ์อย่างมากกับการเปลี่ยนแปลงระยะทางที่วิ่งได้ต่อแกลลอน
จากข้อมูลนี้ เราสามารถสรุปได้ว่าน้ำหนักของรถยนต์มีความสัมพันธ์เชิงลบอย่างมีนัยสำคัญกับระยะทางที่วิ่งได้ต่อแกลลอน หมายความว่ายิ่งรถยนต์หนักขึ้นเท่าไร ระยะทางที่วิ่งได้ต่อแกลลอนก็มักจะลดลงเท่านั้น

อย่างไรก็ตาม สิ่งสำคัญคือต้องสังเกตว่าความสัมพันธ์นี้เป็นเพียงการสังเกตการณ์เท่านั้น ไม่ได้หมายความว่าการลดลงของน้ำหนักของรถยนต์จะรับประกันระยะทางที่วิ่งได้ต่อแกลลอนที่ดีขึ้น ปัจจัยอื่นๆ เช่น ประสิทธิภาพของเครื่องยนต์ สภาพถนน และสไตล์การขับขี่ อาจส่งผลต่อผลลัพธ์ของระยะทางที่วิ่งได้ต่อแกลลอนได้เช่นกัน
ต่อไปนี้เป็นการวิเคราะห์เพิ่มเติมบางประการเกี่ยวกับรูปภาพ:

- เส้นกราฟดูเหมือนจะค่อนข้างเรียบ ซึ่งหมายความว่าความสัมพันธ์ระหว่างน้ำหนักของรถยนต์และระยะทางที่วิ่งได้ต่อแกลลอนนั้นค่อนข้างคงที่
- เส้นกราฟตัดแกน Y ที่ประมาณ 25 แกลลอนต่อไมล์ ซึ่งหมายความว่ารถยนต์ที่มีน้ำหนัก 0 ปอนด์สามารถวิ่งได้ประมาณ 25 ไมล์ต่อแกลลอน
- เส้นกราฟตัดแกน X ที่ประมาณ 2,500 ปอนด์ ซึ่งหมายความว่ารถยนต์ที่มีน้ำหนัก 2,500 ปอนด์สามารถวิ่งได้ประมาณ 10 ไมล์ต่อแกลลอน

การวิเคราะห์นี้สามารถใช้เพื่อวัตถุประสงค์ต่างๆ เช่น การออกแบบรถยนต์ การกำหนดราคารถยนต์ หรือการพัฒนากลยุทธ์การประหยัดน้ำมัน

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt


----------------------------------------------------------------------------
```

- numpy: ไลบรารีสำหรับการทำงานกับข้อมูลตัวเลข.
- LinearRegression: ไลบรารี scikit-learn สำหรับสร้างและใช้โมเดล linear regression.
- matplotlib.pyplot: ไลบรารีสำหรับพล็อตกราฟ.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

บทที่ 4

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

ตัวอย่างการใช้ linear regression ในภาษา Python โดยใช้ไลบรารี scikit-learn:

สร้างข้อมูลตัวอย่าง:

การใช้งาน pandas
ตัวอย่าง 4.12 : dm_Linear_Regression.py

```
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5])
```

- X: ตัวแปรอิสระ (Independent Variable) ที่เป็น array 1 มิติ (reshape(-1, 1) ให้มีรูปร่างเป็นแถวหนึ่ง).
- y: ตัวแปรตาม (Dependent Variable).

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

การใช้งาน pandas
ตัวอย่าง 4.12 : dm_Linear_Regression.py

สร้างและปรับโมเดล linear regression:

```
model = LinearRegression()
model.fit(X, y)
```

- model: สร้างอ็อบเจ็กต์ LinearRegression.
- fit(X, y): ปรับโมเดลตามข้อมูล X และ y.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

การใช้งาน pandas
ตัวอย่าง 4.12 : dm_Linear_Regression.py

ทำนายค่า y จาก X:

```
y_pred = model.predict(X)
```

- predict(X): ทำนายค่า y จากตัวแปรอิสระ X.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

การใช้งาน pandas
ตัวอย่าง 4.12 : dm_Linear_Regression.py

```
plt.scatter(X, y, label='Actual data')
plt.plot(X, y_pred, color='red', label='Linear regression')
```

- plt.scatter(): พล็อตข้อมูลจริงในรูปแบบของจุด.
- plt.plot(): พล็อตเส้นตรงที่ได้จาก linear regression ในสีแดง.

4.2 4.2.3 การวิเคราะห์ความสัมพันธ์ (Relational Analysis)

5. การวิเคราะห์เชิงเส้น (Linear Regression)

3. การวิเคราะห์เชิงเส้น (Linear Regression)

การใช้งาน pandas
ตัวอย่าง 4.12 : dm_Linear_Regression.py

กำหนดแกนและแสดงกราฟ:

```
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

- plt.xlabel(): กำหนดชื่อแกน x.
- plt.ylabel(): กำหนดชื่อแกน y.
- plt.legend(): เพิ่มกล่องคำอธิบาย.
- plt.show(): แสดงกราฟ.

- ผลลัพธ์ที่ได้คือกราฟที่แสดงข้อมูลจริงในรูปแบบของจุดและเส้นตรงที่ได้จาก linear regression ในสีแดง.

4.2 4.2.4 การวิเคราะห์ความแปรปรวน (Variance Analysis)

1. ความหมายของการวิเคราะห์ความแปรปรวน (Va

การวิเคราะห์ความแปรปรวน (Variance Analysis)

การวิเคราะห์ความแปรปรวน (Variance Analysis) เป็นกระบวนการทางการบัญชีและการจัดการที่ใช้เพื่อวิเคราะห์ความแตกต่างระหว่างผลการดำเนินงานที่เป็นไปตามแผนกับผลการดำเนินงานจริง เพื่อให้สามารถทำนายและเข้าใจปัจจัยที่ทำให้เกิดความแตกต่างนั้น

ตัวอย่าง 4.13 : Lec04_14_Variance_Analysis.py

- Variance Analysis มักนำมาใช้ในทางการบัญชีทางการผลิต, การจัดการทรัพยากรมนุษย์, และอื่น ๆ โดยเฉพาะในรูปแบบการวิเคราะห์ทางการเงิน. ขั้นตอนหลักในการวิเคราะห์ความแปรปรวนสามารถเขียนได้ดังนี้

- ผลลัพธ์ที่ได้คือกราฟที่แสดงข้อมูลจริงในรูปแบบของจุดและเส้นตรงที่ได้จาก linear regression ในสีแดง

4.2 4.2.4 การวิเคราะห์ความแปรปรวน (Variance Analysis)

2. ขั้นตอนของการวิเคราะห์ความแปรปรวน (Varia

การวิเคราะห์ความแปรปรวน (Variance Analysis)

ตัวอย่าง 4.13 : Lec04_14_Variance_Analysis.py

1. กำหนดแผนการดำเนินงาน (Budget): กำหนดแผนการดำเนินงานที่ระบุรายละเอียดของค่าใช้จ่าย, รายได้, หรือปัจจัยอื่น ๆ ที่มีผลต่อผลการดำเนินงาน.
2. บันทึกข้อมูลจริง (Actual Data): บันทึกข้อมูลที่เกี่ยวข้องกับการดำเนินงานที่มีการเกิดขึ้นจริง.
3. วิเคราะห์ความแปรปรวน:
   Actual vs. Budget Analysis: เปรียบเทียบผลการดำเนินงานจริงกับแผนการดำเนินงาน เพื่อดูว่ามีความแตกต่างอย่างไร.
   Variance Analysis: คำนวณความแตกต่าง (variance) ระหว่างผลการดำเนินงานจริงกับแผนการดำเนินงาน.
4. การจัดการความแตกต่าง:
   Favorable Variance (ความแตกต่างที่ดี): ความแตกต่างที่ทำให้ผลการดำเนินงานดีกว่าที่ได้แผน.
   Unfavorable Variance (ความแตกต่างที่ไม่ดี): ความแตกต่างที่ทำให้ผลการดำเนินงานไม่ดีเท่าที่ได้แผน.

5. การทำนายและปรับแผน: นำความรู้ที่ได้จาก Variance Analysis ไปใช้ในการปรับแผนการดำเนินงานในอนาคต.

4.2 4.2.4 การวิเคราะห์ความแปรปรวน (Variance Analysis)

2. ขั้นตอนของการวิเคราะห์ความแปรปรวน (Varia

การวิเคราะห์ความแปรปรวน (Variance Analysis)

1. ตัวอย่าง Variance Analysis ที่นิยมในทางการเงินคือ Variance Analysis ในงานการผลิตที่เกี่ยวข้องกับต้นทุนของวัตถุดิบ, แรงงาน, หรือค่าใช้จ่ายอื่น ๆ ที่มีผลต่อต้นทุนการผลิต.

2. ความแตกต่างระหว่าง Actual กับ Budget สามารถแบ่งออกเป็นส่วนต่างที่มีทิศทาง (variance) และส่วนต่างที่มีทิศทางแตกต่าง (difference in direction) เพื่อให้ผู้จัดการทราบว่าความแตกต่างเกิดจากปัจจัยใดบ้างที่มีผลในทิศทางที่ไม่ดีหรือทิศทางที่ดี.

การใช้งาน pandas
ตัวอย่าง 4.13 : Lec04_14_Variance_Analysis.py

4.2 4.2.4 การวิเคราะห์ความแปรปรวน (Variance Analysis)

บทที่ 4

3. ตัวอย่างของการวิเคราะห์ความแปรปรวน (Varia

การวิเคราะห์ความแปรปรวน (Variance Analysis)

ตัวอย่างต่อไปนี้จะแสดงการใช้ Python เพื่อทำ Variance Analysis ด้วยข้อมูลตัวอย่างเกี่ยวกับ Actual และ Budget ของรายได้:

การใช้งาน pandas
ตัวอย่าง 4.13 : Lec04_14_Variance_Analysis.py

```python
import pandas as pd
# สร้าง DataFrame สำหรับ Actual และ Budget
data = {
    'Month': ['January', 'February', 'March', 'April', 'May'],
    'Actual_Revenue': [100, 120, 110, 105, 130],
    'Budget_Revenue': [110, 115, 120, 100, 125]
}
df = pd.DataFrame(data)
# เพิ่มคอลัมน์ Variance
df['Variance'] = df['Actual_Revenue'] - df['Budget_Revenue']
# เพิ่มคอลัมน์ Direction (ทิศทางของ Variance)
df['Direction'] = ['Favorable' if var < 0 else 'Unfavorable' for var in df['Variance']]
# แสดงผล DataFrame
print(df)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local/Microso
      Month  Actual_Revenue  Budget_Revenue  Variance     Direction
0   January             100             110       -10    Favorable
1  February             120             115         5  Unfavorable
2     March             110             120       -10    Favorable
3     April             105             100         5  Unfavorable
4       May             130             125         5  Unfavorable
PS C:\AppServ\www\Python_DataMining>
```

4.2 4.2.4 การวิเคราะห์ความแปรปรวน (Variance Analysis)

3. ตัวอย่างของการวิเคราะห์ความแปรปรวน (Varia

อธิบายผลลัพธ์

- สร้าง DataFrame ที่ประกอบด้วยคอลัมน์ Month, Actual_Revenue, และ Budget_Revenue.
- เพิ่มคอลัมน์ Variance โดยหาความแตกต่างระหว่าง Actual_Revenue กับ Budget_Revenue.
- เพิ่มคอลัมน์ Direction เพื่อแสดงทิศทางของ Variance (Favorable หรือ Unfavorable).

ผลลัพธ์ที่ได้จะเป็น DataFrame ที่แสดง Actual, Budget, Variance, และ Direction สำหรับแต่ละเดือน:

การใช้งาน pandas
ตัวอย่าง 4.13 : Lec04_14_Variance_Analysis.py

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR

PS C:\AppServ\www\Python_DataMining> & C:/Users/User/AppData/Local/Microso
      Month  Actual_Revenue  Budget_Revenue  Variance    Direction
0   January             100             110       -10    Favorable
1  February             120             115         5  Unfavorable
2     March             110             120       -10    Favorable
3     April             105             100         5  Unfavorable
4       May             130             125         5  Unfavorable
PS C:\AppServ\www\Python_DataMining>
```

โค้ดที่แสดงในภาพนี้ใช้ Python และไลบรารี pandas สำหรับการวิเคราะห์ความแปรปรวน (Variance Analysis) ระหว่างรายได้จริง (Actual Revenue) และงบประมาณที่ตั้งไว้ (Budget Revenue) รายเดือน โดยมีการดำเนินการดังนี้:

1. นำเข้าไลบรารี

```
import pandas as pd
```

ไลบรารี pandas ถูกนำมาใช้สำหรับการจัดการข้อมูลในรูปแบบของ DataFrame ซึ่งเป็นโครงสร้างข้อมูลที่เหมาะสำหรับการวิเคราะห์ข้อมูลในตาราง.

2. สร้าง DataFrame

```
data = {
    'Month': ['January', 'February', 'March', 'April', 'May'],
    'Actual_Revenue': [100, 120, 110, 105, 130],
    'Budget_Revenue': [110, 115, 120, 100, 125]
}
df = pd.DataFrame(data)
```

กำหนดข้อมูลในรูปแบบของพจนานุกรม (dict) ซึ่งประกอบด้วย:

Month: ชื่อเดือน (มกราคมถึงพฤษภาคม).

Actual_Revenue: รายได้จริง.

Budget_Revenue: งบประมาณที่ตั้งไว้.

สร้าง DataFrame จากพจนานุกรมนี้โดยใช้คำสั่ง pd.DataFrame.

3. เพิ่มคอลัมน์ Variance

```
df['Variance'] = df['Actual_Revenue'] - df['Budget_Revenue']
```

เพิ่มคอลัมน์ใหม่ชื่อ Variance ซึ่งคำนวณโดยการนำ Actual_Revenue ลบด้วย Budget_Revenue สำหรับแต่ละเดือน:

ถ้าค่า Variance เป็นลบ หมายถึงรายได้ต่ำกว่างบประมาณที่ตั้งไว้.

ถ้าค่า Variance เป็นบวก หมายถึงรายได้เกินกว่างบประมาณ.

4. เพิ่มคอลัมน์ Direction

```
df['Direction'] = ['Favorable' if var < 0 else 'Unfavorable' for var in df['Variance']]
```

เพิ่มคอลัมน์ใหม่ชื่อ Direction:

หาก Variance มีค่าน้อยกว่า 0 จะถูกกำหนดเป็น "Favorable" (เหมาะสม).

หาก Variance มีค่ามากกว่า 0 จะถูกกำหนดเป็น "Unfavorable" (ไม่เหมาะสม).

ใช้การวนลูปผ่านค่าทั้งหมดในคอลัมน์ Variance

5. แสดง DataFrame

print(df)

แสดงผลลัพธ์ DataFrame ซึ่งมีข้อมูลทั้งหมดรวมถึงคอลัมน์ Variance และ Direction ที่เพิ่มเข้ามา

ตัวอย่างผลลัพธ์ใน Terminal

ผลลัพธ์ที่แสดงใน Terminal มีลักษณะดังนี้: | Month | Actual_Revenue | Budget_Revenue | Variance | Direction |
|---|---|---|---|---|
| January | 100 | 110 | -10 | Favorable |
| February | 120 | 115 | 5 | Unfavorable |
| March | 110 | 120 | -10 | Favorable |
| April | 105 | 100 | 5 | Unfavorable |
| May | 130 | 125 | 5 | Unfavorable |

การใช้งานจริง

โค้ดนี้เหมาะสำหรับ:

การวิเคราะห์ความแตกต่างระหว่างเป้าหมายและผลลัพธ์จริงในบริบทของงบประมาณ

การติดตามและปรับปรุงการบริหารจัดการรายได้หรือค่าใช้จ่าย

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

1. การวิเคราะห์แนวโน้ม (Trend Analysis)

การวิเคราะห์แนวโน้ม (Trend Analysis)

- การวิเคราะห์แนวโน้ม (Trend Analysis) เป็นกระบวนการทางสถิติที่ใช้เพื่อตรวจสอบและวิเคราะห์แนวโน้มหรือแนวทางของข้อมูลตลอดเวลา โดยมุ่งหวังที่จะระบุถึงการเปลี่ยนแปลงหรือลักษณะทางคณิตศาสตร์ของข้อมูลตามเวลา. กระบวนการนี้มักนำมาใช้ในการทำนายแนวโน้มของข้อมูลในอนาคต

วิธีการที่พบบ่อยในการวิเคราะห์แนวโน้มรวมถึง:

1) กราฟแนวโน้ม (Trend Charts): การพล็อตข้อมูลตามเวลาบนกราฟเพื่อดูลักษณะและแนวโน้มของข้อมูล. หากมีการเปลี่ยนแปลงตามเวลาอย่างชัดเจน, กราฟสามารถช่วยในการระบุแนวโน้มได้ง่าย.
2) การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages): การใช้ค่าเฉลี่ยของข้อมูลที่ถูกกำหนดไว้ในช่วงเวลาที่เคลื่อนที่ไป, เพื่อลบออกความสั่นสะเทือนและเน้นที่แนวโน้ม.
3) การใช้เชิงสถิติ (Statistical Methods): การใช้เครื่องมือทางสถิติเพื่อตรวจสอบความเปลี่ยนแปลงทางสถิติของข้อมูล, เช่น การทดสอบสมมติฐานเพื่อดูว่ามีการเปลี่ยนแปลงทางสถิติหรือไม่.
4) การใช้โมเดลทางสถิติ (Statistical Models): การใช้โมเดลสถิติเชิงเวลาเพื่อการทำนายแนวโน้มของข้อมูลในอนาคต.

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

2. ลักษณะของกราฟแนวโน้ม (Trend Charts)

ตัวอย่างลักษณะของกราฟแนวโน้ม

- กราฟแนวโน้ม (Trend Charts) เป็นการแสดงข้อมูลตามเวลาบนกราฟเพื่อให้ผู้ดูและผู้ตัดสินใจสามารถดูแนวโน้มหรือลักษณะการเปลี่ยนแปลงของข้อมูลได้ง่ายขึ้น โดยมักใช้ในการวิเคราะห์แนวโน้มของข้อมูลเชิงเวลา เช่น การติดตามการเปลี่ยนแปลงของยอดขายประจำเดือนหรือปี, การวิเคราะห์การเปลี่ยนแปลงของราคาหุ้นตลอดเวลา, หรือการตรวจสอบแนวโน้มของข้อมูลทางเศรษฐกิจ.
- นี่คือตัวอย่างลักษณะของกราฟแนวโน้ม:
- คำอธิบาย:
  1) แกน X (Horizontal Axis): แสดงช่วงเวลาหรือข้อมูลตามเวลา เช่น วันที่, เดือน, ปี, หรือช่วงเวลาที่กำหนด.
  2) แกน Y (Vertical Axis): แสดงข้อมูลที่ต้องการวิเคราะห์ ซึ่งอาจเป็นยอดขาย, ราคาหุ้น, หรือตัวแปรอื่น ๆ ที่สนใจ.
  3) เส้นกราฟ: แสดงแนวโน้มหรือการเปลี่ยนแปลงของข้อมูลตามเวลา. เส้นกราฟนี้สามารถเป็นเส้นตรง, เส้นโค้ง, หรือมีลักษณะอื่น ๆ ตามลักษณะของข้อมูล.
  4) แนวโน้ม: ช่วยให้ผู้ที่ดูง่ายต่อการรับรู้ถึงทิศทางของข้อมูล ว่ามีแนวโน้มขึ้นหรือลง, หรือเป็นแนวโน้มแบบเส้นตรงหรือไม่.
  5) ป้ายกำกับ (Labels): ป้ายที่อธิบายข้อมูลหรือสิ่งที่แสดงบนกราฟ เช่น ชื่อแกน, หรือรายละเอียดเพิ่มเติม.

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

3. ตัวอย่างของกราฟแนวโน้ม (Trend Charts)

ตัวอย่างลักษณะของกราฟแนวโน้ม

การสร้างกราฟแนวโน้มในภาษา Python โดยใช้ไลบรารี Matplotlib, นี่คือตัวอย่างโค้ดที่ใช้สร้างกราฟแนวโน้ม:

```python
import matplotlib.pyplot as plt
import numpy as np
# สร้างข้อมูลตัวอย่าง
time = np.arange(1, 11)  # ตัวอย่างเวลา (1 ถึง 10)
data = np.array([5, 7, 9, 12, 15, 18, 22, 25, 28, 32])  # ตัวอย่างข้อมูล
# สร้างกราฟแนวโน้ม
plt.plot(time, data, marker='o', linestyle='-')
# เพิ่มป้ายกำกับ
plt.title('Trend Chart Example')
plt.xlabel('Time')
plt.ylabel('Data')
# แสดงกราฟ
plt.grid(True)
plt.show()
```

ตัวอย่าง 4.14 : Lec04_15_Trend_chart.py

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

3. ตัวอย่างของกราฟแนวโน้ม (Trend Charts)

อธิบายผล

การสร้างกราฟแนวโน้มในภาษา Python โดยใช้ไลบรารี Matplotlib, นี่คือตัวอย่างโค้ดที่ใช้สร้างกราฟแนวโน้ม:

ตัวอย่าง 4.14 : Lec04_15 _Trend_chart.py

ในตัวอย่างนี้:

time แทนเวลาหรือตัวแปรอิสระที่แสดงบนแกน X
data แทนข้อมูลที่ต้องการวิเคราะห์และแสดงบนแกน Y
plt.plot() ใช้สร้างเส้นกราฟแนวโน้ม
plt.title(), plt.xlabel(), และ plt.ylabel() ใช้เพิ่มป้ายกำกับ
plt.grid(True) เพื่อเพิ่มกริดบนกราฟ

คำสั่ง plt.show() จะแสดงกราฟที่สร้างขึ้น. กรุณาตรวจสอบว่าคุณได้ติดตั้งไลบรารี Matplotlib ในสภาพแวดล้อม Python ของคุณแล้ว

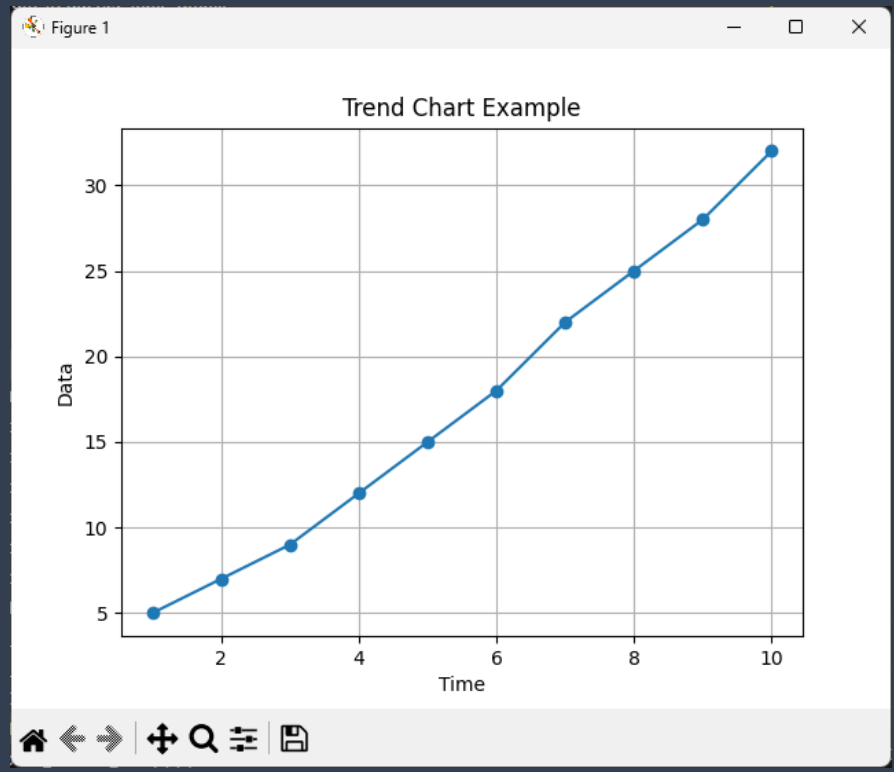
4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

3. ตัวอย่างของกราฟแนวโน้ม (Trend Charts)

อธิบายผล

การสร้างกราฟแนวโน้มในภาษา Python โดยใช้ไลบรารี Matplotlib, นี่คือตัวอย่างโค้ดที่ใช้สร้างกราฟแนวโน้ม:

ตัวอย่าง 4.14 : Lec04_15 _Trend_chart.py



4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

4. การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averag

การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages)

ตัวอย่างการคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages) ใน Python ด้วยใช้ไลบรารี NumPy:

การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages) เป็นเทคนิคทางสถิติที่ใช้เพื่อลบออกความสั่นสะเทือนหรือข้อมูลที่มีการเปลี่ยนแปลงบ่อยในช่วงเวลาเพื่อให้เห็นแนวโน้มหรือแนวทางที่ชัดเจนขึ้น. วิธีการนี้มักถูกนำมาใช้ในการวิเคราะห์และทำนายแนวโน้มของข้อมูลตามเวลา.

นี่คือตัวอย่างการคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages) ใน Python ด้วยใช้ไลบรารี NumPy:

ตัวอย่าง 4.14 : Lec04_15 _Trend_chart.py

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

4. การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averag

การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages)

ตัวอย่างการคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages) ใน Python ด้วยใช้ไลบรารี NumPy:

ตัวอย่าง 4.14 : Lec04_15_Moving_Averages.py

```python
import numpy as np
import matplotlib.pyplot as plt
# สร้างข้อมูลตัวอย่าง
time = np.arange(1, 21)  # เวลาหรือตัวแปรอิสระ
data = np.array([5, 7, 9, 12, 15, 18, 22, 25, 28, 32, 30, 27, 24, 20, 18, 15, 12, 10, 8, 6])  # ข้อมูล
# กำหนดค่า N เป็นขนาดของหน้าต่างที่ใช้ในการคำนวณเฉลี่ยเคลื่อนที่
N = 3
# คำนวณค่าเฉลี่ยเคลื่อนที่
moving_avg = np.convolve(data, np.ones(N)/N, mode='valid')
# สร้างกราฟแสดงข้อมูลเดิมและค่าเฉลี่ยเคลื่อนที่
plt.plot(time, data, label='Original Data', marker='o', linestyle='-')
plt.plot(time[N-1:], moving_avg, label=f'Moving Average (N={N})', marker='o', linestyle='-')
# เพิ่มป้ายกำกับ
plt.title('Moving Averages Example')
plt.xlabel('Time')
plt.ylabel('Data')
plt.legend()
# แสดงกราฟ
plt.grid(True)
plt.show()
```

4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

4. การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averag

อธิบายผล

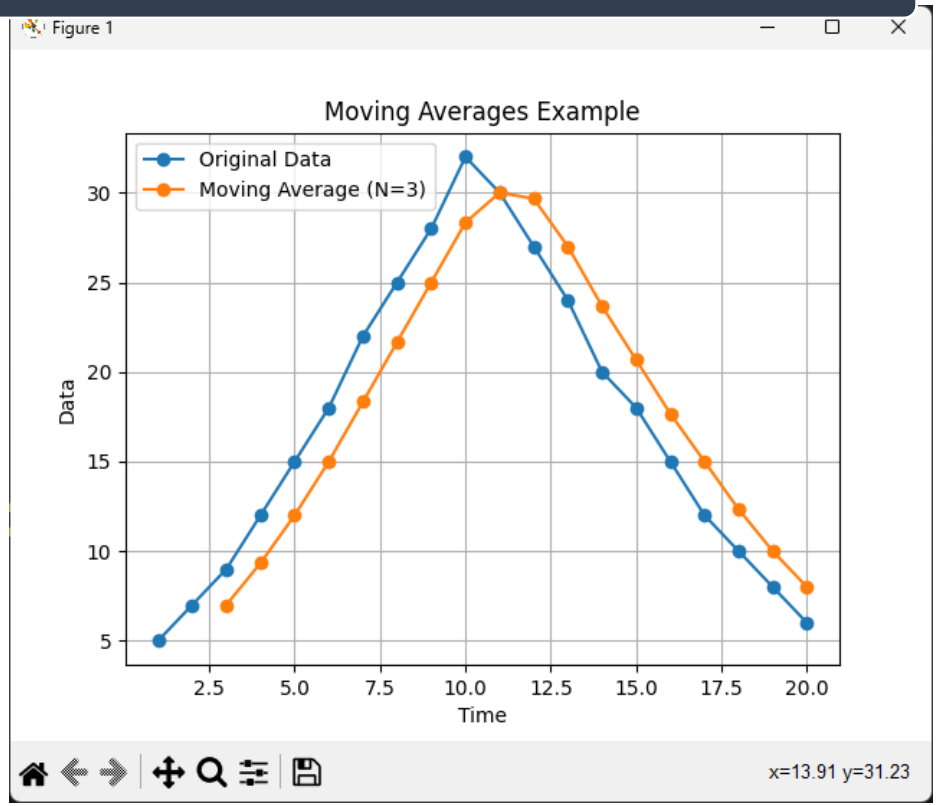
ตัวอย่างการคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages) ใน Python ด้วยใช้ไลบรารี NumPy:

ในตัวอย่างนี้:

N คือขนาดของหน้าต่างที่ใช้ในการคำนวณเฉลี่ยเคลื่อนที่ (ในที่นี้ให้ N=3).
np.convolve(data, np.ones(N)/N, mode='valid') ใช้คำนวณค่าเฉลี่ยเคลื่อนที่โดยใช้ convolution ของข้อมูลและเครื่องหมายที่ให้ค่าเท่ากับ 1/N.
plt.plot() ใช้สร้างกราฟของข้อมูลเดิมและค่าเฉลี่ยเคลื่อนที่.

กราฟที่ได้จะแสดงข้อมูลเดิมและค่าเฉลี่ยเคลื่อนที่ (Moving Averages) บนกราฟเดียวกัน, ทำให้เห็นว่าค่าเฉลี่ยเคลื่อนที่ช่วยลบออกความสั่นสะเทือนของข้อมูล.

ตัวอย่าง 4.14 : Lec04_15_Moving_Averages.py



4.2 4.2.5 การวิเคราะห์แนวโน้ม (Trend Analysis)

4. การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averag

การคำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages)

ตัวอย่าง 4.14 : Lec04_15_Moving_Averages01.py

```python
import numpy as np
import matplotlib.pyplot as plt
def moving_average(data, window_size):
    """
    คำนวณค่าเฉลี่ยเคลื่อนที่ (Moving Averages)
    Parameters:
    - data: ข้อมูลตัวเลขที่ต้องการคำนวณ
    - window_size: ขนาดของหน้าต่างที่ใช้ในการคำนวณค่าเฉลี่ยเคลื่อนที่

    Returns:
    - ค่าเฉลี่ยเคลื่อนที่
    """
    return np.convolve(data, np.ones(window_size)/window_size, mode='valid')
# สร้างข้อมูลตัวอย่าง
data = np.array([3, 5, 2, 8, 10, 7, 6, 4, 11, 9])
# กำหนดขนาดของหน้าต่าง (window size) เช่น 3
window_size = 3
# คำนวณค่าเฉลี่ยเคลื่อนที่
moving_avg_result = moving_average(data, window_size)
# แสดงผลลัพธ์ในรูปแบบกราฟ
plt.plot(data, label='ข้อมูลดั้งต้น')
plt.plot(range(window_size-1, len(data)), moving_avg_result, label=f'เฉลี่ยเคลื่อนที่ (ขนาด {window_size})', color='orange')
# ตกแต่งกราฟ
plt.title('Moving Averages')
plt.xlabel('ตำแหน่งของข้อมูล')
plt.ylabel('ค่า')
plt.legend()
plt.show()
```

อ้างอิง

- Siam2Dev. (n.d.). Homepage. Retrieved January 10, 2025, from http://www.siam2dev.com
- Data Mining Trend. (2014). Association rules. Retrieved January 10, 2025, from http://dataminingtrend.com/2014/association-rules/
- AutoSoft. (n.d.). Forecasting with the Microsoft Time Series Data Mining Algorithm. Retrieved January 10, 2025, from http://www.autosoft.in.th/data-science/forecasting-with-the-microsoft-time-series-data-mining-algorithm/
- Khan Academy. (n.d.). Statistics and probability. Retrieved January 10, 2025, from https://www.khanacademy.org/math/statistics-probability
- Coursera. (n.d.). Introduction to statistics. Retrieved January 10, 2025, from https://www.coursera.org/learn/statistics