รายงาน

เรื่อง

Machine Learning และภาษาไพธอนสำหรับวิทยาศาสตร์ข้อมูล

จัดทำโดย

นายกฤษณพงษ์ เพ็งบุญ 6330300038

นายจิรเมธ สุทธาวาณิชย์ 6330300119

นายชญานนท์ พูลวาสน์ 6330300151

นายชญานิน ตลับเงิน 6330300160

เสนอ

ผศ.ดร.กุลวดี สมบูรณ์วิวัฒน์

รายงานนี้เป็นส่วนหนึ่งของรายวิชา

03603351 วิทยาศาสตร์ข้อมูลเบื้องต้นหมู่เรียนบรรยาย 800

ภาคต้น ปีการศึกษา 2565

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา 03603351 วิทยาศาสตร์ข้อมูลเบื้องต้นเพื่อใช้ในการศึกษาค้นคว้าเกี่ยวกับ Machine Learning และภาษาไพธอนสำหรับวิทยาศาสตร์ข้อมูล

ทางผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อื่นที่สนใจในเรื่อง Machine Learning และภาษาไพธอนสำหรับวิทยาศาสตร์ข้อมูล หากมีข้อผิดพลาดประการใด ทางผู้จัดทำก็ขออภัยมา ณ ที่นี้ด้วย

ผู้จัดทำ

นายกฤษณพงษ์ เพ็งบุญ 6330300038

นายจิรเมธ สุทธาวาณิชย์ 6330300119

นายชญานนท์ พูลวาสน์ 6330300151

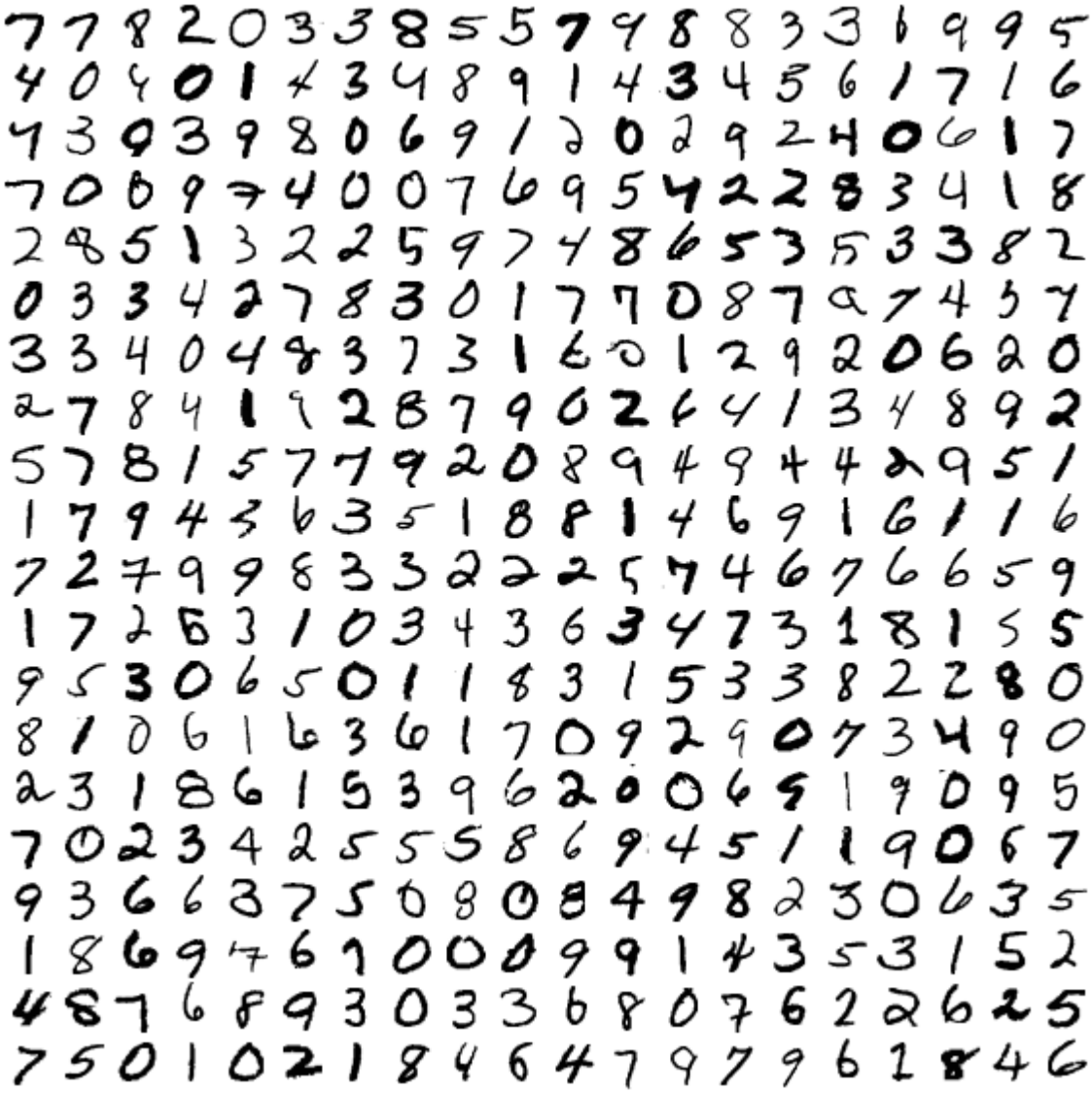
นายชญานิน ตลับเงิน 6330300160

สารบัญ

Machine learning บน MNIST dataset

ฐานข้อมูลของตัวเลขที่เขียนด้วยลายมือที่ใช้กันทั่วไปสำหรับการฝึกอบรมต่างๆ การประมวลผลภาพระบบฐานข้อมูลที่ใช้กันอย่างแพร่หลายสำหรับการฝึกอบรมและการทดสอบในด้านการเรียนรู้ของเครื่องสร้างขึ้นโดย "ผสมซ้ำ" ตัวอย่างจากชุดข้อมูลดั้งเดิมของ NIST ผู้สร้างรู้สึกว่าเนื่องจากชุดข้อมูลการฝึกอบรมของ NIST ถูกนำมาจากสำนักสำรวจสำมะโนประชากรของอเมริกาพนักงานในขณะที่ชุดข้อมูลการทดสอบถูกนำมาจากนักเรียนมัธยมปลายชาวอเมริกัน แต่ก็ไม่เหมาะสำหรับการทดลอง แมชชีนเลิร์นนิง นอกจากนี้ภาพขาวดำจาก NIST ยังถูกทำให้เป็นมาตรฐานเพื่อให้พอดีกับกรอบขอบ 28x28 พิกเซลและการต่อต้านนามแฝงซึ่งนำเสนอระดับสีเทา

ฐานข้อมูล MNIST ประกอบด้วยภาพการฝึกอบรม 60,000 ภาพและภาพทดสอบ 10,000 ภาพ ครึ่งหนึ่งของชุดการฝึกและครึ่งหนึ่งของชุดทดสอบถูกนำมาจากชุดข้อมูลการฝึกของ NIST ในขณะที่อีกครึ่งหนึ่งของชุดฝึกและอีกครึ่งหนึ่งของชุดทดสอบนั้นนำมาจากชุดข้อมูลการทดสอบของ NIST ผู้สร้างเดิมของฐานข้อมูลจะเก็บรายการวิธีการบางอย่างที่ทดสอบไว้ในกระดาษต้นฉบับพวกเขาใช้เครื่องสนับสนุนเวกเตอร์เพื่อให้ได้อัตราความผิดพลาด 0.8% ชุดข้อมูลเพิ่มเติมที่คล้ายกับ MNIST ที่เรียกว่า EMNIST ได้รับการเผยแพร่ในปี 2560 ซึ่งมีภาพการฝึกอบรม 240,000 ภาพและภาพการทดสอบตัวเลขและอักขระที่เขียนด้วยลายมือ 40,000 ภาพ



รูปที่ 1 อักขระที่เขียนด้วยลายมือ

ดึงข้อมูล MNIST โดยใช้ sklearn

เป็นตัวดึงข้อมูล คำสั่งในการดึงข้อมูลอยู่ในมอดูลย่อยชื่อว่า datasets เป็นการดึงข้อมูลที่มีอยู่แล้วมาใช้ ข้อมูลเหล่านั้นมีขนาดใหญ่เกินกว่าที่จะใส่ติดมาในตัวมอดูล จึงถูกใส่ไว้อยู่ในเว็บ จึงต้องต่อเน็ตอยู่จึงจะใช้ได้ฟังก์ชัน fetch_openml ซึ่งอยู่ภายในมอดูล datasets มีไว้ดึงข้อมูลที่ถูกเตรียมไว้ในเว็บ http://openml.org
เว็บนี้เป็นเว็บที่รวบรวมข้อมูลที่ใช้เป็นแบบฝึกสำหรับการเรียนรู้ของเครื่องมากมายหลายชนิด
สามารถเขียนได้ดังนี้

```
from sklearn import datasets
mnist = datasets.fetch_openml('mnist_784')
```

รูปที่ 2 import datasets

ในที่นี้โหลดมาแล้วข้อมูลจะถูกเก็บอยู่ในตัวแปร mnist ข้อมูลจะถูกเก็บอยู่ในแอตทริบิวต์ที่ชื่อ data โดยเป็นอาเรย์สองมิติ ส่วน ค่าตัวเลขที่เป็นคำตอบ (0~9) จะอยู่ใน target

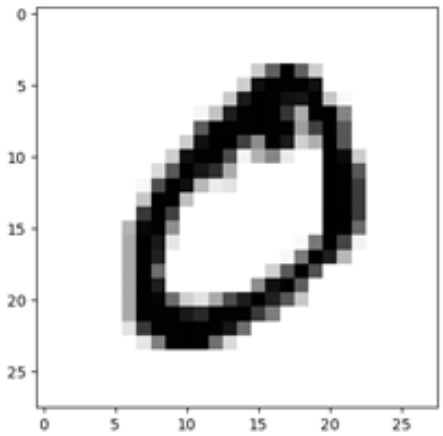
```
print(mnist.data.shape) # ได้ (70000, 784)
print(mnist.target.shape) # ได้ (70000,)
print(mnist.target) # ได้ ['5' '0' '4' ... '4' '5' '6']
```

รูปที่ 3 ขนาดของข้อมูล

ข้อมูลทั้งหมด 70000 แถว ก็คือมีภาพตัวเลขทั้งหมด 70000 ภาพ ส่วน 784 นี่คือขนาดของข้อมูล โดยข้อมูลนี้เป็นค่าความเข้มของดินสอในแต่ละช่อง ขนาดภาพ 28×28 จึงมี 784 ค่า
ลอง print(mnist.data[1]) จะได้ค่าตัวเลขตั้งแต่ 0 (บริเวณว่าง) ไปจนถึง 255

```
import matplotlib.pyplot as plt
plt.imshow(mnist.data[1].reshape(28,28),cmap='gray_r')
plt.show()
```

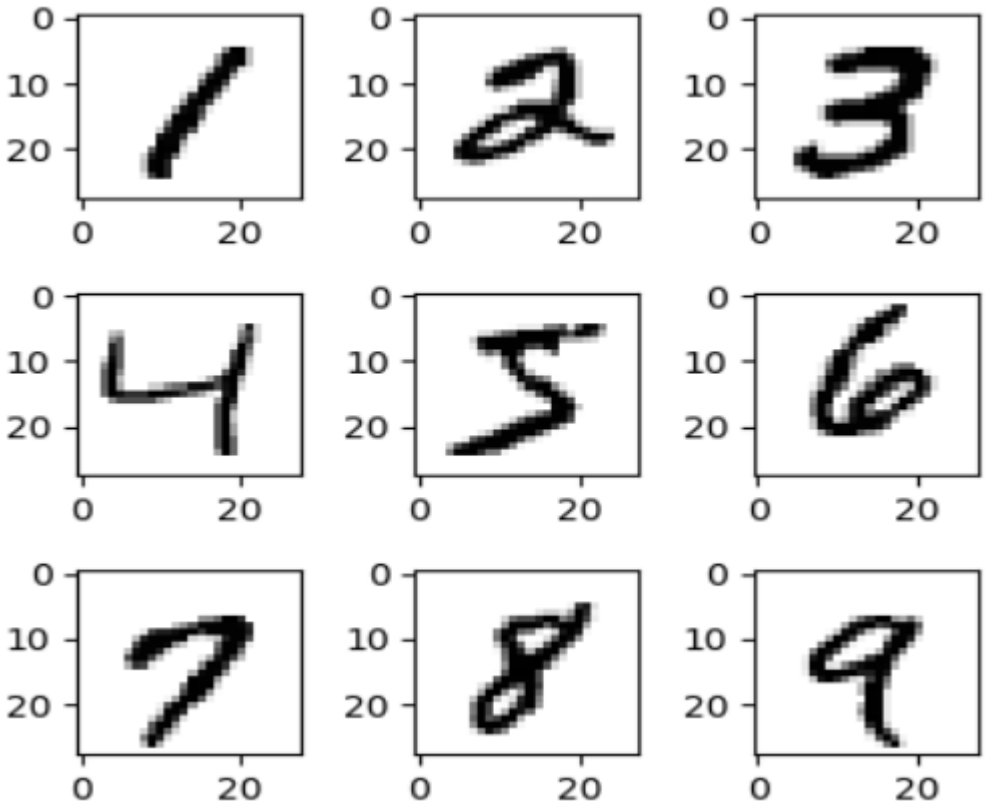
รูป 4 เอามาวาดเป็นภาพดูได้ โดยต้องทำการ reshape เป็น 28×28 ก่อน



รูป 5 จะเห็นว่าเป็นรูปเลข 0

```
target = mnist.target.astype(int)

plt.figure(figsize=[4,4],dpi=100)
for i in range(1,10):
    plt.subplot(330+i)
    ii = list(target).index(i)
    plt.imshow(mnist.data[ii].reshape(28,28),cmap='gray_r')
plt.tight_layout()
plt.show()
```

รูป 6 ลองดูตัวเลขอื่นๆ



รูป 7 ตัวเลข 1 ถึง 9

ตัวเลขพวกนี้คือสิ่งที่จะป้อนให้โปรแกรมของเราเรียนรู้ แบบนี้โปรแกรมเราก็จะเหมือนกับเด็กน้อยไร้เดียงสาที่พยายามหัดจดจำ หัดแยกแยะว่าภาพนี้คือตัวเลขนี้ ภาพนั้นคือตัวเลขนั้น

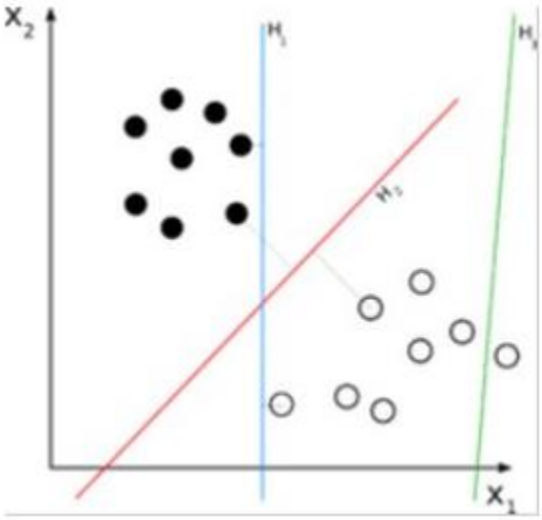
สรุปหลักการของอัลกอริทึม k-Means และ SVM

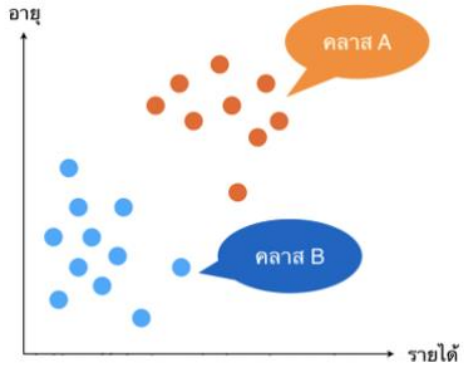
Support Vector Machine

เป็นอัลกอริทึมที่สามารถนำมาช่วยแก้ปัญหาการจำแนกข้อมูล ใช้ในการวิเคราะห์ข้อมูลและจำแนกข้อมูล โดยอาศัยหลักการของการหาสัมประสิทธิ์ของสมการเพื่อสร้างเส้นแบ่งแยกกลุ่มข้อมูลที่ถูกป้อนเข้าสู่กระบวนการสอนให้ระบบเรียนรู้ โดยเน้นไปยังเส้นแบ่งแยกแยะกลุ่มข้อมูลได้ดีที่สุด

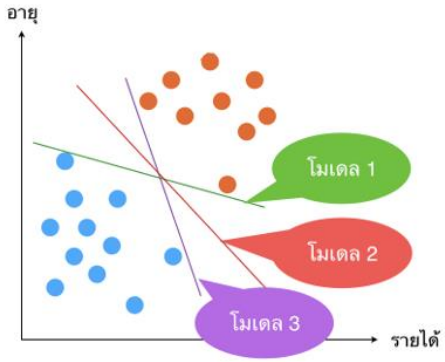
แนวความคิดของ Support Vector Machine

เกิดจากการที่นำค่าของกลุ่มข้อมูลมาวางลงในฟีเจอร์สเปซ (Feature Space) จากนั้นจึงหาเส้นที่ใช้แบ่งข้อมูลทั้งสองออกจากกันโดยจะสร้างเส้นแบ่ง (Hyperplane) ที่เป็นเส้นตรงขึ้นมา และเพื่อให้ทราบว่าเส้นตรงที่แบ่งสองกลุ่มออกจากกันนั้น เส้นตรงใดเป็นเส้นที่ดีที่สุด





รูปที่ 1



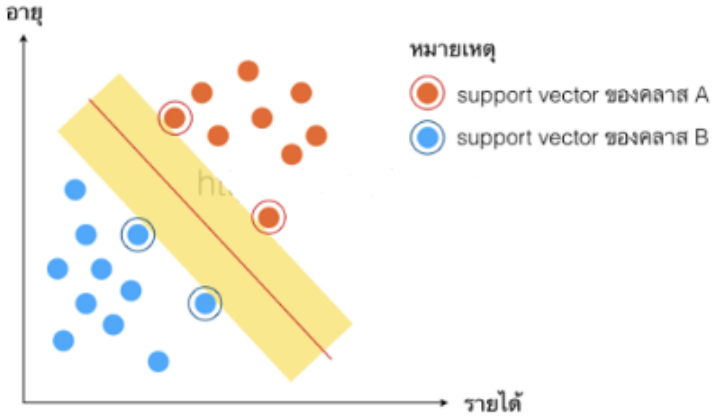
รูปที่ 2

สมมติว่าเราต้องการคัดแยกข้อมูลออกเป็น 2 กลุ่ม โดยใช้เส้นแบ่งที่เป็นเส้นตรง จะเห็นว่ามีเส้นตรงจำนวนมากที่สามารถคัดแยกได้ แต่เส้นตรงเส้นไหนที่ดีที่สุด เราจะนิยาม Margin เป็นผลรวมระยะห่างของเส้นตรงที่เป็นเส้นแบ่ง ถึงเส้นตรงที่ผ่านข้อมูลที่ใกล้ที่สุดและขนานกับเส้นแบ่งของทั้งสองกลุ่ม จะเห็นว่า H1 แม้จะสามารถแบ่งข้อมูลทั้งสองกลุ่มออกได้เช่นกัน แต่ ระยะในการแบ่งจากเส้นแบ่งไปถึงข้อมูลที่ใกล้ที่สุดนั้นมีขนาดน้อย แต่จากเส้น H2 จะเป็นเส้นที่แบ่งกลุ่มที่กว้างมากที่สุดของทั้งสองกลุ่มคือให้ค่า maximum margin เราเรียกข้อมูลที่อยู่บน margin นี้ว่า Support Vector

จากการกระจายตัวของข้อมูลในรูปที่ 1 จะเห็นว่าสามารถแบ่งแยกออกเป็น 2 กลุ่มได้อย่างชัดเจน ซึ่งโดยปกติแล้วเราจะใช้ linear model (หรือสมการเส้นตรง) เพื่อทำการแบ่งข้อมูลออกเป็น 2 คลาส ทว่า linear model นี้สามารถเป็นไปได้หลากหลายเส้นดังในรูปที่ 2

การเลือกโมเดล

เราควรจะเลือกโมเดลที่ไม่ overfitting หรือโมเดลที่ไม่จำรูปแบบของข้อมูล training มากเกินไป จากตัวอย่างจะเห็นว่าโมเดลที่ 1 และโมเดลที่ 3 จะมีจุดหนึ่งที่ linear model อยู่ใกล้กับข้อมูลแต่ละคลาสมากเกินไป นั่นคือ ถ้ามีข้อมูลใหม่ที่อยู่ห่างออกไปสักเล็กน้อยก็จะทำนายผิดไปพลาดไป ดังนั้นในตัวอย่างนี้จึงควรจะเลือกโมเดลที่ 2 (เส้นสีแดง) และนั่นคือหลักการของ SVM ที่เลือก linear model ที่มีระยะห่างระหว่าง 2 คลาสห่างกันมากที่สุด ดังแสดงในรูป



รูปที่ 3

Overfitting คือ

การที่โมเดลจดจำรูปแบบของข้อมูล training มากเกินไปจนไม่สามารถทำนายข้อมูล unseen data ได้

ประโยชน์

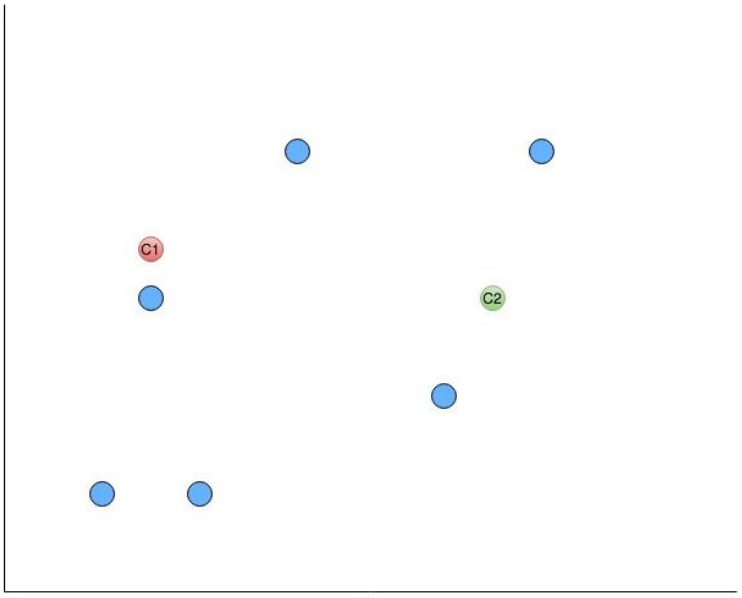
การวิเคราะห์จำแนกกลุ่ม ด้วยการวิเคราะห์จากตัวแปรตาม 1 ตัวและตัวแปรอิสระตั้งแต่ 1 ตัวขึ้นไป การวิเคราะห์ด้วยวิธีนี้นอกจากจะสามารถจำแนกความ แตกต่างระหว่างกลุ่มได้และยังสามารถบอกได้ว่าตัวแปรใดจำแนกได้ดีมากน้อยกว่ากัน นั่นคือสามารถ บอกประสิทธิภาพหรือน้ำหนักในการจำแนกของการจัดเข้ากลุ่ม นอกจากนี้การวิเคราะห์จำแนกกลุ่มยัง สามารถพยากรณ์การเข้าสู่กลุ่มของข้อมูลใหม่ด้วย ดังนั้นการวิเคราะห์จำแนกกลุ่มจึงเป็นเทคนิคการวิเคราะห์ความสัมพันธ์หรือการหาสาเหตุได้

K means

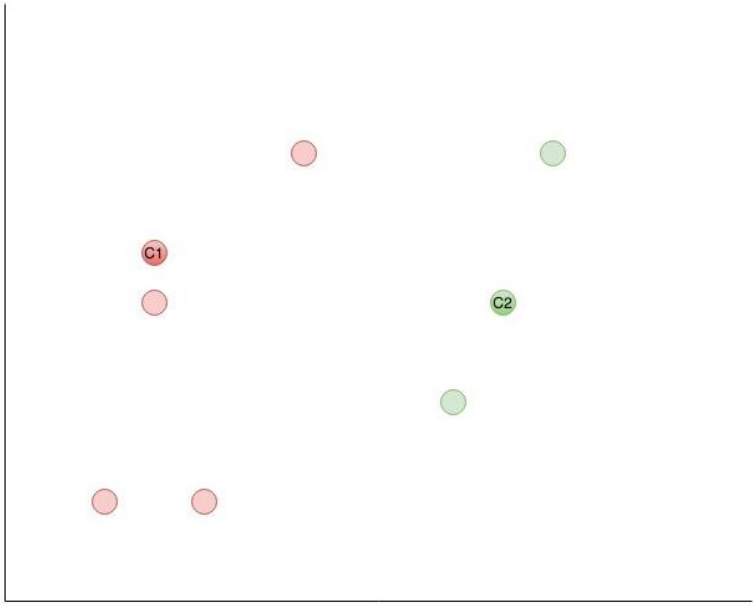
วิธีการหนึ่งใน Data mining อยู่ในกลุ่มของ Unsupervised Learning หรือแปลตรงๆคือการเรียนรู้แบบไม่ต้องสอน (Supervised Learning ต้องสอนก่อนต้องจับ Train และต้อง Test เป็นต้น) โดยหน้าที่หลักของ K-means คือการแบ่งกลุ่ม แบบ Clustering ซึ่งการแบ่งกลุ่มในลักษณะนี้จะใช้พื้นฐานทางสถิติ ซึ่งแน่นอนว่าต้องมีตัวเลขประกอบ อย่างน้อย 2 ตัวแปรขึ้นไป

วิธีการของ K-means มี 4 ขั้นตอนดังนี้

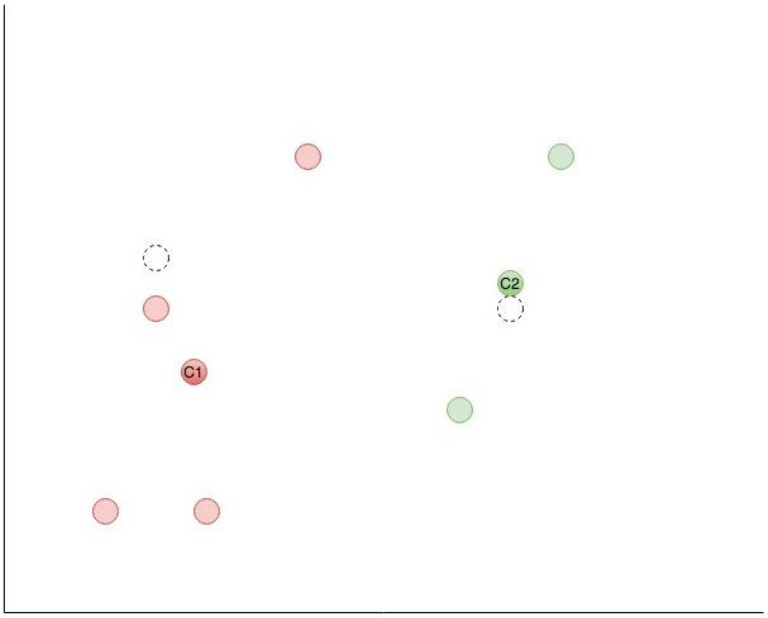
1. กำหนดจำนวนกลุ่มขึ้นมาก่อนเช่น 2 กลุ่มหรือหมายความว่าค่า K=2 (กำหนดเป็น C1 และ C2) และสุ่มตำแหน่งแกน x, y ให้กับ C1 และ C2 จะได้ C1(x1,y1) และ C2(x2,y2)
2. ดูตำแหน่งของสมาชิกแต่ละสมาชิกว่าอยู่ใกล้ใครมากกว่ากันก็ให้คนนั้นเป็นสมาชิกของ C นั้น จาก ตรงนี้เราจะรู้แล้วว่าสมาชิกแต่ละคนอยู่ในกลุ่มใดระหว่าง C1 และ C2
3. ปรับ x, y ของ C1 และ C2 ใหม่ให้อยู่ตรงกลางของกลุ่ม
4. ทำ ตามข้อ 2 และข้อ 3 อีกครั้งจนกว่า C1 และ C2 ตำแหน่งไม่เปลี่ยน



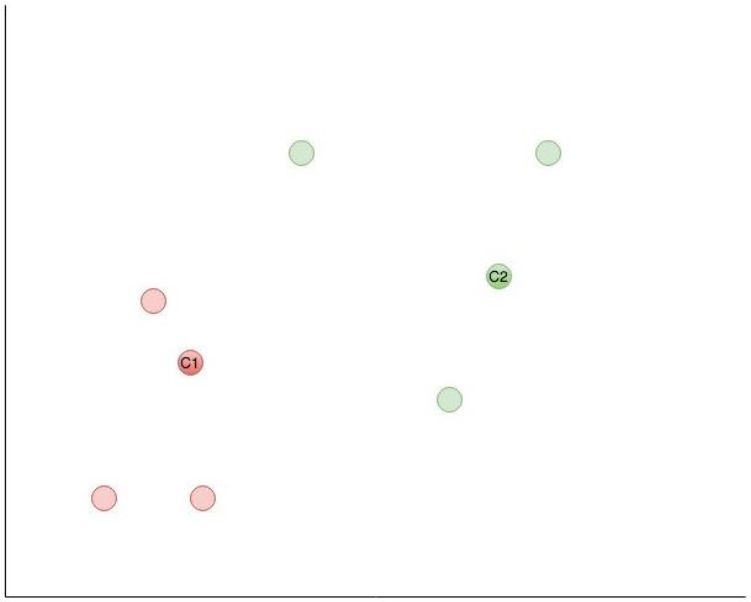
รูปที่ 1 หาสมาชิกที่อยู่ใกล้กลุ่มที่สุด



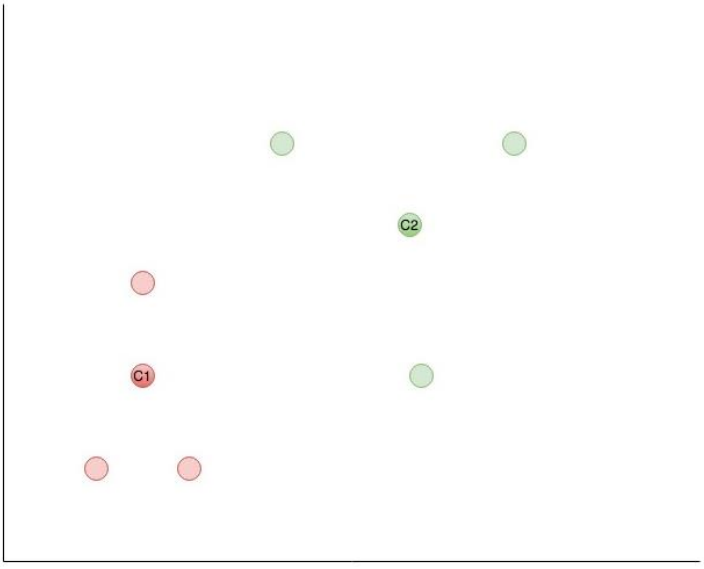
รูปที่ 2 หาสมาชิกที่อยู่ใกล้กลุ่มที่สุด



รูปที่ 3 C1 และ C2 ย้ายไปตรงกลาง



รูปที่ 4 หาสมาชิกที่อยู่ใกล้กลุ่มที่สุด



รูปที่ 5 C1 และ C2 ย้ายไปตรงกลาง

Logistic Regression Analysis

การวิเคราะห์การทดถอยทางโลจีสติก เป็นเทคนิคการวิเคราะห์ตัวแปรเชิงพหุที่มีวัตถุประสงค์ เพื่อประมาณค่าหรือทำนายเหตุการณ์ที่สนใจว่าจะเกิดขึ้นหรือไม่เหตุการณ์นั้นภายใต้อิทธิตัวปัจจัย แบบจำลองโลจีสติกประกอบด้วยตัวแปรตามหรือตัวแปรเกณฑ์ ที่ต้องเป็นตัวแปรแบบทวินาม กล่าวคือ มีได้ 2 ค่า เช่น “เกิด” หรือ “ไม่เกิด” หรือ “เสี่ยง” หรือ “ไม่เสี่ยง” เป็นต้น และตัวแปรอิสระ ที่อาจจะมีตัวเดียวหรือหลายตัวที่เป็นได้ทั้งตัวแปรเชิงกลุ่ม หรือ ตัวแปรแบบต่อเนื่อง การวิเคราะห์แบบถดถอยโลจีสติก เกี่ยวข้องกับทฤษฎีความน่าจะเป็นทวินาม ถูกเรียกว่า Binomial Logistic Regression ถ้าตัวแปรตามเป็นพหุนามจะเรียกว่า Multinomial Logistic Regression การถดถอยโลจีสติก จัดเป็นเครื่องมือวิเคราะห์ข้อมูลในการศึกษาวิจัยที่มีวัตถุประสงค์เพื่อทำนายเหตุการ์ณ หรือ ประเมินความเสี่ยง

การจำลองการถดถอยโลจีสติก

การวิเคราะห์การทดถอยทางโลจีสติกการประเมินค่าความน่าจะเป็นของการเกิดเหตุการณ์ มีตัวแบบมาจากฟังก์ชั่นโลจีสติก หากมีตัวแปรอิสระเพียงตัวเดียวฟังก์ชั่นโลจีสติกจะแสดงความน่าจะเป็นของเหตุการณ์ดังรูปต่อไปนี้

$$Prob\ (event) = \frac{e^{(\beta_0+\beta_1 X)}}{1+e^{(\beta_0+\beta_1 X)}}$$

หรือ

$$Prob\ (event) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

เมื่อ

β_0 คือ ค่าคงที่ (เมื่อไม่มีอิทธิพลจากตัวแปรอิสระใด)

β_i คือ ค่าสัมประสิทธิ์ของตัวแปรอิสระ (ประมาณได้จากข้อมูลสังเกต)

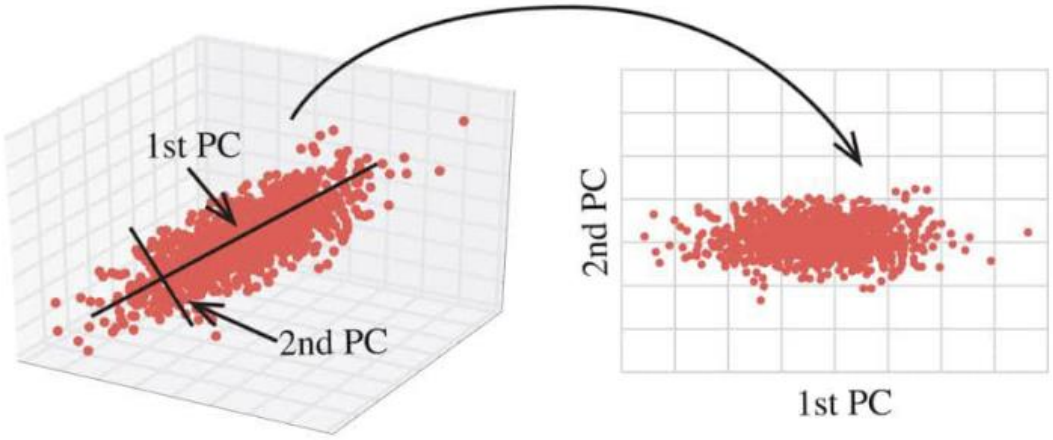
X คือ ตัวแปรอิสระ (ตัวแปรทำนาย)

e คือ ลอการิทึมธรรมชาติ (มีค่าประมาณ 2.71828....)

Principal Components Analysis (PCA)

คือ การวิเคราะห์องค์ประกอบหลัก การที่จะวิเคราะห์ประกอบหลักได้นั้น องค์ประกอบต้องมีหลายองค์ประกอบรวมกัน อาจจะมีมากกว่าหนึ่ง หรือ มากกว่าหนึ่งร้อย แต่สิ่งที่ต้องการคือองค์ประกอบหลักเท่านั้น ถ้ามองในเชิงสถิติ เพื่อให้เห็นภาพชัดขึ้น

สมมุติมี Feature อยู่ 50 ตัว การสร้างโมเดลจากข้อมูล Feature ทั้งหมดนั้นทำได้ยาก เพราะโอกาสที่ Target variable จะมีความสัมพันธ์กับ 50 Feature นั้นเป็นไปแทบไม่ได้เลย ดังนั้นการทำ PCA จึงเป็นความสัมพันธ์ของข้อมูลภายใน Feature ด้วยกันเอง และให้ความสำคัญกับ Feature ที่มีความสัมพันธ์มากกว่าก่อน PCA จึงเป็นเทคนิคในการลดมิติข้อมูลที่มีอยู่มาก ในเหลือน้อยลงในการคำนวณ



รูปที่ 1 การลดมิติของข้อมูล

การใช้งาน PCA

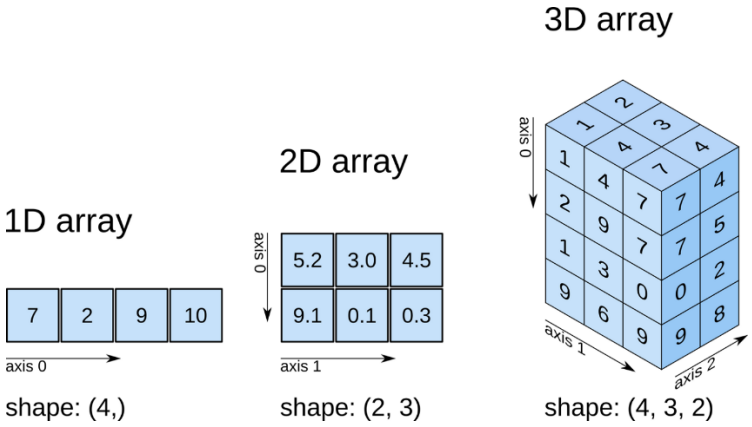
- เหตุผลหลักของ PCA คือการลดมิติของข้อมูล (High dimensional) ให้มีมิติลดลง เห็นได้บ่อยครั้งในการใช้ PCA สำหรับงาน Image
- เพื่อลดจำนวนตัวแปร (Features) ทั้งนี้ PCA ไม่ได้ทำการลบข้อมูลแต่อย่างใด แต่เป็นการ Transform ข้อมูลในอยู่ในรูปแบบอื่น และการอธิบายแบบอื่น ซึ่งมีประโยชน์มากในงาน Regression เป็นต้น

- การทำ PCA ถือเป็นการแก้ไขปัญหา Multicollinearity ไปโดยอัตโนมัติ เพราะ Process ในการสร้าง Components ของ PCA จะไม่มีความสัมพันธ์กันเกิดขึ้น

ภาษาไพธอนสำหรับวิทยาศาสตร์ข้อมูล

Numpy

Numpy คือ Library ที่ใช้สำหรับการจัดการและสร้างโตนงสร้างข้อมูลประเภท array เพื่อจัดเก็บและเตรียมไปใช้ต่อมีความสามารถในการคำนวณทางคณิตศาสตร์พื้นฐานบางอย่างที่ List ไม่สามารถทำได้และกิน Memory น้อยกว่า ซึ่งเป็น Library พื้นฐาน สำหรับการพัฒนาโปรเจค Data Sci



รูปที่ 1

ความแตกต่างระหว่าง Numpy กับ Python Lists

- NumPy Arrays สามารถคำนวณและดำเนินการทางตรรกะใน Matrix , Array หลายมิติ และ Array ได้อย่างรวดเร็ว มากกว่า Python Lists
- ในการใช้งาน NumPy Arrays จะประหยัด Memory ได้มากกว่าใช้ Python Lists
- NumPy Arrays มีขนาดคงที่เมื่อสร้าง ซึ่งแตกต่างจาก Python Lists (ซึ่งสามารถขยายได้แบบไดนามิก) การเปลี่ยนขนาดของ ndarray จะสร้างอาร์เรย์ใหม่และลบต้นฉบับ
- การเปลี่ยนแปลงข้อมูลใน NumPy Arrays ก็สามารถทำได้เร็วกว่า Python Lists
- NumPy Arrays สามารถเข้าถึงข้อมูลภายในได้เร็วกว่า Python Lists

ความแตกต่างระหว่าง Numpy กับ Pandas

- การเข้าถึงข้อมูลในแต่ละ index ของ Numpy เร็วกว่า Pandas อย่างมาก
- การทำงานของ Numpy มีเวลาการทำงานอยู่ในหน่วย **nanosecond**
- การทำงานของ Pandas มีเวลาการทำงานอยู่ในหน่วย **millisecond**

ชนิดข้อมูลของนัมไพ

Array numpy ต้องเก็บข้อมูลชนิดใดชนิดหนึ่ง ตัวอย่างชนิดข้อมูลดังนี้

ชนิดข้อมูล	รหัสแทน	คำอธิบาย
strings	s หรือ Sn หรือ |Sn	ข้อมูลที่เป็นข้อความ จะเขียนภายใต้เครื่องหมาย ' ' เช่น 'NumPy' โดยที่ n คือความยาวสูงสุดของสมาชิกในอาร์เรย์ เช่น S10 หมายถึง มีความยาวสูงสุดไม่เกิน 10 อักขระ
integer	i	ตัวเลขจำนวนเต็ม เช่น -1, -2, 3, 4
float	f	เลขทศนิยม เช่น 1.2, 29.95
boolean	b	ข้อมูลแบบค่าจริง (True) หรือ เท็จ (False)
complex	c	จำนวนเชิงซ้อน เช่น 1.0 + 2.0j, 1.5 + 2.5j
unicode	U หรือ <Un	อักขระ Unicode โดย n คือความยาวสูงสุดของอักขระ เช่น <U8
object		เป็นข้อมูลชนิดใดก็ได้

แอททริบิวต์ของ Numpy

ใช้บอกโครงสร้างของ array คือ

แอททริบิวต์	คำอธิบาย
ndim	แสดงขนาดมิติของอาร์เรย์
dtype	แสดงชนิดข้อมูลของอาร์เรย์
shape	รูปร่างของข้อมูล
size	จำนวนสมาชิกทั้งหมดที่จัดเก็บในอาร์เรย์
itemsize	ขนาดของอาร์เรย์สำหรับสมาชิก 1 ตัว ซึ่งมีหน่วยเป็นไบต์ (Byte)

สร้างนัมไพอาร์เรย์แบบใช้ข้อมูลค่าเดียวกัน

การสร้างอาร์เรย์ที่มีสมาชิกมี ค่าเหมือนกันทั้งหมด โดยมีฟังก์ชันสำหรับการสร้างอาร์เรย์ ดังนี้

ฟังก์ชัน	รูปแบบ	คำอธิบาย
zeros()	np.zeros(shape, dtype)	สมาชิกทุกตัวมีค่าเป็น 0 ทั้งหมด
ones()	np.ones(shape, dtype)	สมาชิกทุกตัวมีค่าเป็น 1 ทั้งหมด
full()	np.full(shape, value, dtype)	สมาชิกทุกตัวมีค่าตามที่ระบุ

สร้างนัมไพอาร์เรย์แบบกำหนดช่วงข้อมูล

การสร้างอาร์เรย์ที่มีสมาชิกอยู่ในช่วงตัวเลขที่กำหนด โดยมีฟังก์ชันสำหรับการสร้างอาร์เรย์ ดังนี้

ฟังก์ชัน	รูปแบบ	คำอธิบาย
arange()	np.arange(*[start,]* stop, *[step]*)	กำหนดช่วงข้อมูลตั้งแต่ค่าเริ่มต้น จนถึงค่าสิ้นสุด -1 หากไม่ระบุค่าเริ่มต้น ค่า default คือ 0 [0, stop) หากไม่ระบุค่าที่เพิ่มขึ้น ค่า default คือ เพิ่มขึ้นทีละ 1
linspace()	np.linspace(start, stop, *[num]*)	กำหนดช่วงข้อมูลตั้งแต่ค่าเริ่มต้น จนถึงค่าสิ้นสุด โดยมีระยะห่างเท่าๆ กัน ตามจำนวนสมาชิกที่กำหนด หากไม่ระบุจำนวน ค่า default คือ 50 จำนวน

สร้างนัมไพอาร์เรย์แบบสุ่ม

การสร้างอาร์เรย์ที่มีสุ่มค่าตัวเลข ด้วยเมธอด random โดยมีฟังก์ชันสำหรับการสร้างอาร์เรย์ ดังนี

ฟังก์ชัน	รูปแบบ	คำอธิบาย
randint()	np.random.randint(low, high, size)	สุ่มเลขจำนวนเต็มตามจำนวนที่กำหนด ตั้งแต่ 0 ถึง ค่าต่ำสุด - 1 กรณีไม่กำหนดค่าสูงสุด [0, low) หรือ ตั้งแต่ค่าต่ำสุด ถึง ค่าสูงสุด -1 [low,high)
uniform	np.random.uniform(low, high, size)	สุ่มเลขทศนิยมตามจำนวนที่กำหนด ตั้งแต่ต่ำสุด ถึงค่าสูงสุด - 1 [low,high)
rand()	np.random.rand(size)	สุ่มเลขทศนิยมระหว่าง 0.0 ถึง 1.0 ตามจำนวนที่กำหนด
choice	np.random.choice(sequence, size)	สุ่มค่าจากรายการ (sequence) ตามจำนวนที่กำหนด

ฟังก์ชันทางสถิติ

ไลบรารี Numpy มีฟังก์ชันสำหรับดำเนินการต่าง ๆ เป็นจำนวนมาก โดยตัวอย่างฟังก์ชันทางสถิติมีดังนี้

ฟังก์ชัน	รูปแบบ	คำอธิบาย
sum()	ndarray.sum()	หาผลรวมของสมาชิกอาร์เรย์
cumsum()	ndarray.cumsum()	หาผลบวกสะสมของสมาชิก
mean()	ndarray.mean()	หาค่าเฉลี่ยของสมาชิก
min()	ndarray.min()	หาค่าต่ำสุดของสมาชิก
max()	ndarray.max()	หาค่ามากสุดของสมาชิก
unique()	ndarray.unique()	แสดงค่าสมาชิกที่มีค่าไม่ซ้ำกัน (ไม่ต้องระบุแกน)

การคัดลอกสมาชิก

ฟังก์ชัน	รูปแบบ	คำอธิบาย
copy()	ndarray.copy()	คัดลอกสมาชิก หากมีการเปลี่ยนแปลงสมาชิกที่อาร์เรย์หนึ่ง จะไม่ส่งผลให้สมาชิกอีกอาร์เรย์หนึ่งเปลี่ยนแปลงตามไปด้วย
view()	ndarray.view()	คัดลอกสมาชิกแบบอ้างอิงสมาชิก หากมีการเปลี่ยนแปลงสมาชิกที่อาร์เรย์หนึ่ง จะส่งผลให้สมาชิกอีกอาร์เรย์หนึ่งเปลี่ยนแปลงตามไปด้วย

การเพิ่มสมาชิก

ฟังก์ชัน	รูปแบบ	คำอธิบาย
append()	np.append(ndarray, values, axis)	เพิ่มข้อมูลต่อท้ายอีกอาร์เรย์หนึ่ง
insert()	np.insert(ndarray, index, values, axis)	แทรกข้อมูลเข้าไปในอาร์เรย์ในตำแหน่งที่กำหนด

การลบสมาชิก

ฟังก์ชัน	รูปแบบ	คำอธิบาย
delete()	np.delete(ndarray, index, axis)	ลบข้อมูลในตำแหน่งที่กำหนดออกจากอาร์เรย์

การเปลี่ยนขนาดอาร์เรย์

ฟังก์ชัน	รูปแบบ	คำอธิบาย
resize()	np.resize((row,column))	เปลี่ยนขนาดของอาร์เรย์แบบไม่อ้างอิง หากมีการเปลี่ยนแปลงค่าของสมาชิกที่อาร์เรย์หนึ่ง จะไม่ส่งผลต่ออีกอาร์เรย์หนึ่ง
reshape()	np.reshape((row,column))	เปลี่ยนขนาดของอาร์เรย์แบบอ้างอิง หากมีการเปลี่ยนแปลงค่าของสมาชิกที่อาร์เรย์หนึ่ง จะส่งผลต่ออีกอาร์เรย์หนึ่ง

ฟังก์ชัน	รูปแบบ	คำอธิบาย
flatten()	ndarray.flatten()	เปลี่ยนขนาดอาร์เรย์ให้เป็นแบบ 1 มิติ โดยข้อมูลไม่อ้างอิงถึงกัน
ravel()	ndarray.ravel()	เปลี่ยนขนาดอาร์เรย์ให้เป็นแบบ 1 มิติ โดยข้อมูลอ้างอิงถึงกัน

ฟังก์ชัน	รูปแบบ	คำอธิบาย
transpose()	ndarray.transpose()	เปลี่ยนแกนของอาร์เรย์ จากเดิม แนวตั้ง เปลี่ยนให้เป็น แนวนอน จากเดิม แนวนอน เปลี่ยนให้เป็น แนวตั้ง

การดำเนินการอื่น ๆ

ฟังก์ชัน	รูปแบบ	คำอธิบาย
sort()	np.sort(ndarray, axis)	เรียงลำดับค่าของสมาชิก
unique()	np.unique()	แสดงค่าสมาชิกที่มีค่าไม่ซ้ำกัน (ไม่ต้องระบุแกน)

ตัวอย่างโปรแกรมที่ใช้ numpy

[141]:
```
import numpy as np
```

[142]:
```
height = [150, 178, 167, 189, 157 , 176]
```

[143]:
```
weight = [40, 55, 67 , 52, 77, 110]
```

[144]:
```
H = np.array(height)
```

[145]:
```
W = np.array(weight)
```

[146]:
```
H = H/100
```

[147]:
```
BMI = W/(H*H)
```

[148]:
```
print(BMI)
```

```
[17.77777778 17.35891933 24.02380867 14.55726323 31.2385898  35.51136364]
```

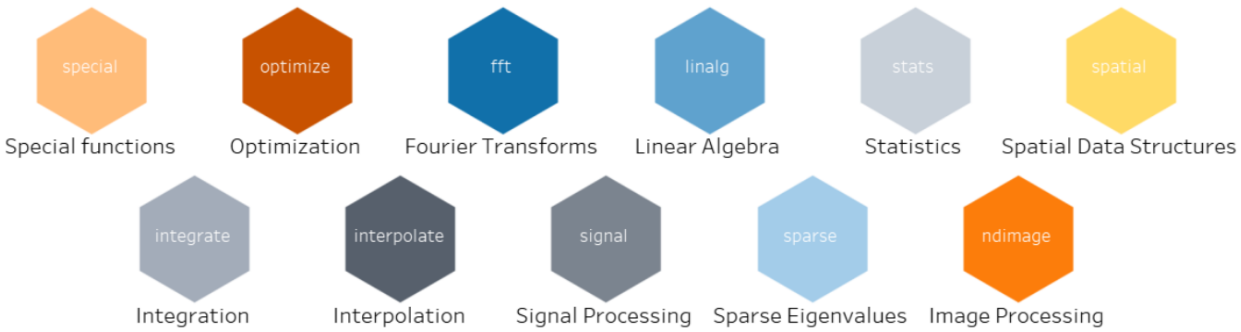
Scipy

SciPy เป็นชุดของอัลกอริทึมทางคณิตศาสตร์และฟังก์ชันความสะดวกสบายที่พัฒนาขึ้นจาก NumPy ของ Python เพิ่มประสิทธิภาพการทำงานให้กับ Python แบบโต้ตอบโดยให้ผู้ใช้มีคำสั่งและคลาสระดับสูงสำหรับการจัดการและแสดงภาพข้อมูล ด้วย SciPy เซสชัน Python แบบโต้ตอบจะกลายเป็นระบบการประมวลผลข้อมูลและการสร้างต้นแบบระบบเช่น MATLAB, IDL, Octave, R-Lab และ SciLab

ความแตกต่างระหว่าง Numpy กับ Scipy

- NumPy Arrays สามารถคำนวณและดำเนินการทางตรรกะใน Matrix , Array หลายมิติ และ Array ได้อย่างรวดเร็ว มากกว่า Python Lists
- Numpy มีการดำเนินการทางคณิตศาสตร์ขั้นพื้นฐานเท่านั้นและการดำเนินการขั้นสูงแค่บางอย่าง
- Scipy มีการดำเนินการทางสถิติที่ซับซ้อนทั้งหมดที่จำเป็นสำหรับการวิเคราะห์ข้อมูลที่เหมาะสม

ฟังชันก์ต่างในการทำงานของ Scipy มีดังนี้



special ฟังก์ชันพิเศษใน scipy **ใช้เพื่อดำเนินการทางคณิตศาสตร์กับข้อมูลที่กำหนด** ฟังก์ชันพิเศษใน scipy เป็นโมดูลที่มีอยู่ในแพ็คเกจ scipy ภายในฟังก์ชันพิเศษนี้วิธีการที่มีอยู่คือ cbrt - ซึ่งให้รากลูกบาศก์ของจำนวนที่กำหนด

stats ประกอบด้วยการแจกแจงความน่าจะเป็น, สถิติสรุปและความถี่, ฟังก์ชันสหสัมพันธ์และการทดสอบทางสถิติ, สถิติที่สวมหน้ากาก, การประมาณค่าความหนาแน่นของเคอร์เนล, ฟังก์ชันกึ่งมอนติคาร์โล และอื่น ๆ

สถิติเป็นพื้นที่ที่มีขนาดใหญ่มากและมีหัวข้อที่อยู่นอกขอบเขตสำหรับ SciPy และครอบคลุมโดยแพ็คเกจอื่น ๆ คือ

- statsmodels: การถดถอย, แบบจำลองเชิงเส้น, การวิเคราะห์อนุกรมเวลา, ส่วนขยายไปยังหัวข้อที่ครอบคลุมโดย .scipy.stats
- Pandas: ข้อมูลตาราง, ฟังก์ชันอนุกรมเวลา, อินเทอร์เฟซไปยังภาษาสถิติอื่น ๆ
- PyMC: การสร้างแบบจำลองทางสถิติแบบ Bayesian, การเรียนรู้ของเครื่องความน่าจะเป็น
- scikit-learn: การจำแนกประเภทการถดถอยการเลือกแบบจำลอง
- Seaborn: การสร้างภาพข้อมูลทางสถิติ
- rpy2: งูหลามถึงสะพาน R

spatial สามารถคำนวณสามเหลี่ยมไดอะแกรม Voronoi โดยใช้ประโยชน์จากไลบรารี Qhull นอกจากนี้ยังมีการใช้งาน **KDTree** สำหรับการสืบค้นจุดเพื่อนบ้านที่ใกล้ที่สุดและยูทิลิตี้สำหรับการคำนวณระยะทางในเมตริกต่าง ๆ

sparse เป็นเครื่องมือสำหรับการสร้างเมทริกซ์ที่ละเอียดโดยใช้โครงสร้างข้อมูลหลายแบบ รวมถึงเครื่องมือสำหรับการแปลงเมทริกซ์ที่หนาแน่นเป็นเมทริกซ์ที่ละเอียด ฟังก์ชันพีชคณิตเชิงเส้น NumPy และ SciPy จำนวนมากที่ทำงานบนอาร์เรย์ NumPy สามารถทำงานบนอาร์เรย์เบาบางของ SciPy ได้อย่างโปร่งใส

signal คำนวณอาร์เรย์ดัชนีความล่าช้า / การกระจัดสำหรับสหสัมพันธ์ข้าม 1 มิติ

optimize ฟังก์ชันสำหรับการลด (หรือเพิ่ม) ฟังก์ชันวัตถุประสงค์ให้เหลือน้อยที่สุด ซึ่งอาจอยู่ภายใต้ข้อจำกัด มันรวมถึงตัวแก้ปัญหาสำหรับปัญหาที่ไม่เชิงเส้น การเขียนโปรแกรมเชิงเส้นสี่เหลี่ยมจัตุรัสที่ จำกัด และไม่เชิงเส้นการค้นหารากและการปรับเส้นโค้งให้เหมาะสม

odr การถดถอยระยะทางมุมฉากซึ่งใช้ในการศึกษาการถดถอย การถดถอยเชิงเส้นพื้นฐานมักใช้เพื่อประเมินความสัมพันธ์ระหว่างตัวแปรสองตัว y และ x โดยการวาดเส้นที่เหมาะสมที่สุดบนกราฟ

ndimage หมายถึงภาพ n มิติ งานที่พบบ่อยที่สุดในการประมวลผลภาพมีดังนี้ อินพุต/ เอาต์พุตแสดงภาพ การปรับแต่งพื้นฐาน - การครอบตัดการพลิกการหมุน ฯลฯ

linalg คำนวณ eigenvalues จากปัญหา eigenvalue ทั่วไปหรือทั่วไป ฟังก์ชันนี้ส่งกลับค่า Eigen และเวกเตอร์ Eigen

io ฟังก์ชั่นยูทิลิตี้เพิ่มเติมสำหรับการเพิ่มประสิทธิภาพสถิติและการประมวลผลสัญญาณ

interpolate สร้างฟังก์ชันตามจุดข้อมูลคงที่ซึ่งสามารถประเมินได้ทุกที่ภายในโดเมนที่กำหนดโดยข้อมูลที่กำหนดโดยใช้การแก้ไขเชิงเส้น ด้วยการใช้ข้อมูลข้างต้นให้เราสร้างฟังก์ชันการแก้ไขและวาดกราฟที่แก้ไขใหม่

integrate การรวมเชิงตัวเลขคือการคำนวณโดยประมาณของอินทิกรัลโดยใช้เทคนิคเชิงตัวเลข

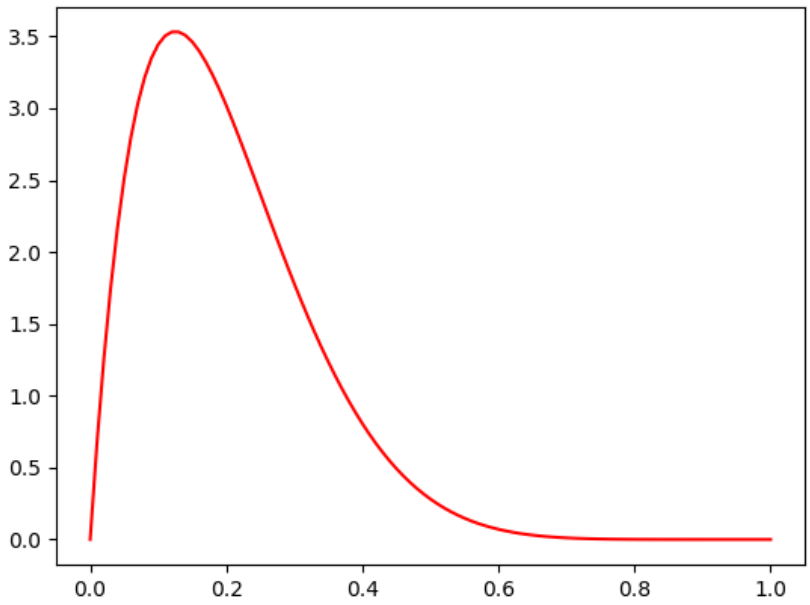
fftpack ช่วยให้การคำนวณการแปลงฟูริเยร์เป็นไปอย่างรวดเร็ว เป็นภาพประกอบสัญญาณอินพุต (มีเสียงดัง) อาจมีลักษณะดังนี้ - นำเข้า numpy เป็น np time_step = 0.02 ช่วงเวลา = 5

constants เป็นโมดูลย่อยภายในไลบรารี Scipy ที่ทำสิ่งนี้ให้เรา มันมีรายการที่สมบูรณ์ของค่าคงที่ทางคณิตศาสตร์สากลค่าคงที่ทางกายภาพและหน่วย

cluster โมดูลลำดับชั้นมีฟังก์ชันสำหรับการจัดกลุ่มแบบลำดับชั้นและการสลับกัน คุณสมบัติของมันรวมถึงการสร้างคลัสเตอร์ลำดับชั้นจากระยะไกล

ตัวอย่างโปรแกรมที่ใช้ scipy

[17]:
```
import scipy.stats
import numpy as np
import matplotlib.pyplot as plt
a = 2
b = 8
beta = scipy.stats.beta(a,b)
x = np.linspace(0,1,101)
y = beta.pdf(x)
plt.plot(x,y,'r')
plt.show()
```



Matplotlib

เป็น library ที่มีคำสั่งใช้สร้างกราฟต่างๆ หลายรูปแบบที่สามารถปรับเปลี่ยนการแสดงผล สี เนื้อหารายละเอียดต่างๆ ได้ครบถ้วน และเป็นเครื่องมือพื้นฐานที่ใช้กันในการแสดงผลข้อมูล

การสร้างกราฟเส้น

ใช้คำสั่ง plt.plot() ที่รับค่า x และค่า y โดย ในตัวอย่างใส่เป็น array matplotlib จะนำข้อมูลที่ตำแหน่งตรงกันมาประกอบเป็นจุดพิกัดและนำไป plot เช่นจุดแรกจะเกิดจากข้อมูลตัวแรกจาก x และข้อมูลตัวแรกจาก y สร้างเป็นพิกัด x,y

ข้อมูลที่ใช้

กราฟเส้นเหมาะกับการใช้แสดงข้อมูลที่มีความต่อเนื่อง เช่น ความสัมพันธ์ของ x และ y = x**2

In [7]:

```
x = np.linspace(1,10,10)

y = x**2
```

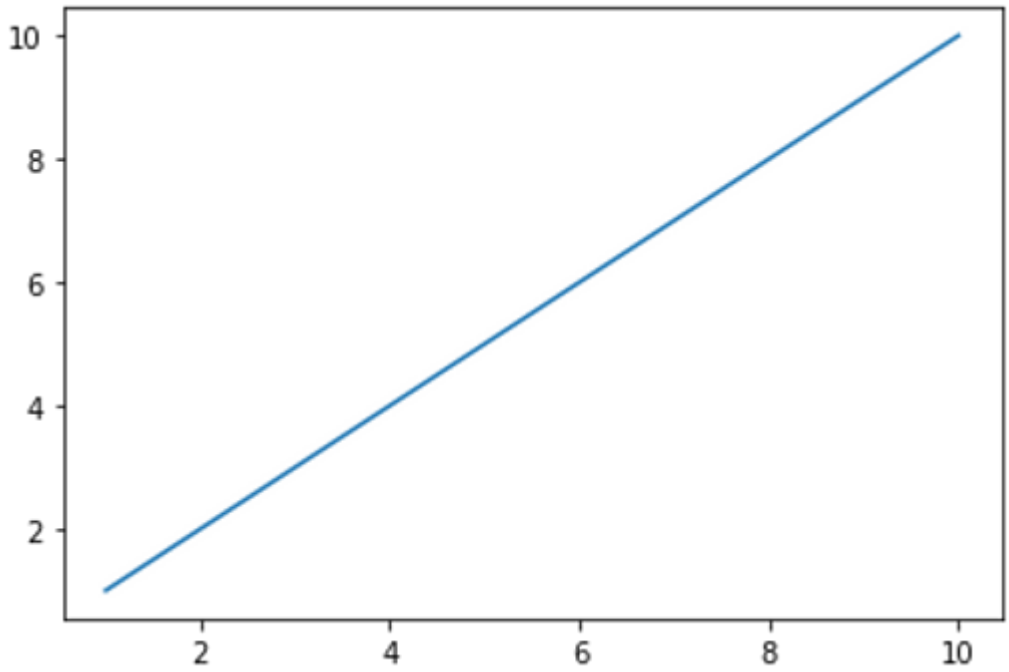
การใช้งาน

เพื่อสร้างกราฟจะใช้การสั่งคำสั่งเพื่อทำสิ่งต่างๆ และตามด้วยคำสั่ง plt.show() เพื่อประมวลผลคำสั่งต่างๆ ที่เกี่ยวข้องกับการแสดงผลกราฟ และแสดงออกมาเป็นกราฟตามคำสั่ง

การ plot กราฟ y = x

In [8]:

```
plt.plot(x,x)

plt.show()
```

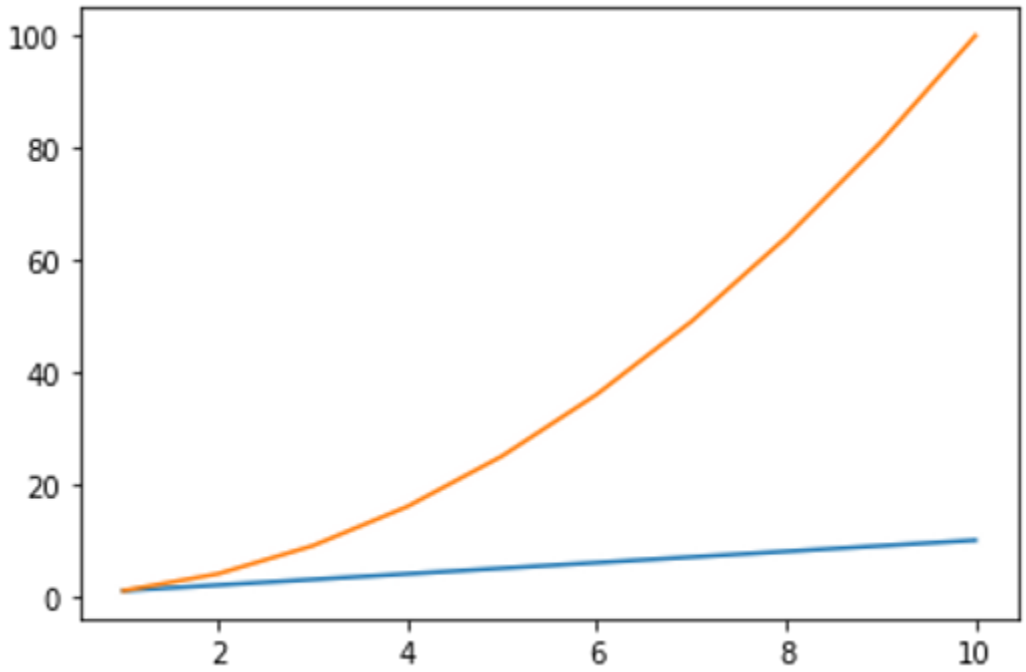


การ plot หลายข้อมูล

ใช้การสั่งคำสั่ง plt.plot() แทนการสร้างกราฟสำหรับ 1 ข้อมูล สามารถสร้างกราฟของกี่ข้อมูลก็ได้ ให้สั่งคำสั่งตามแต่ละลำดับไป เช่น การสร้างกราฟจากข้อมูล y = x และข้อมูล y = x**2 บนกราฟเดียวกัน สามารถเขียนได้ลักษณะดังนี้

In [9]:

```
plt.plot(x,x)

plt.plot(x,x**2)

plt.show()
```



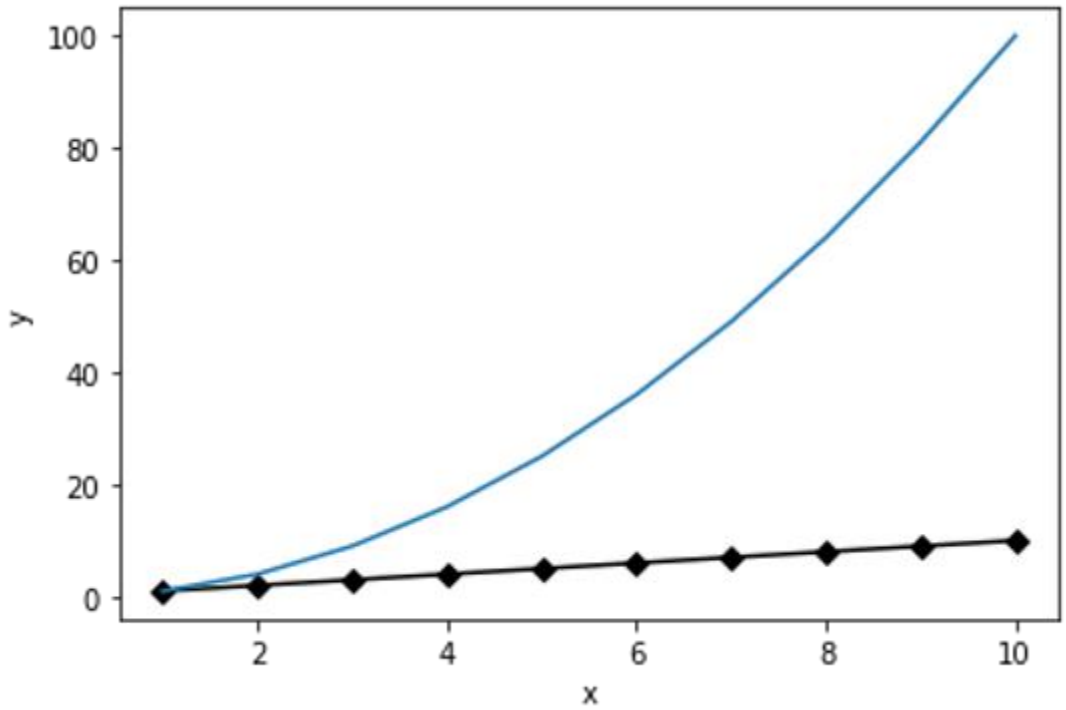
สร้างความแตกต่างแต่ละข้อมูล

ในการอ่านทำความเข้าใจกราฟที่ประกอบไปด้วยหลายข้อมูลอาจทำให้เกิดความสับสนได้ เราสามารถสร้างความแตกต่างของการแสดงผลแต่ละข้อมูลได้ผ่านการเปลี่ยน สี ลักษณะเส้นกราฟ และสัญลักษณ์ ของแต่ละข้อมูล

ผ่านการระบุ string ที่ใส่สัญลักษณ์ของแต่ละค่าเอาไว้ เช่น กราฟเส้นสีดำใช้ 'k' กราฟเส้นทึบใช้ '-' กราฟที่มีสัญลักษณ์ข้าวหลามตัดใช้ 'D' เขียนรวมได้เป็น 'k-D' และระบุเป็น parameter เอาไว้ใน plt.plot() ใช้งานในลักษณะดังนี้

In [10]:

```
plt.plot(x,x,'k-D')

plt.plot(x,x**2)

plt.show()
```

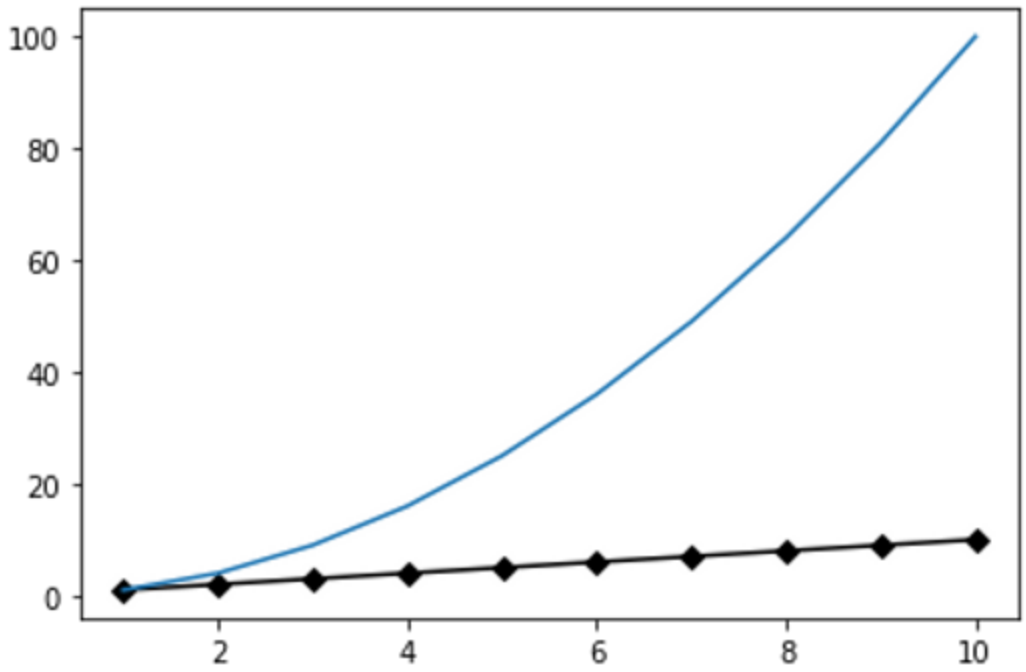


การใส่แกน x,y

เพื่อเพิ่มความเข้าใจค่าที่ระบุในแกน เราสามารถแปะ label สำหรับแกน x,y ได้ด้วยคำสั่ง .xlabel() .ylabel() ที่รับค่า string ที่เก็บชื่อที่ต้องการเอาไว้

In [11]:

```
plt.plot(x,x,'k-D')

plt.plot(x,x**2)

plt.xlabel('x')

plt.ylabel('y')

plt.show()
```



ชื่อแต่ละข้อมูล

เราสามารถแสดงผลหลายข้อมูลบนกราฟได้พร้อมกัน ซึ่งอาจทำให้เกิดความสับสนเกี่ยวกับข้อมูลที่แสดงอยู่ได้ ซึ่งการระบุชื่อของตัวข้อมูลหรือ label เพิ่มเติมจากสี ลักษณะเส้นกราฟ และสัญลักษณ์ พิเศษจะช่วยให้กราฟเข้าใจได้ง่ายขึ้น

label

ในการระบุชื่อข้อมูลที่ต้องการ เราจะใช้การระบุ label เป็น parameter สำหรับ plt.plot() เพื่อตั้งชื่อให้กับข้อมูล ซึ่ง label จะแสดงผลผ่าน legend ซึ่งต้องเขียนคำสั่งเพื่อแสดงผล legend

legend

คือ กล่องข้อความที่แสดงชื่อของแต่ละข้อมูล โดยการระบุผ่านสี ลักษณะเส้นกราฟ และสัญลักษณ์ ซึ่งจะแสดงต่อเมื่อใช้คำสั่ง plt.legend()

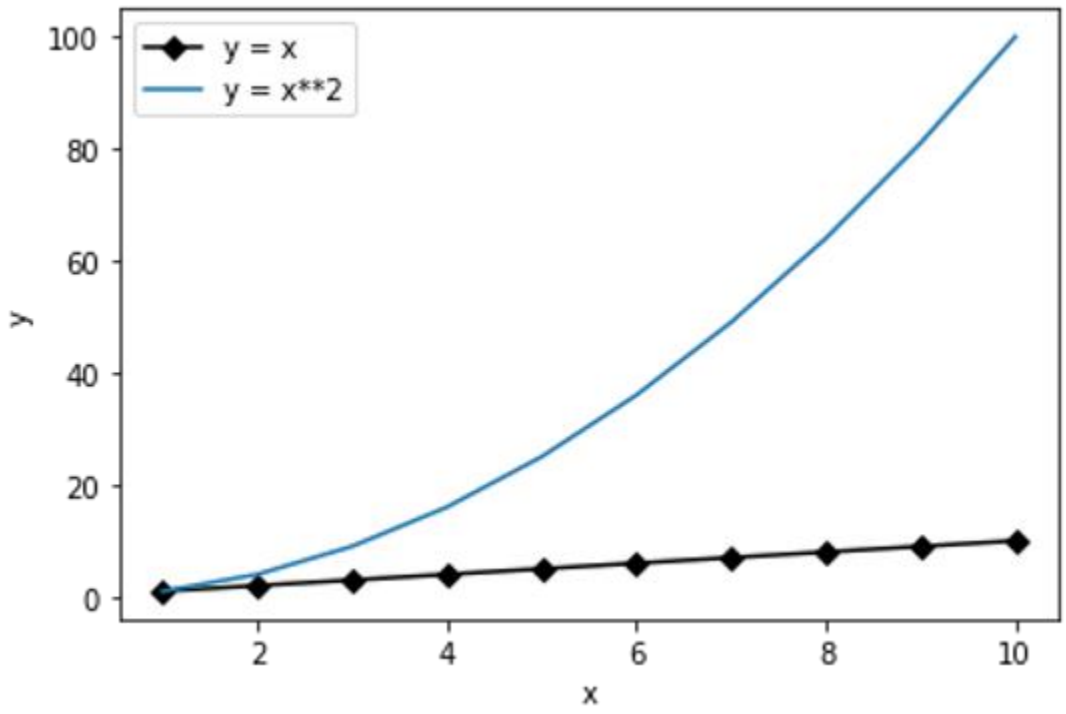
• ตำแหน่งของ legend

สามารถปรับแต่งได้โดยใช้ parameter loc= ตามด้วย string ที่ระบุค่าการกำหนดตำแหน่งที่มีให้เลือก ดังนี้ best, upper right, upper left, lower left, lower right, right, center left, center right, lower center, upper center, center

ซึ่งหากไม่ระบุตำแหน่งจะใช้ default เป็น best คือให้ matplotlib เลือกให้ ว่าควรอยู่ที่ใด

In [12]:

```
plt.plot(x,x,'k-D',label='y = x')

plt.plot(x,x**2,label='y = x**2')

plt.legend(loc='best')

plt.xlabel('x')

plt.ylabel('y')

plt.show()
```

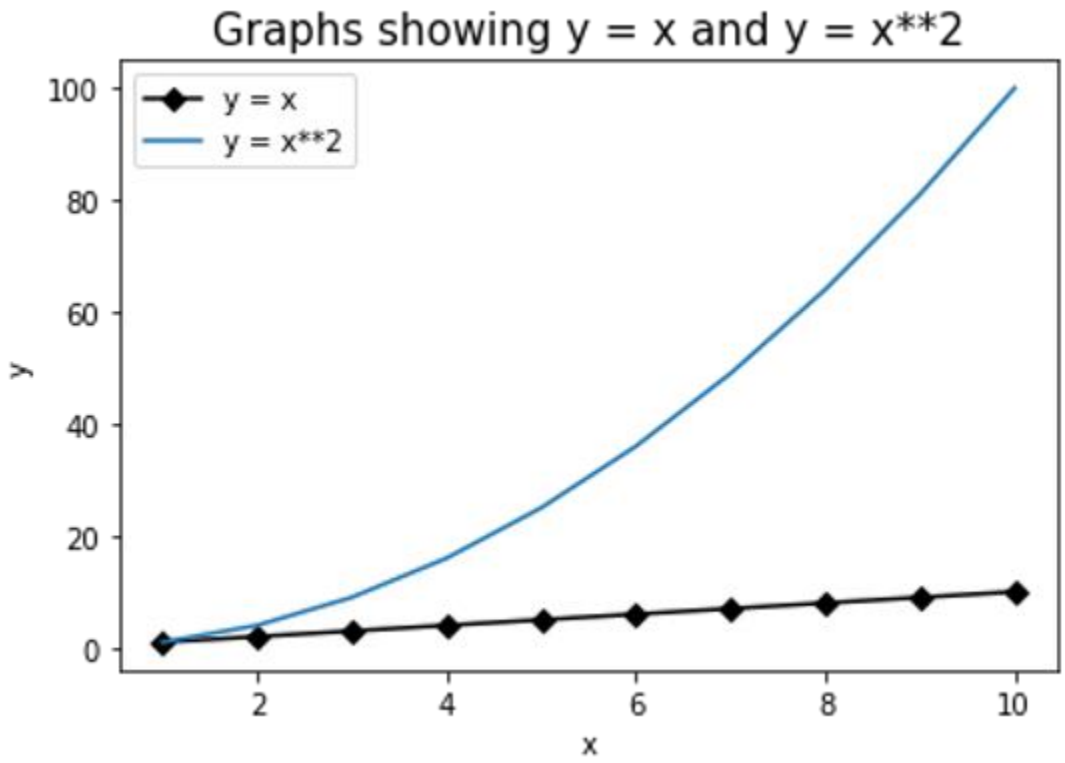


การใส่ชื่อกราฟ

ใช้คำสั่ง plt.title() ที่รับค่า string ที่เก็บข้อมูลชื่อที่ต้องการตั้ง และสามารถใส่ parameter fontsize= เพื่อกำหนดขนาดชื่อกราฟ

In [13]:

```
plt.title("Graphs showing y = x and y = x**2",fontsize=15)

plt.plot(x,x,'k-D',label='y = x')

plt.plot(x,x**2,label='y = x**2')

plt.legend(loc='best')

plt.xlabel('x')

plt.ylabel('y')

plt.show()
```



กราฟแท่ง

ใช้คำสั่ง .bar() สำหรับกราฟแท่งแนวตั้ง และ .barh() สำหรับกราฟแท่งแนวนอน ในการสร้างโดยรับค่า x, y เป็น parameter หลัก และสามารถใช้ร่วมกับการตั้งชื่อกราฟ สร้างความแตกต่างแต่ละข้อมูล ตั้งชื่อข้อมูล และอื่นๆ ได้

ข้อมูลที่ใช้

กราฟแท่งเหมาะกับการใช้เปรียบเทียบข้อมูลที่ไม่ต่อเนื่องกัน เช่น ยอดขายของสินค้าแต่ละประเภท

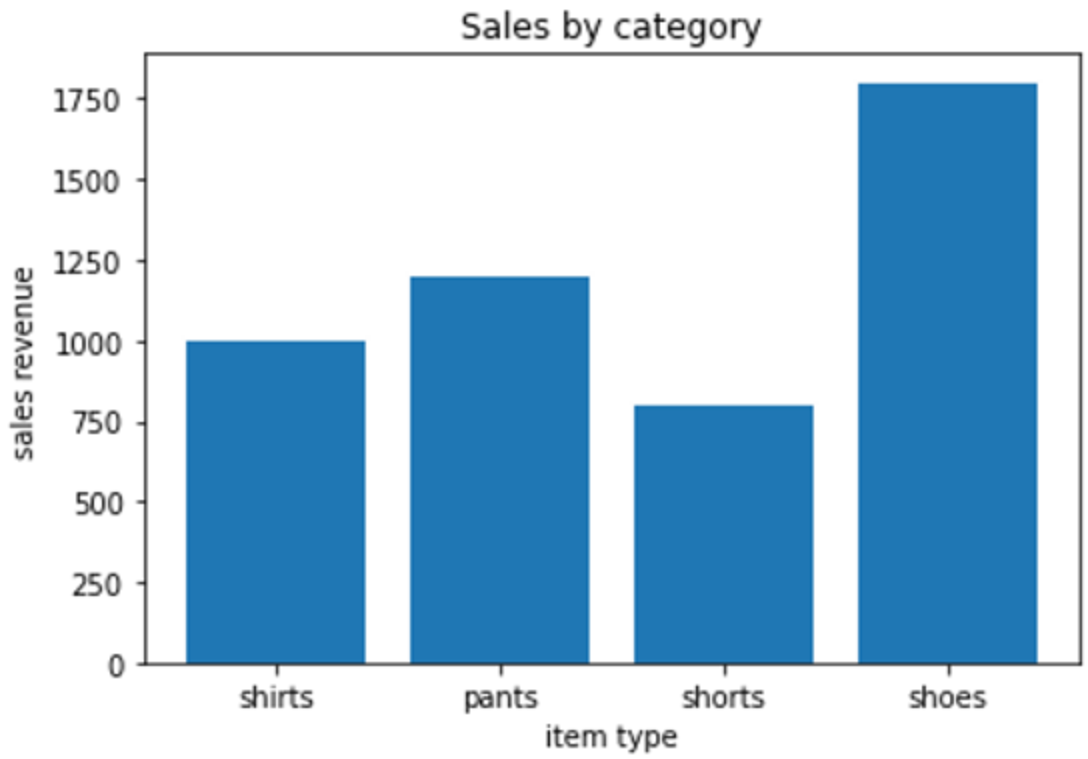
In [14]:

```
x = ['shirts','pants','shorts','shoes']

y = [1000,1200,800,1800]
```

กราฟแท่งแนวตั้ง

In [15]:

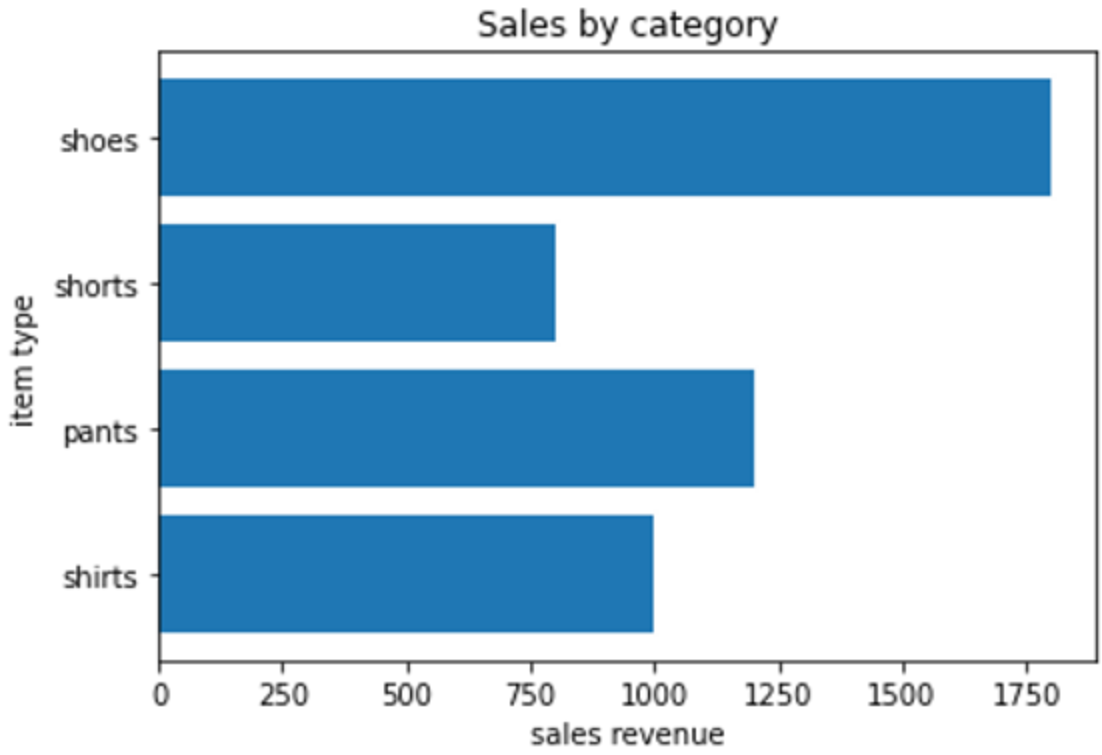
```
plt.bar(x,y)

plt.title('Sales by category')

plt.ylabel('sales revenue')

plt.xlabel('item type')

plt.show()
```



กราฟแท่งแนวนอน

In [16]:

```
plt.barh(x,y)

plt.title('Sales by category')

plt.ylabel('item type')

plt.xlabel('sales revenue')

plt.show()
```



Histogram

เป็นกราฟที่ใช้เพื่อดูความถี่ของการเกิดขึ้นของข้อมูล จะเหมาะกับข้อมูลที่มีจำนวนมาก ซึ่ง Histogram จะมีการแบ่งข้อมูลออกเป็น bins และนับการเกิดของแต่ละ bins เพื่อนับความถี่ของแต่ละ bins ที่เกิดขึ้นและสร้างเป็นกราฟลักษณะคล้ายกราฟแท่ง

ข้อมูลที่ใช้

เราจะสร้างข้อมูล จำนวน 100 ข้อมูล จากการสุ่มจากคำสั่ง np.random.normal() ที่สุ่มข้อมูลมาจาก normal distribution

In [17]:

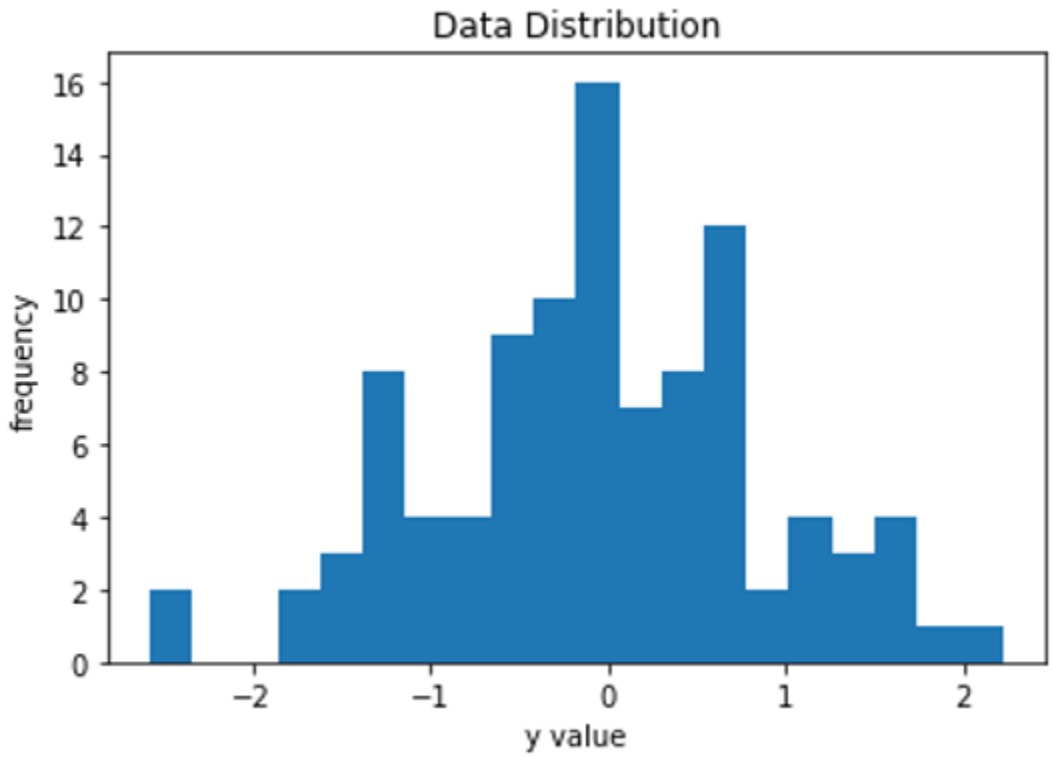
```
x = np.random.normal(size=100)
```

สร้าง histogram

ใช้คำสั่ง .hist() รับค่า x และจำนวน bins หรือจำนวนแท่งกราฟที่ต้องการ โดย plt.hist จะทำการแบ่งข้อมูลเป็นกลุ่มต่างๆ จำนวนเท่ากับ bins และทำการนับแต่ละ bins

In [18]:

```
plt.title('Data Distribution')

plt.hist(x,bins=20)

plt.xlabel('y value')

plt.ylabel('frequency')

plt.show()
```

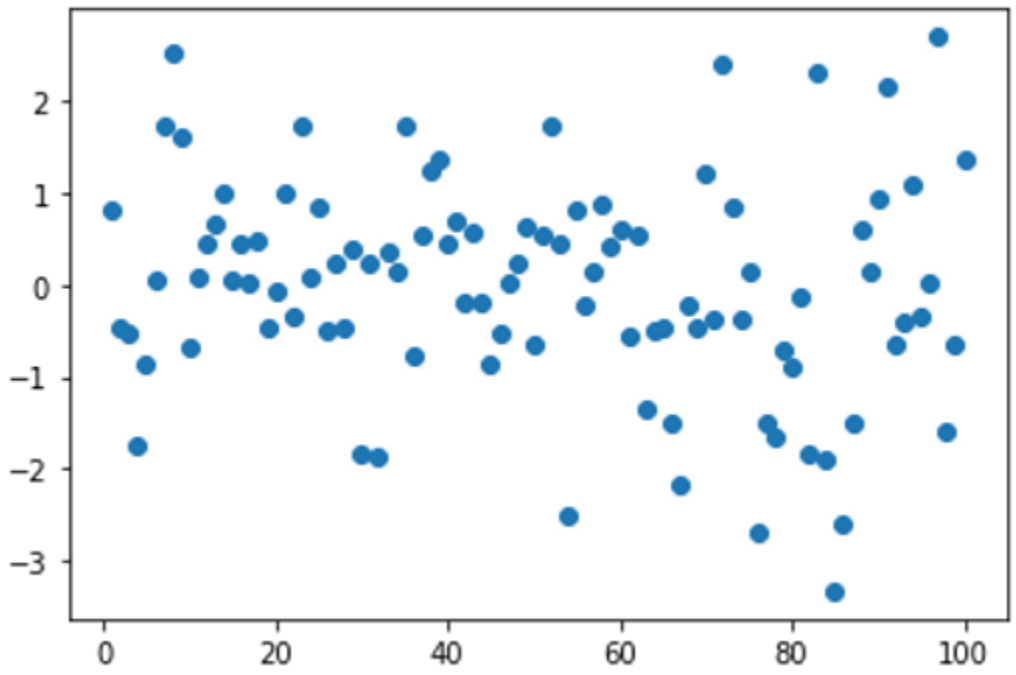


Scatter Plot

นิยมใช้แสดงผลข้อมูลเพื่อดูความสัมพันธ์ระหว่างแกน x,y หรือ ระหว่างในชุดข้อมูลเอง การสร้างใช้คำสั่ง .scatter() ที่รับค่า x, y

In [19]:

```
plt.scatter(np.linspace(1,100,100),np.random.normal(size=100))

plt.show()
```



Box Plot

เป็นกราฟที่ใช้ดูการกระจายตัวของข้อมูล โดยมีพื้นฐานการคำนวนจากการวัด Quartile ซึ่งทำการเรียงข้อมูลทั้งหมดจากน้อยไปมาก และแบ่งข้อมูลออกเป็น 4 ส่วน Q1 Q2 Q3 Q4 โดยมีกล่องตรงกลางแสดงพื้นที่ Q1 ถึง Q3 มีเส้นสีส้มแสดงค่า median และระความยาวของกล่องตรงกลางเรียกว่าค่า IQR (Interquartile-range, Q3-Q1) และมีเส้นตรงขีดออกไปยังค่า Q1-(1.5 x IQR) ในด้านที่ค่าน้อยกว่า และ Q3+(1.5 x IQR) ในด้านที่ค่ามากกว่า โดยหากข้อมูลใดมีค่ามากกว่า หรือน้อยกว่าจุดสิ้นสุดของเส้นตรงดังกล่าว เราจะเรียกข้อมูลเหล่านั้นว่า Outlier

ข้อมูลที่ใช้

ใน boxplot เหมาะกับข้อมูลจำนวนมาก ที่เราต้องการดูการกระจายตัว ในตัวอย่างจะสร้างเลขจำนวน 100 ตัวจากการสุ่มจาก normal distribution

In [20]:

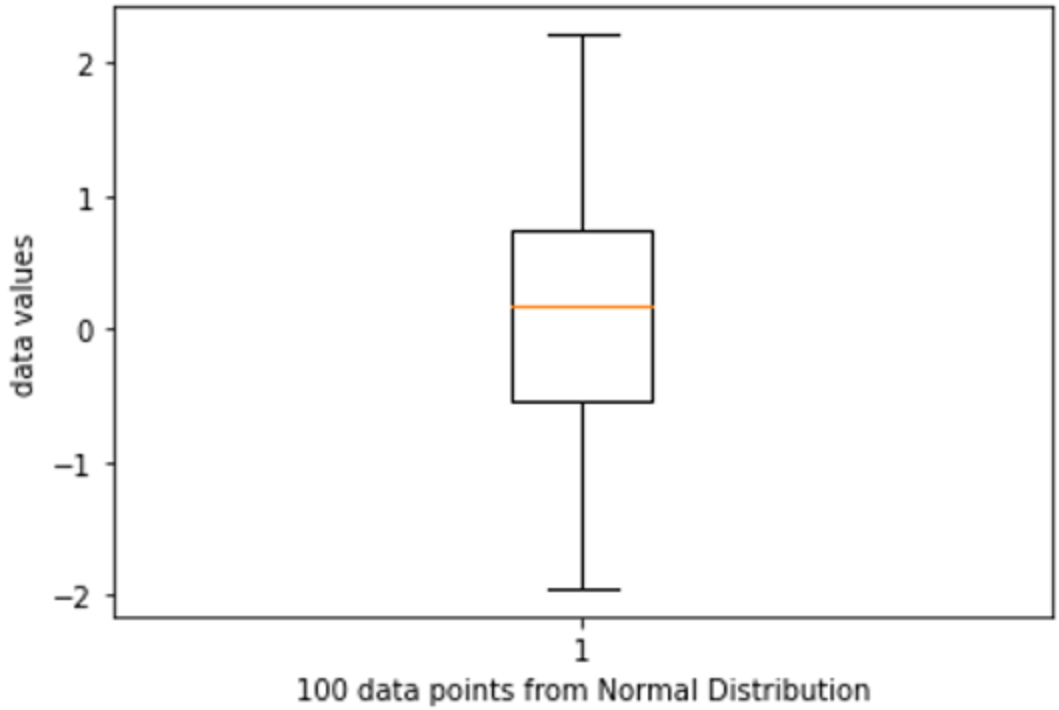
```
x = np.random.normal(size=100)
```

การใช้งาน boxplot

ใช้คำสั่ง .boxplot() โดยรับค่าที่ต้องการดูการกระจายตัวข้อมูล

In [21]:

```
plt.boxplot(x)
plt.xlabel('100 data points from Normal Distribution')
plt.ylabel('data values')
plt.show()
```



การสร้างหลายกราฟ

ใช้การสร้างพื้นที่ในการสร้างกราฟที่แบ่งแต่ละกราฟออกเป็นส่วนๆ และทำการใช้คำสั่งสร้างกราฟที่เราเรียนไปต่างๆ สร้างกราฟลงไป

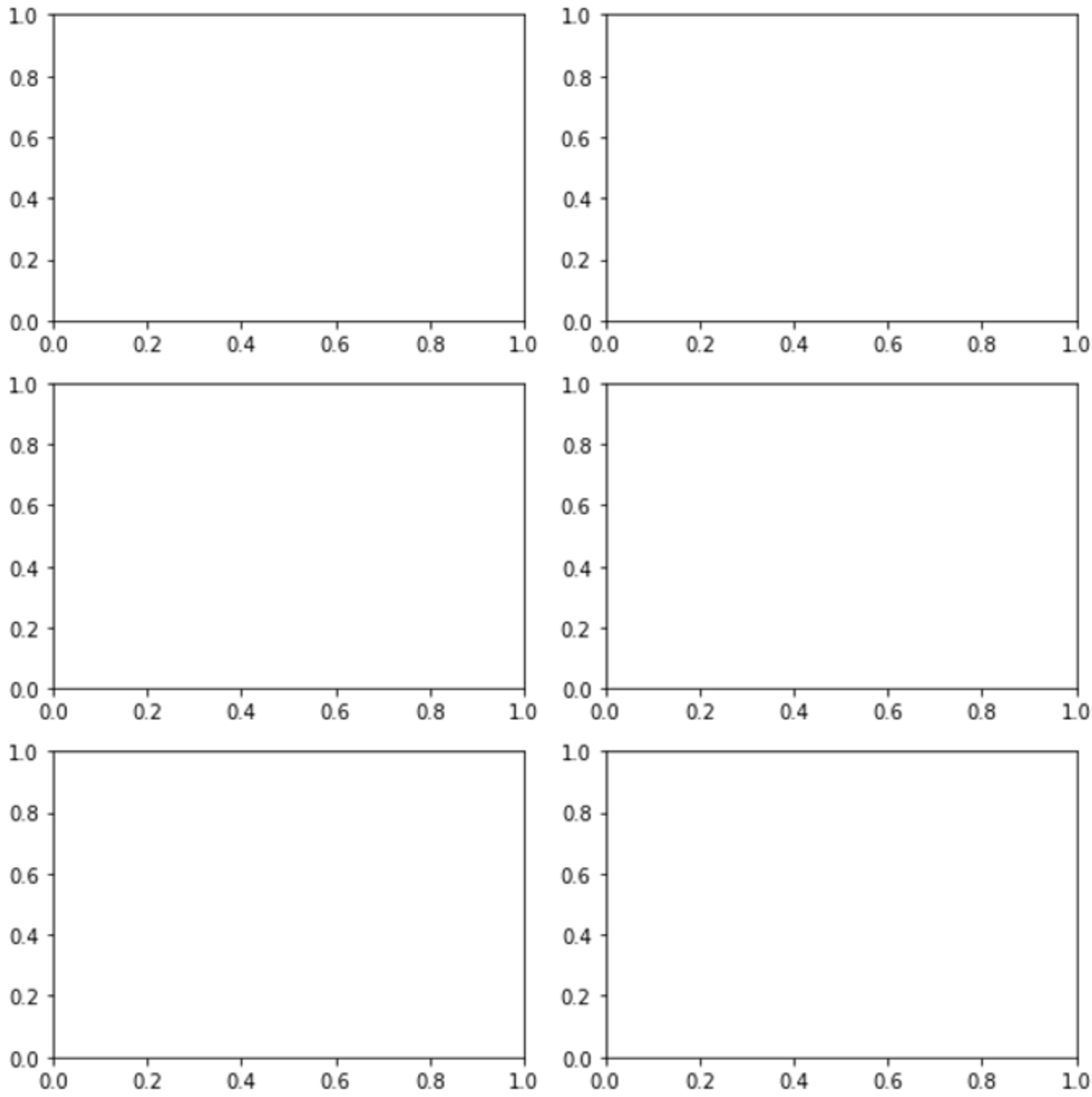
สร้างพื้นที่การทำงาน

ด้วยคำสั่ง plt.subplots() โดยให้กำหนด nrows= กำหนดจำนวนแถว ncols= กำหนดจำนวนกราฟแต่ละแถว และ figsize= กำหนดขนาดของกราฟทั้งหมดเป็น tuple ที่ประกอบไปด้วย (ความกว้าง,ความสูง)

เช่น การสร้างพื้นที่เพื่อสร้าง 6 กราฟ โดยแบ่งเป็น 3 แถว แถวละ 2 กราฟ โดยมีขนาด 8 x 8 โดยเราจะสร้างตัวแปร fig, axes มา unpack ข้อมูลที่ได้จาก คำสั่ง .subplots()

In [22]:

```
fig, axes = plt.subplots(nrows=3,ncols=2,figsize=(8,8))

plt.tight_layout()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

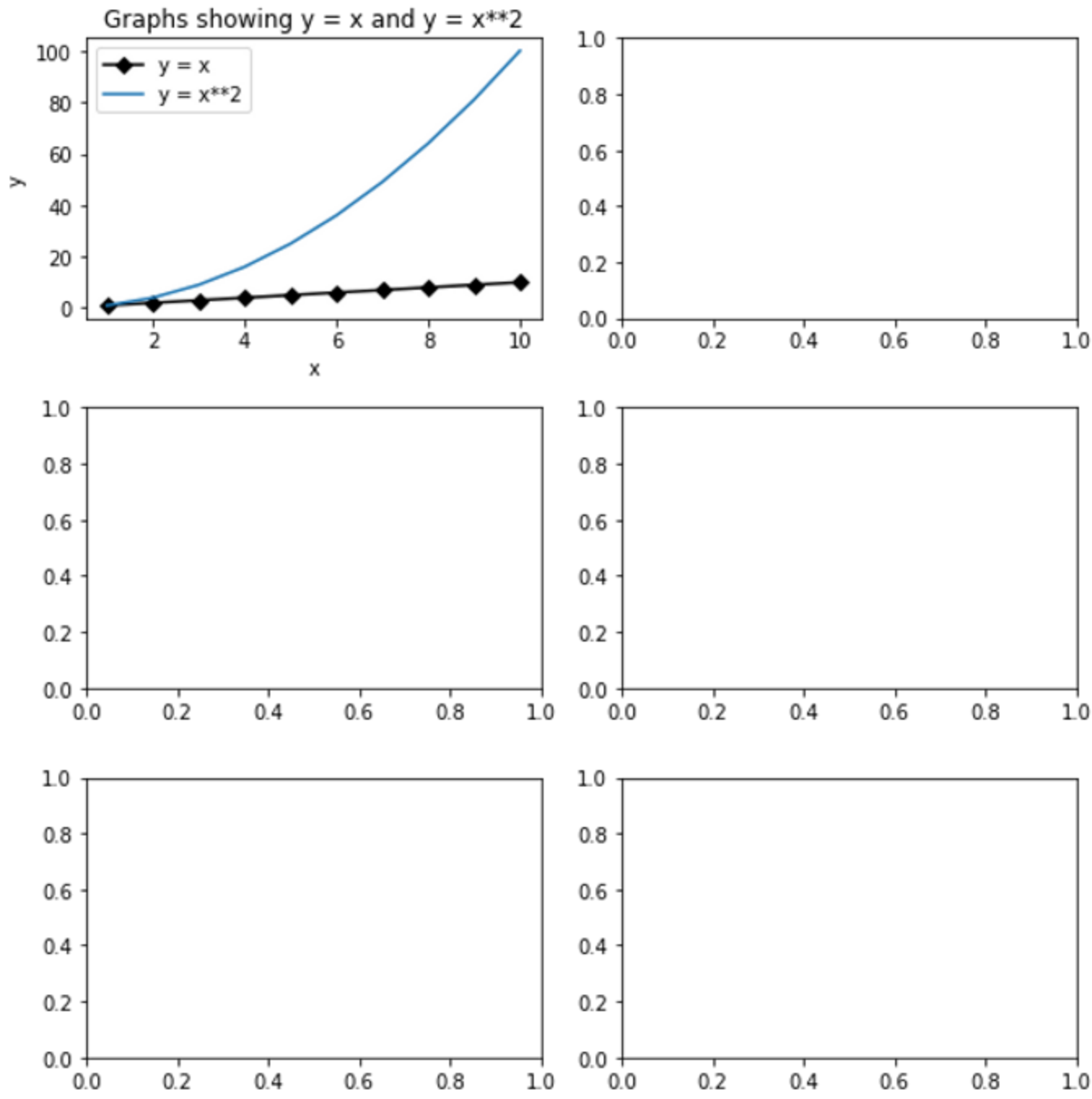
ใส่กราฟลงพื้นที่

โดยใช้ข้อมูลในตัวแปร axes ตามด้วยการระบุตำแหน่งตามหลังด้วยสัญลักษณ์ [แถว,คอลัมน์] เพื่อระบุตำแหน่งของกราฟที่ต้องการใส่ ตำแหน่งแรกเริ่มจาก 0

เช่น การใส่กราฟเส้นลงไปที่ตำแหน่ง [0,0] หรือ ซ้ายบน และตามด้วยคำสั่ง .plot() (เพื่อสร้างกราฟเส้น) และเราจะใช้คำสั่ง plt.tight_layout() เพื่อปรับขนาดให้ตัวอักษรไม่ทับกันโดยอัตโนมัติ

In [23]:

```
fig, axes = plt.subplots(nrows=3,ncols=2,figsize=(8,8))


axes[0,0].set_title("Graphs showing y = x and y = x**2")

axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10),'k-D',label='y = x')

axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10)**2,label='y = x**2')

axes[0,0].legend(loc='best')

axes[0,0].set_xlabel('x')

axes[0,0].set_ylabel('y')


plt.tight_layout()
```

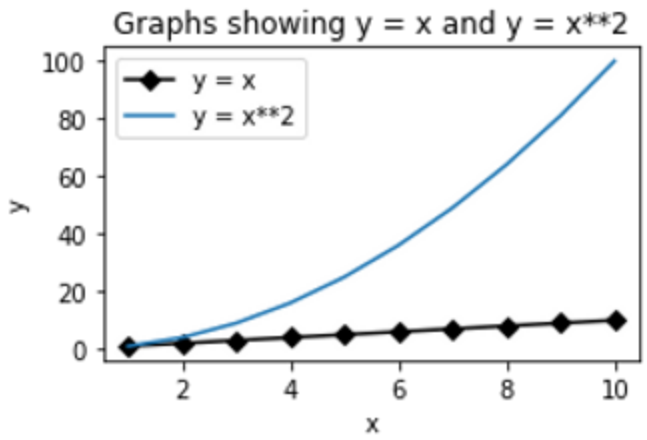

Graphs showing y = x and y = x**2
y = x
y = x**2
x
y

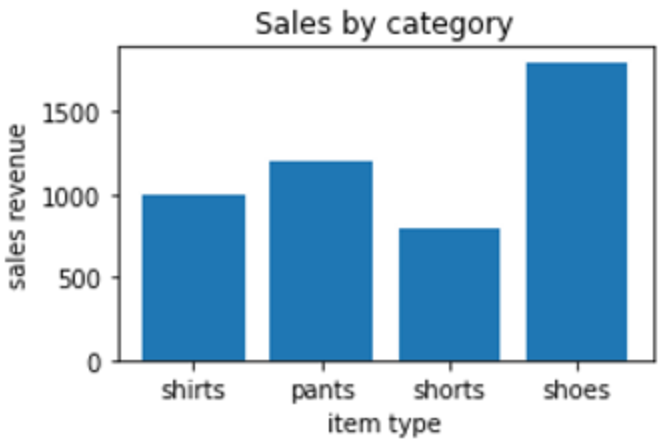
ทำซ้ำจนครบทุกกราฟ

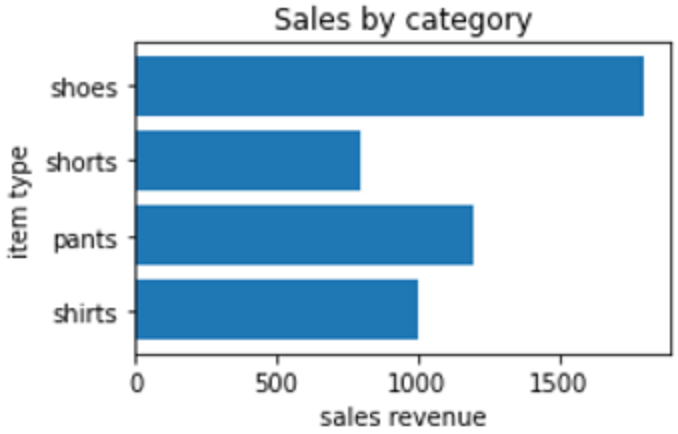
In [24]:

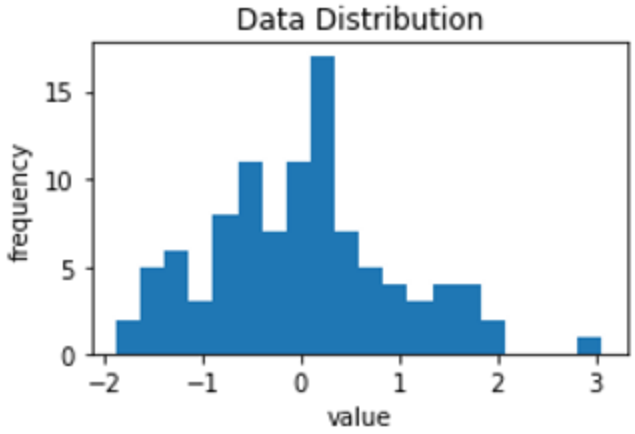
```
fig, axes = plt.subplots(nrows=3,ncols=2,figsize=(8,8))


axes[0,0].set_title("Graphs showing y = x and y = x**2")
axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10),'k-D',label='y = x')
axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10)**2,label='y = x**2')
axes[0,0].legend(loc='best')
axes[0,0].set_xlabel('x')
axes[0,0].set_ylabel('y')


axes[0,1].set_title('Sales by category')
axes[0,1].bar(['shirts','pants','shorts','shoes'],[1000,1200,800,1800])
axes[0,1].set_ylabel('sales revenue')
axes[0,1].set_xlabel('item type')


axes[1,0].set_title('Sales by category')
axes[1,0].barh(['shirts','pants','shorts','shoes'],[1000,1200,800,1800])
axes[1,0].set_ylabel('item type')
axes[1,0].set_xlabel('sales revenue')


axes[1,1].set_title('Data Distribution')
```

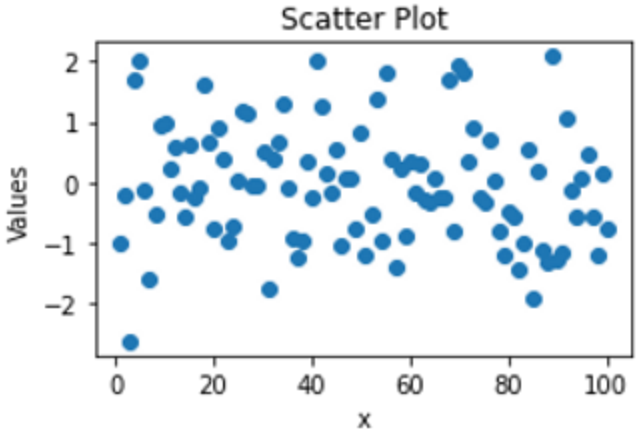
```
axes[1,1].hist(np.random.normal(size=100),bins=20)

axes[1,1].set_xlabel('value')

axes[1,1].set_ylabel('frequency')


axes[2,0].set_title('Scatter Plot')

axes[2,0].scatter(np.linspace(1,100,100),np.random.normal(size=100))

axes[2,0].set_xlabel('x')

axes[2,0].set_ylabel('Values')


axes[2,1].set_title('Boxplot')

axes[2,1].boxplot(np.random.normal(size=100))

axes[2,1].set_xlabel('100 Randomly Generated')

axes[2,1].set_ylabel('Values')


plt.tight_layout()
```

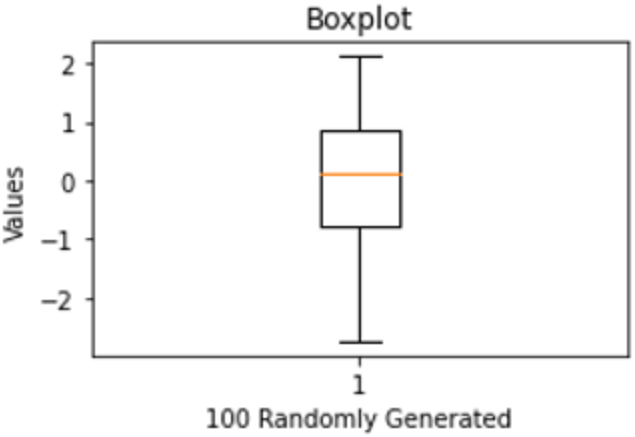

Graphs showing y = x and y = x**2
y = x
y = x**2
y
x
0
20
40
60
80
100
2
4
6
8
10


Sales by category
sales revenue
item type
shirts
pants
shorts
shoes
0
500
1000
1500


Sales by category
item type
sales revenue
shoes
shorts
pants
shirts
0
500
1000
1500


Data Distribution
frequency
value
0
5
10
15
−2
−1
0
1
2
3


Scatter Plot
Values
x
0
20
40
60
80
100
2
1
0
−1
−2


Boxplot
Values
100 Randomly Generated
1
2
1
0
−1
−2

เปลี่ยน style

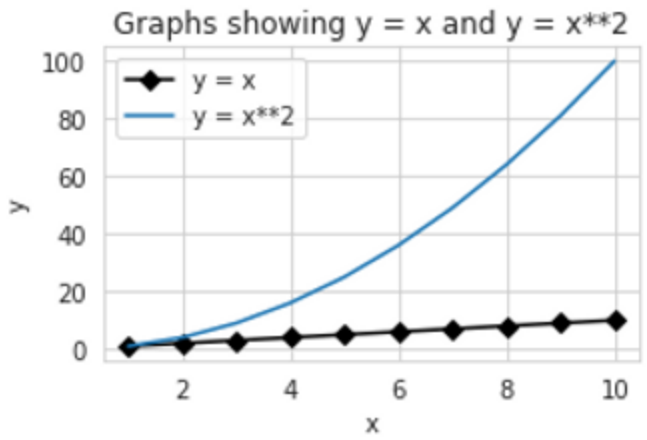
สามารถทำได้โดยการใช้ style ที่ถูกสร้างไว้สำเร็จรูป ซึ่ง seaborn เป็นหนึ่งใน library ที่ใช้ตกแต่ง รวมไปถึง plot กราฟในรูปแบบต่างๆ ที่เราจะใช้เพื่อเปลี่ยนรูปแบบการแสดงผล

เราจะใช้คำสั่ง sns.set_style() และกำหนด parameter style= ซึ่งมีให้เลือกเบื้องต้น white, dark, whitegrid, darkgrid, ticks สามารถใช้งานได้ดังนี้

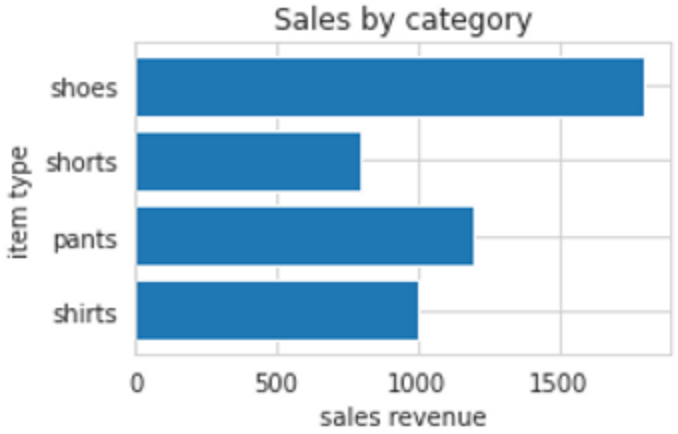
In [25]:

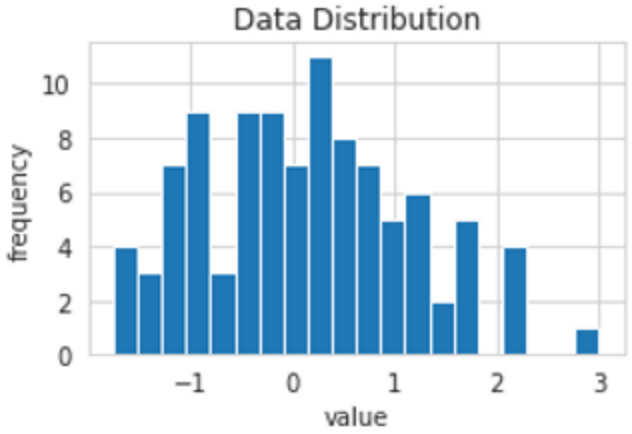
```
import seaborn as sns

sns.set_style(style='whitegrid')

fig, axes = plt.subplots(nrows=3,ncols=2,figsize=(8,8))

axes[0,0].set_title("Graphs showing y = x and y = x**2")

axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10),'k-D',label='y = x')

axes[0,0].plot(np.linspace(1,10,10),np.linspace(1,10,10)**2,label='y = x**2')

axes[0,0].legend(loc='best')

axes[0,0].set_xlabel('x')

axes[0,0].set_ylabel('y')

axes[0,1].set_title('Sales by category')

axes[0,1].bar(['shirts','pants','shorts','shoes'],[1000,1200,800,1800])

axes[0,1].set_ylabel('sales revenue')

axes[0,1].set_xlabel('item type')

axes[1,0].set_title('Sales by category')

axes[1,0].barh(['shirts','pants','shorts','shoes'],[1000,1200,800,1800])

axes[1,0].set_ylabel('item type')
```

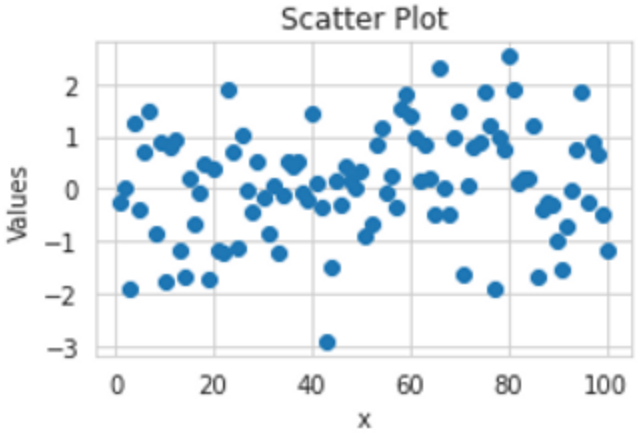
```
axes[1,0].set_xlabel('sales revenue')

axes[1,1].set_title('Data Distribution')

axes[1,1].hist(np.random.normal(size=100),bins=20)

axes[1,1].set_xlabel('value')

axes[1,1].set_ylabel('frequency')

axes[2,0].set_title('Scatter Plot')

axes[2,0].scatter(np.linspace(1,100,100),np.random.normal(size=100))

axes[2,0].set_xlabel('x')

axes[2,0].set_ylabel('Values')

axes[2,1].set_title('Boxplot')

axes[2,1].boxplot(np.random.normal(size=100))

axes[2,1].set_xlabel('100 Randomly Generated')

axes[2,1].set_ylabel('Values')

plt.tight_layout()
```

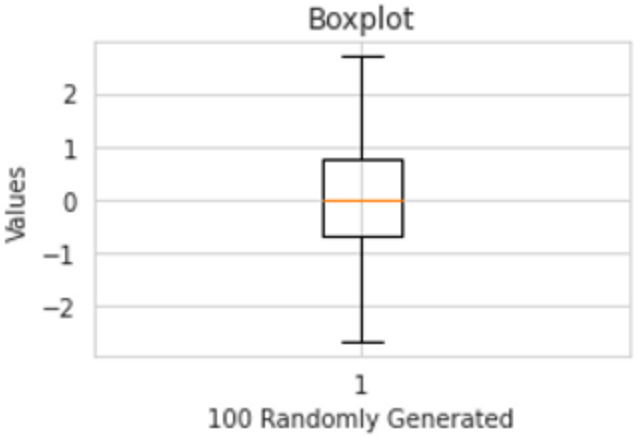

Graphs showing y = x and y = x**2
y = x
y = x**2
y
x
0
20
40
60
80
100
2
4
6
8
10


Sales by category
sales revenue
item type
shirts
pants
shorts
shoes
0
500
1000
1500


Sales by category
item type
sales revenue
shoes
shorts
pants
shirts
0
500
1000
1500


Data Distribution
frequency
value
0
2
4
6
8
10
−1
0
1
2
3


Scatter Plot
Values
x
2
1
0
−1
−2
−3
0
20
40
60
80
100

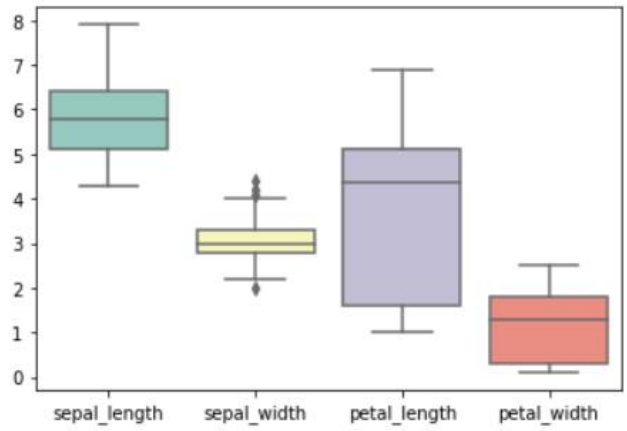

Boxplot
Values
100 Randomly Generated
2
1
0
−1
−2
1

Seaborn

Seaborn เป็นไลบรารีสำหรับสร้างกราฟิกทางสถิติใน Python สร้างขึ้นจาก matplotlib และทำงานร่วมกันอย่างใกล้ชิดกับ โครงสร้างข้อมูล Pandas ด้วย Seaborn Design สามารถสำรวจและทำความเข้าใจข้อมูลได้อย่างรวดเร็ว Seaborn รวบรวมเฟรมข้อมูลหรืออาร์เรย์ทั้งหมดที่มีข้อมูลทั้งหมดและทำหน้าที่ภายในทั้งหมดที่จำเป็นสำหรับการแมปความหมายและการรวมสถิติเพื่อเปลี่ยนข้อมูลให้เป็นแผนภูมิข้อมูลเป็นนามธรรมความซับซ้อนและช่วยให้สามารถออกแบบแปลงตามความต้องการได้

ตัวอย่างการใช้งาน Seaborn

```
In [37]: import seaborn as sns
         iris=sns.load_dataset("iris")
         ax=sns.boxplot (data=iris,orient ="V",palette ="Set3")
```



Pandas

Pandas เป็น software library ที่ถูกเขียนด้วยภาษา Python ใช้ในการจัดการและวิเคราะห์ข้อมูล หรือ data โดยมี data structures และ operations ต่างๆ ที่ใช้ในการจัดการกับตารางตัวเลขและ time series

Feature ของ Library Pandas

- DataFrame Object สำหรับจัดการข้อมูลด้วยการทำดัชนีแบบรวม
- เครื่องมือสำหรับอ่านและเขียนข้อมูลระหว่างโครงสร้างข้อมูลที่อยู่ในหน่วยความจำและไฟล์ใน Formats อื่นๆ
- การจัดตำแหน่งข้อมูลและรวบรวมข้อมูลสำหรับ Missing Data หรือ ข้อมูลที่ขาดหายไป
- การ Reshape และการ pivot สำหรับชุดหรือข้อมูลหรือ Data set
- การทำ Label สำหรับการ Slicing,การทำ Indexing หรือ การทำดัชนี การย่อยชุดข้อมูลที่มีขนาดใหญ่
- การแทรกและการลบคอลัมน์ ของ Data Structure
- จัดกลุ่มตาม Engine เพื่อให้สามารถใช้ดำเนินการ Split-apply-combine กับ Data set
- การ Merging และ Joining ของ Data set
- การ Indexing แกนแบบตามลำดับชั้น เพื่อในการทำงานกับข้อมูลแบบ High-dimensional ใน Data structure ที่มีมิติต่ำกว่า
- ฟังก์ชั่น Time series: การสร้าง Range ของวันและความถี่ของการเปลี่ยนแปลง,การย้าย Window statistics,การย้าย Window linear regression,การ Shift วัน และ การ Lagging

Dataframes

หลักๆแล้ว Pandas จะถูกใช้สำหรับการวิเคราะห์ข้อมูลโดยสามารถนำข้อมูลเข้ามาได้หลายรูปแบบ เช่น JSON,SQL,Microsoft Excel เป็นต้น และ มี Operations มากมาที่ใช้ในการเก็บข้อมูล

การสร้าง dataframe โดยกำหนดข้อมูลด้วย Numpy array หรือ Dictionary สามารถกำหนด Index ได้ด้วย Argument ชื่อ Index เช่น

1. DataFrame

```
In [1]: import pandas as pd
        import numpy as np

In [2]: data = np.array([
            ("Alice", 21, "F"),
            ("Bob", 32, "M"),
            ("Carol", 27, "F"),
            ("David", 24, "M"),
            ["Eve", 18, "F"]
        ])

        df0 = pd.DataFrame(data, columns=("Name", "Age", "Gender"))
        df0
```

Out[2]:

	Name	Age	Gender
0	Alice	21	F
1	Bob	32	M
2	Carol	27	F
3	David	24	M
4	Eve	18	F

Read_csv

สร้าง Dataframe โดยอ่านจากไฟล์ CSV สามารถกำหนด Index ด้วย Index_col เช่น

2. read_csv

```
In [4]: df = pd.read_csv("mars-weather.csv", index_col="terrestrial_date")
        df
```

Out[4]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny
...	...	...	...	...	...	...	...	...	...
2012-08-18	24	12	156	Month 6	-76.0	-18.0	741.0	NaN	Sunny
2012-08-17	13	11	156	Month 6	-76.0	-11.0	740.0	NaN	Sunny
2012-08-16	2	10	155	Month 6	-75.0	-16.0	739.0	NaN	Sunny
2012-08-15	232	9	155	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-07	1	1	150	Month 6	NaN	NaN	NaN	NaN	Sunny

1894 rows × 9 columns

Columns & dtypes

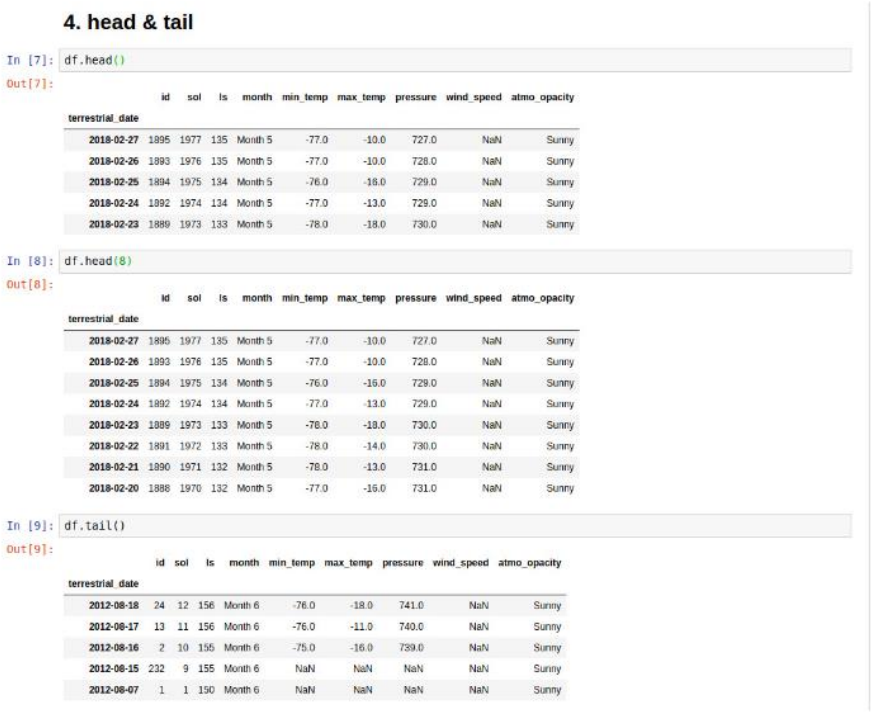
Column ใช้ดูชื่อ Column ทั้งหมดใน Dataframe dtypes ใช้สำหรับดูชนิดของข้อมูลแต่ละ Column ใน Dataframe

3. columns & dtypes

```
In [5]: df.columns
Out[5]: Index(['id', 'sol', 'ls', 'month', 'min_temp', 'max_temp', 'pressure',
               'wind_speed', 'atmo_opacity'],
              dtype='object')

In [6]: df.dtypes
Out[6]: id              int64
        sol             int64
        ls              int64
        month          object
        min_temp      float64
        max_temp      float64
        pressure      float64
        wind_speed    float64
        atmo_opacity   object
        dtype: object
```

Head & tail

ดูตัวอย่างข้อมูลด้วยคำสั่ง Head หรือ tail โดย head ใช้ดูสำหรับช่วงต้นของ Dataframe และ tail ใช้สำหรับดูช่วงท้ายของ Dataframe สามารถระบุช่วงจำนวนตัวอย่างที่ต้องการได้ โดยถ้าไม่ระบุคำสั่งจะแสดง 5 ตัวอย่าง



4. head & tail

```
In [7]: df.head()
```

Out[7]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny

```
In [8]: df.head(8)
```

Out[8]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny
2018-02-22	1891	1972	133	Month 5	-78.0	-14.0	730.0	NaN	Sunny
2018-02-21	1890	1971	132	Month 5	-78.0	-13.0	731.0	NaN	Sunny
2018-02-20	1888	1970	132	Month 5	-77.0	-16.0	731.0	NaN	Sunny

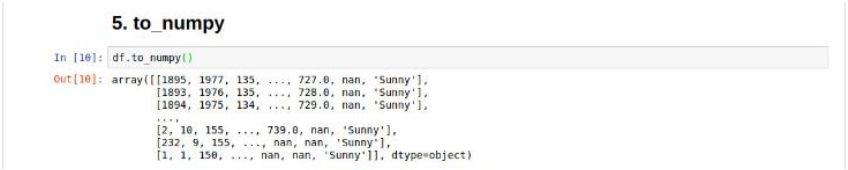
```
In [9]: df.tail()
```

Out[9]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2012-08-18	24	12	156	Month 6	-76.0	-18.0	741.0	NaN	Sunny
2012-08-17	13	11	156	Month 6	-76.0	-11.0	740.0	NaN	Sunny
2012-08-16	2	10	155	Month 6	-75.0	-16.0	739.0	NaN	Sunny
2012-08-15	232	9	155	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-07	1	1	150	Month 6	NaN	NaN	NaN	NaN	Sunny

to_numpy

แปลง Dataframe เป็น Numpy Array

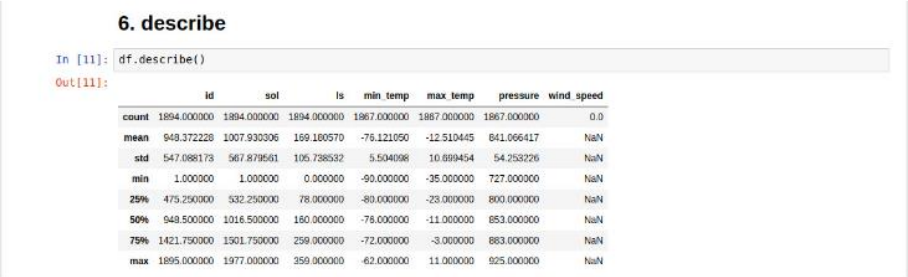


5. to_numpy

```
In [10]: df.to_numpy()
Out[10]: array([[1895, 1977, 135, ..., 727.0, nan, 'Sunny'],
                [1893, 1976, 135, ..., 728.0, nan, 'Sunny'],
                [1894, 1975, 134, ..., 729.0, nan, 'Sunny'],
                ...,
                [2, 10, 155, ..., 739.0, nan, 'Sunny'],
                [232, 9, 155, ..., nan, nan, 'Sunny'],
                [1, 1, 150, ..., nan, nan, 'Sunny']], dtype=object)
```

Describe

สรุปสถิติเบื้องต้นของข้อมูลแต่ละ column ใน DataFrame เช่น ค่าเฉลี่ย ค่า standard deviation (std) ค่าต่ำสุด ค่าสูงสุด



6. describe

```
In [11]: df.describe()
```

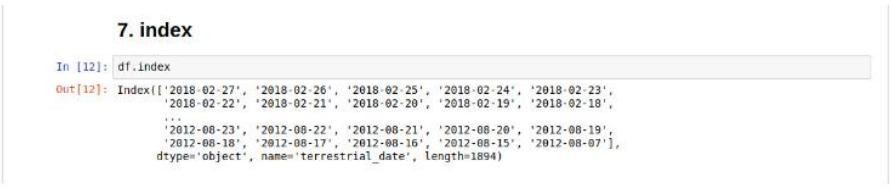
Out[11]:

	id	sol	ls	min_temp	max_temp	pressure	wind_speed
count	1894.000000	1894.000000	1894.000000	1867.000000	1867.000000	1867.000000	0.0
mean	948.372228	1007.930306	169.180570	-76.121050	-12.510445	841.066417	NaN
std	547.088173	567.879561	105.738532	5.504098	10.699454	54.253226	NaN
min	1.000000	1.000000	0.000000	-90.000000	-35.000000	727.000000	NaN
25%	475.250000	532.250000	78.000000	-80.000000	-23.000000	800.000000	NaN
50%	948.500000	1016.500000	160.000000	-76.000000	-11.000000	853.000000	NaN
75%	1421.750000	1501.750000	259.000000	-72.000000	-3.000000	883.000000	NaN
max	1895.000000	1977.000000	359.000000	-62.000000	11.000000	925.000000	NaN

Index

แสดง index ของ DataFrame

ถ้าเราไม่ได้กำหนด index ตอนสร้าง DataFrame เราจะได้เลขลำดับ 0, 1, 2, 3, … เป็น index



7. index

```
In [12]: df.index
Out[12]: Index(['2018-02-27', '2018-02-26', '2018-02-25', '2018-02-24', '2018-02-23',
                '2018-02-22', '2018-02-21', '2018-02-20', '2018-02-19', '2018-02-18',
                ...
                '2012-08-23', '2012-08-22', '2012-08-21', '2012-08-20', '2012-08-19',
                '2012-08-18', '2012-08-17', '2012-08-16', '2012-08-15', '2012-08-07'],
               dtype='object', name='terrestrial_date', length=1894)
```

Sort_index & sort_values

sort_index เรียงลำดับ row ตามค่าของ index

sort_values เรียงลำดับ row ตามค่าของ column ที่กำหนด

8. sort_index & sort_values

```
In [13]: df.sort_index()
```

Out[13]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2012-08-07	1	1	150	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-15	232	9	155	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-16	2	10	155	Month 6	-75.0	-16.0	739.0	NaN	Sunny
2012-08-17	13	11	156	Month 6	-76.0	-11.0	740.0	NaN	Sunny
2012-08-18	24	12	156	Month 6	-76.0	-18.0	741.0	NaN	Sunny
...	...	...	...	...	...	...	...	...	...
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny

1894 rows × 9 columns

```
In [14]: df.sort_values(by="max_temp")
```

Out[14]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2015-11-16	1096	1165	68	Month 3	-84.0	-35.0	897.0	NaN	Sunny
2015-11-15	1095	1164	68	Month 3	-86.0	-35.0	896.0	NaN	Sunny
2014-02-18	489	546	91	Month 4	-85.0	-34.0	855.0	NaN	Sunny
2015-11-07	1087	1157	65	Month 3	-82.0	-32.0	900.0	NaN	Sunny
2017-10-25	1773	1855	78	Month 3	-78.0	-32.0	861.0	NaN	Sunny
...	...	...	...	...	...	...	...	...	...
2012-08-26	110	20	161	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-25	101	19	160	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-24	90	18	160	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-15	232	9	155	Month 6	NaN	NaN	NaN	NaN	Sunny
2012-08-07	1	1	150	Month 6	NaN	NaN	NaN	NaN	Sunny

1894 rows × 9 columns

df[ ] & df.

เลือก column ที่ต้องการตามชื่อ column โดยสามารถใส่เป็น string ใน [ ] หรือเป็นชื่อ attribute ก็ได้

9. df[ ] & df.

```
In [15]: df["max_temp"]
```

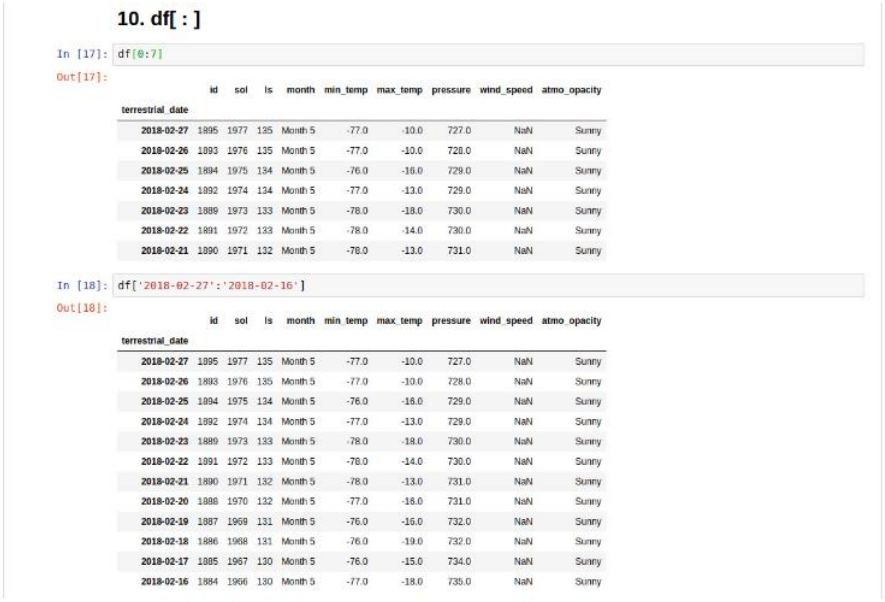
```
Out[15]: terrestrial_date
2018-02-27   -10.0
2018-02-26   -10.0
2018-02-25   -16.0
2018-02-24   -13.0
2018-02-23   -18.0
             ...
2012-08-18   -18.0
2012-08-17   -11.0
2012-08-16   -16.0
2012-08-15     NaN
2012-08-07     NaN
Name: max_temp, Length: 1894, dtype: float64
```

```
In [16]: df.max_temp
```

```
Out[16]: terrestrial_date
2018-02-27   -10.0
2018-02-26   -10.0
2018-02-25   -16.0
2018-02-24   -13.0
2018-02-23   -18.0
             ...
2012-08-18   -18.0
2012-08-17   -11.0
2012-08-16   -16.0
2012-08-15     NaN
2012-08-07     NaN
Name: max_temp, Length: 1894, dtype: float64
```

df[:]

เลือกช่วงของ row ที่ต้องการ โดยใส่เป็นเลขลำดับบรรทัด หรือใส่ตาม index
ในกรณีนี้ index คือวันที่ และเรามีข้อมูลเรียงจากวันที่ใหม่กว่าไปวันที่เก่ากว่า ดังนั้นเราจึงใส่ index ของวันที่แบบถอยหลัง



10. df[:]

```
In [17]: df[0:7]
```

Out[17]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny
2018-02-22	1891	1972	133	Month 5	-78.0	-14.0	730.0	NaN	Sunny
2018-02-21	1890	1971	132	Month 5	-78.0	-13.0	731.0	NaN	Sunny

```
In [18]: df['2018-02-27':'2018-02-16']
```

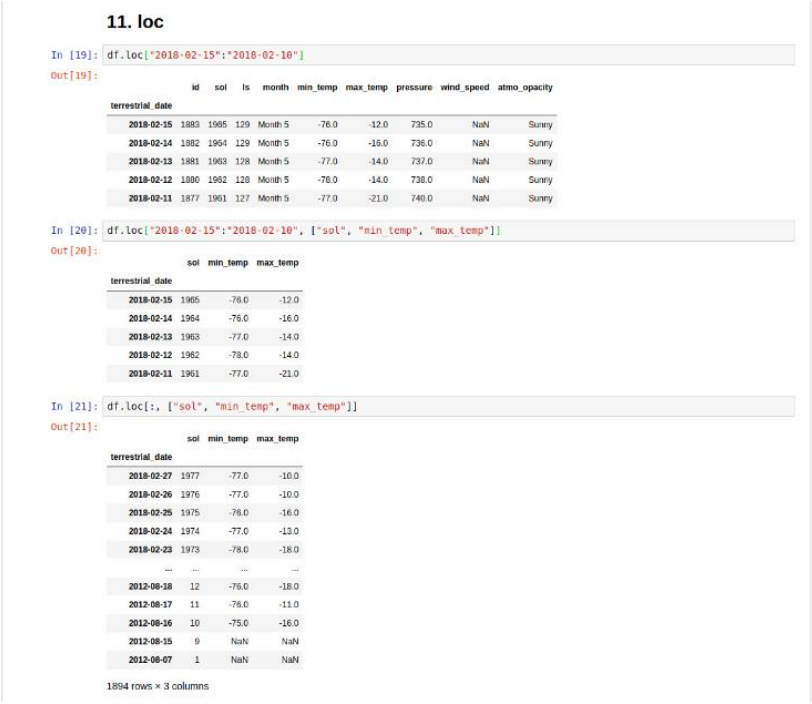
Out[18]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-27	1895	1977	135	Month 5	-77.0	-10.0	727.0	NaN	Sunny
2018-02-26	1893	1976	135	Month 5	-77.0	-10.0	728.0	NaN	Sunny
2018-02-25	1894	1975	134	Month 5	-76.0	-16.0	729.0	NaN	Sunny
2018-02-24	1892	1974	134	Month 5	-77.0	-13.0	729.0	NaN	Sunny
2018-02-23	1889	1973	133	Month 5	-78.0	-18.0	730.0	NaN	Sunny
2018-02-22	1891	1972	133	Month 5	-78.0	-14.0	730.0	NaN	Sunny
2018-02-21	1890	1971	132	Month 5	-78.0	-13.0	731.0	NaN	Sunny
2018-02-20	1888	1970	132	Month 5	-77.0	-16.0	731.0	NaN	Sunny
2018-02-19	1887	1969	131	Month 5	-76.0	-16.0	732.0	NaN	Sunny
2018-02-18	1886	1968	131	Month 5	-76.0	-19.0	732.0	NaN	Sunny
2018-02-17	1885	1967	130	Month 5	-76.0	-15.0	734.0	NaN	Sunny
2018-02-16	1884	1966	130	Month 5	-77.0	-18.0	735.0	NaN	Sunny

LOC

เลือก row ที่ต้องการหรือช่วงของ row ที่ต้องการตาม index

หากไม่ต้องการทุก column สามารถกำหนด column ที่ต้องการได้ด้วยชื่อ column



11. loc

```
In [19]: df.loc["2018-02-15":"2018-02-10"]
```

Out[19]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-15	1883	1965	129	Month 5	-76.0	-12.0	735.0	NaN	Sunny
2018-02-14	1882	1964	129	Month 5	-76.0	-16.0	736.0	NaN	Sunny
2018-02-13	1881	1963	128	Month 5	-77.0	-14.0	737.0	NaN	Sunny
2018-02-12	1880	1962	128	Month 5	-78.0	-14.0	738.0	NaN	Sunny
2018-02-11	1877	1961	127	Month 5	-77.0	-21.0	740.0	NaN	Sunny

```
In [20]: df.loc["2018-02-15":"2018-02-10", ["sol", "min_temp", "max_temp"]]
```

Out[20]:

terrestrial_date	sol	min_temp	max_temp
2018-02-15	1965	-76.0	-12.0
2018-02-14	1964	-76.0	-16.0
2018-02-13	1963	-77.0	-14.0
2018-02-12	1962	-78.0	-14.0
2018-02-11	1961	-77.0	-21.0

```
In [21]: df.loc[:, ["sol", "min_temp", "max_temp"]]
```

Out[21]:

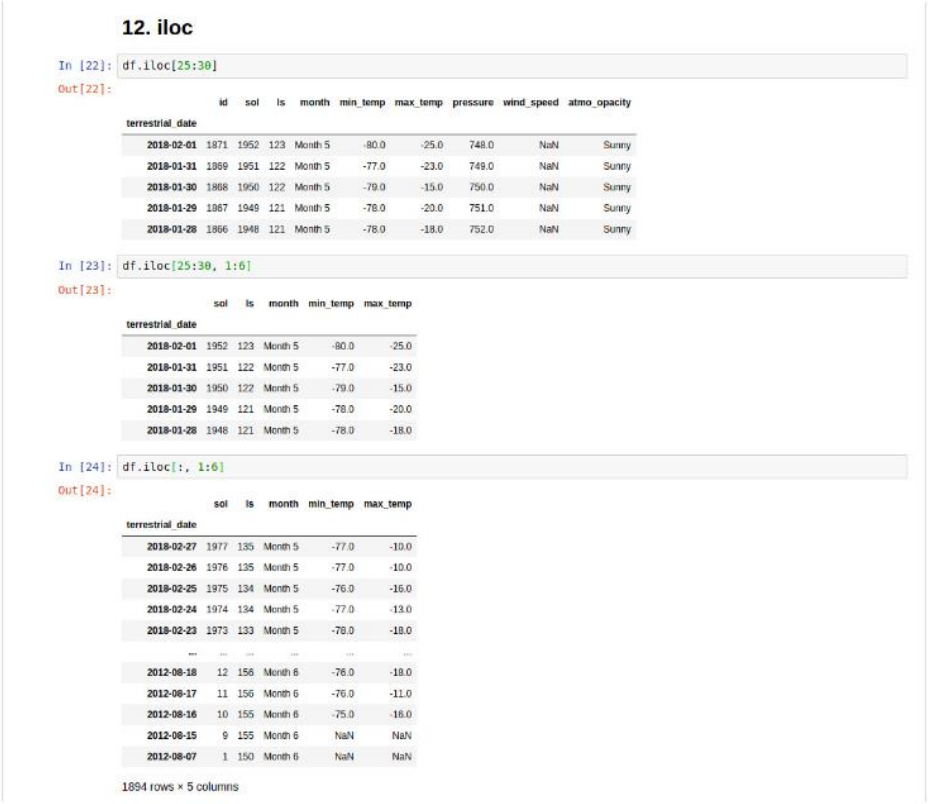
terrestrial_date	sol	min_temp	max_temp
2018-02-27	1977	-77.0	-10.0
2018-02-26	1976	-77.0	-10.0
2018-02-25	1975	-76.0	-16.0
2018-02-24	1974	-77.0	-13.0
2018-02-23	1973	-78.0	-18.0
...	...	...	...
2012-08-18	12	-76.0	-18.0
2012-08-17	11	-78.0	-11.0
2012-08-16	10	-75.0	-16.0
2012-08-15	9	NaN	NaN
2012-08-07	1	NaN	NaN

1894 rows × 3 columns

ILOC

เลือก row ที่ต้องการหรือช่วงของ row ที่ต้องการตามเลขลำดับบรรทัด

หากไม่ต้องการทุก column สามารถกำหนด column ที่ต้องการได้ด้วยเลขลำดับ column



12. iloc

```
In [22]: df.iloc[25:30]
```

Out[22]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2018-02-01	1871	1952	123	Month 5	-80.0	-25.0	748.0	NaN	Sunny
2018-01-31	1869	1951	122	Month 5	-77.0	-23.0	749.0	NaN	Sunny
2018-01-30	1868	1950	122	Month 5	-79.0	-15.0	750.0	NaN	Sunny
2018-01-29	1867	1949	121	Month 5	-78.0	-20.0	751.0	NaN	Sunny
2018-01-28	1866	1948	121	Month 5	-78.0	-18.0	752.0	NaN	Sunny

```
In [23]: df.iloc[25:30, 1:6]
```

Out[23]:

terrestrial_date	sol	ls	month	min_temp	max_temp
2018-02-01	1952	123	Month 5	-80.0	-25.0
2018-01-31	1951	122	Month 5	-77.0	-23.0
2018-01-30	1950	122	Month 5	-79.0	-15.0
2018-01-29	1949	121	Month 5	-78.0	-20.0
2018-01-28	1948	121	Month 5	-78.0	-18.0

```
In [24]: df.iloc[:, 1:6]
```

Out[24]:

terrestrial_date	sol	ls	month	min_temp	max_temp
2018-02-27	1977	135	Month 5	-77.0	-10.0
2018-02-26	1976	135	Month 5	-77.0	-10.0
2018-02-25	1975	134	Month 5	-76.0	-16.0
2018-02-24	1974	134	Month 5	-77.0	-13.0
2018-02-23	1973	133	Month 5	-78.0	-18.0
...	...	...	...	...	...
2012-08-18	12	156	Month 6	-76.0	-18.0
2012-08-17	11	156	Month 6	-76.0	-11.0
2012-08-16	10	155	Month 6	-75.0	-16.0
2012-08-15	9	155	Month 6	NaN	NaN
2012-08-07	1	150	Month 6	NaN	NaN

1894 rows × 5 columns

df[boolean expression]

เลือก row ที่ต้องการตามเงื่อนไขใน boolean expression

13. df[boolean expression]

```
In [25]: df[df.max_temp > 0]
```

Out[25]:

terrestrial_date	id	sol	ls	month	min_temp	max_temp	pressure	wind_speed	atmo_opacity
2017-01-09	1495	1574	295	Month 10	-72.0	4.0	868.0	NaN	Sunny
2016-12-10	1485	1545	277	Month 10	-73.0	3.0	891.0	NaN	Sunny
2016-12-09	1483	1544	276	Month 10	-71.0	1.0	894.0	NaN	Sunny
2016-11-24	1451	1529	267	Month 9	-73.0	3.0	900.0	NaN	Sunny
2016-11-13	1440	1519	260	Month 9	-72.0	5.0	906.0	NaN	Sunny
...	...	...	...	...	...	...	...	...	...
2013-01-09	60	152	241	Month 9	-63.0	1.0	914.0	NaN	Sunny
2013-01-01	51	144	235	Month 8	-64.0	2.0	907.0	NaN	Sunny
2012-12-20	39	133	228	Month 8	-65.0	1.0	891.0	NaN	Sunny
2012-11-12	239	96	205	Month 7	-71.0	2.0	826.0	NaN	Sunny
2012-09-10	176	34	168	Month 6	-73.0	1.0	748.0	NaN	Sunny

168 rows × 9 columns

mean, std, median, max, min

คำนวณค่าทางสถิติของแต่ละ column ด้วยคำสั่งต่างๆ เช่น

mean - ค่าเฉลี่ย

std - ค่า standard deviation

median - ค่ามัธยฐาน

min - ค่าต่ำสุด

max - ค่าสูงสุด

14. mean, median, std, min, max

```
In [26]: df.mean()
Out[26]: id             948.372220
         sol           1007.930306
         ls             169.180570
         min_temp       -76.121050
         max_temp       -12.510445
         pressure       841.066417
         wind_speed            NaN
         dtype: float64

In [27]: df.std()
Out[27]: id             547.088173
         sol           567.079561
         ls            105.738532
         min_temp        5.504098
         max_temp       10.699454
         pressure       54.253226
         wind_speed           NaN
         dtype: float64

In [28]: df.min()
Out[28]: id                    1
         sol                   1
         ls                    0
         month           Month 1
         min_temp            -90
         max_temp            -35
         pressure            727
         wind_speed         None
         atmo_opacity         --
         dtype: object

In [29]: df.max()
Out[29]: id                 1895
         sol                1977
         ls                  359
         month           Month 9
         min_temp            -62
         max_temp             11
         pressure            925
         wind_speed         None
         atmo_opacity      Sunny
         dtype: object
```

Groupby

จัดกลุ่มข้อมูลตาม column ที่กำหนด

15. groupby

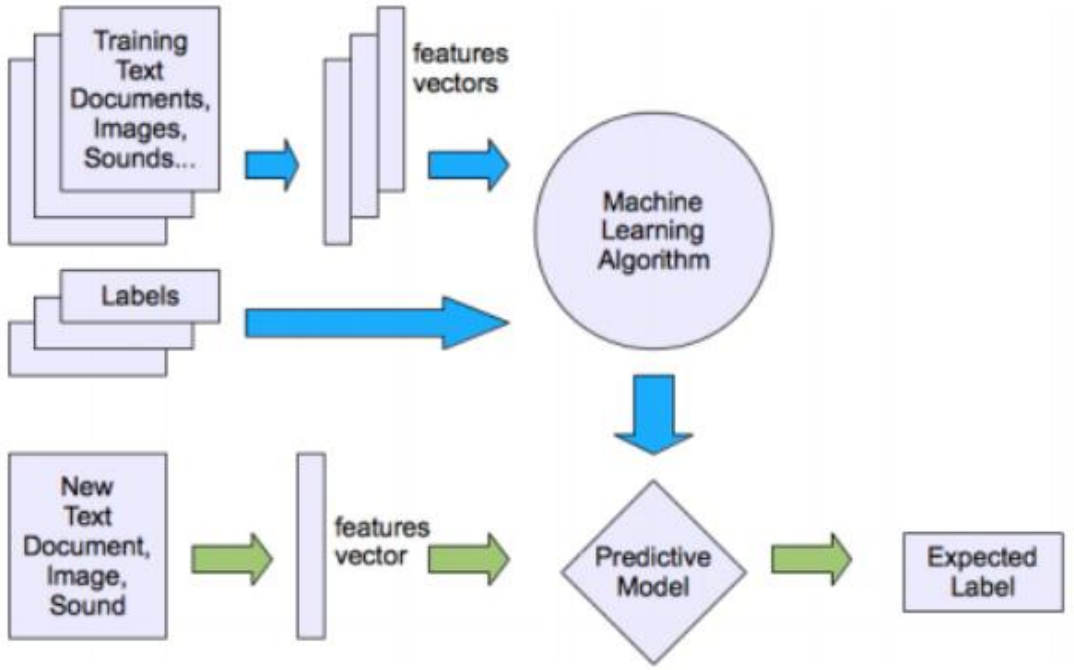
```
In [30]: df.groupby("month").median()
```

Out[30]:

month	id	sol	ls	min_temp	max_temp	pressure	wind_speed
Month 1	983.5	1052.5	15.5	-77.0	-15.0	862.0	NaN
Month 10	838.5	902.5	287.5	-72.0	-4.0	886.5	NaN
Month 11	874.0	939.0	315.0	-72.0	-4.0	856.0	NaN
Month 12	923.5	990.5	345.0	-74.0	-8.0	843.0	NaN
Month 2	1050.5	1119.5	44.0	-80.0	-24.0	890.0	NaN
Month 3	1112.5	1181.5	75.0	-84.0	-28.0	878.0	NaN
Month 4	1175.5	1244.5	104.0	-83.0	-26.0	806.0	NaN
Month 5	1223.0	1294.0	132.0	-79.0	-16.0	747.0	NaN
Month 6	641.0	698.0	165.0	-75.0	-7.0	743.0	NaN
Month 7	686.5	743.5	195.0	-72.0	0.0	793.0	NaN
Month 8	738.0	795.0	224.0	-68.0	-1.0	877.0	NaN
Month 9	785.5	842.5	254.0	-69.0	-1.5	914.0	NaN

Supervised Learning

[caption id="" align="aligncenter" width="519"]



แสดงแผนผังการเรียนรู้แบบมีผู้สอน Supervised Learning

Classification- K Nearest Neighbors

K nearest neighbors (KNN)

เป็นหนึ่งในกระบวนการเรียนรู้ที่ง่ายที่สุด : รับค่าใหม่ เป็นข้อมูลที่ไม่รู้จัก โดยค้นหาในฐานข้อมูลที่อ้างอิงสิ่งที่มีคุณลักษณะใกล้เคียง และมอบหมายให้ predominant class

```
from sklearn import neighbors, datasets
iris = datasets.load_iris()
X, y = iris.data, iris.target # จำนวนตัวอย่าง,จำนวนคุณลักษณะ
knn = neighbors.KNeighborsClassifier(n_neighbors=1)
knn.fit(X, y)
# ดอกไม้ iris อะไร ที่มีขนาด 3cm x 5cm และมีขนาดกลีบเลี้ยง 4cm x 2cm
print(iris.target_names[knn.predict([[3, 5, 4, 2]])])
```

ผลลัพธ์

```
['virginica']
```

```
from sklearn.externals import joblib
joblib.dump(knn, 'filename.pkl')
```

ดึงข้อมูลจาก Classification ที่บันทึกไว้กลับมาใช้งานต่อ

```
knn = joblib.load('filename.pkl')
```

Unsupervised learning

เป็นการเรียนรู้ที่ไม่ใช้ labels ใด ๆ ใช้เพียงข้อมูลที่ได้รับ

Dimensionality reduction กับ PCA (Principal Components Analysis)

สรุปย่อ ๆ PCA คือ การหาแกนที่สำคัญเมื่ออ้างอิงจากข้อมูล

> เป็นส่วนหนึ่งของ factor analysis แนะนำให้ศึกษา factor analysis ก่อน

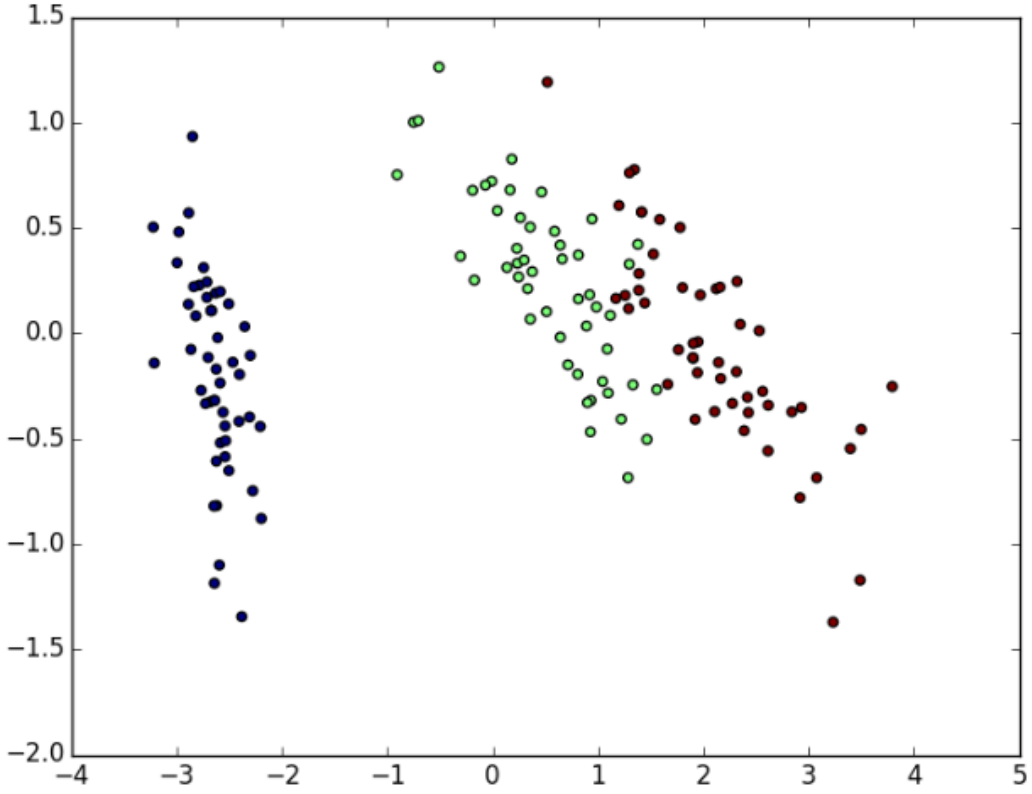
เริ่มต้น Dimensionality reduction กับ PCA

```
from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca.fit(iris.data)
X = pca.transform(iris.data)
```

นำมาแสดงเป็นแผนภาพ

```
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=iris.target)
plt.show()
```

ผลลัพธ์



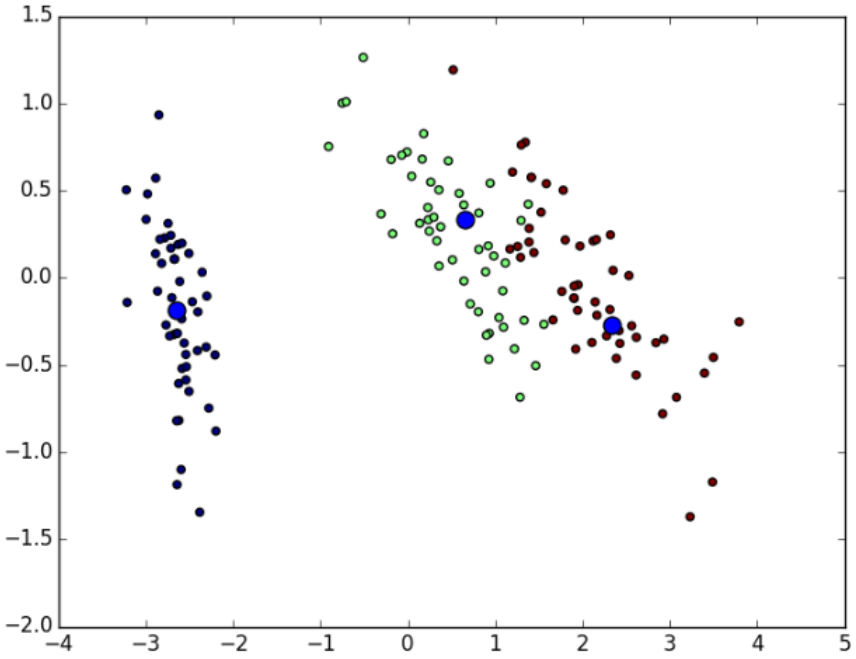
สีน้ำเงิน คือ Setosa สีเขียว คือ Versicolour สีแดง คือ Virginica

Clustering

การจัดกลุ่ม (Clustering) เป็นวิธีการหนึ่งที่แบ่งข้อมูลออกเป็นกลุ่ม (cluster) โดยพิจารณาให้แต่ละแถวเสมือนเป็นวัตถุ แล้วจับกลุ่มให้กับวัตถุที่มีความคล้ายคลึงกันอยู่ใน cluster เดียวกัน วัตถุที่อยู่ต่าง cluster จะมีความคล้ายคลึงกันน้อยที่สุด

```
from sklearn.cluster import KMeans
km = KMeans(3)
km.fit(X)
print(km.cluster_centers_)
plt.scatter(X[:, 0], X[:, 1], c=iris.target)
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], marker='o',
s=100)
plt.show()
```

ผลลัพธ์



จุดสีน้ำเงินใหญ่คือ cluster ที่ถูกแบ่ง