รายงาน

เรื่อง Machine Learning and Python Data Science Tools

กลุ่ม Superman

จัดทำโดย

นางสาวณัฐพร เหลือนับ 6330300313

นางสาววรพร เย็นธงชัย 6330300739

นางสาวสวรรยา ศิริกิจ 6330300933

นางสาวสุวิชาดา พันธ์สุข 6330300992

เสนอ

ผศ.ดร.กุลวดี สมบูรณ์วิวัฒน์

รายงานนี้เป็นส่วนหนึ่งของรายวิชา Introduction to Data Science

คณะวิศวกรรมศาสตร์ศรีราชา สาขาคอมพิวเตอร์และสารสนเทศศาสตร์

ภาคเรียนที่ 1 ปีการศึกษา 2565

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Data Science (03603351) โดยมีจุดประสงค์เพื่อศึกษาความรู้ที่ได้จากเรื่อง Machine Learning and Python Data Science Tools ซึ่งรายงานนี้มีเนื้อหาเกี่ยวกับความรู้จากการศึกษา Machine Learning and Python Data Science

ผู้จัดทำได้เลือกหัวข้อนี้ในการทำรายงานเนื่องมาจากเป็นเรื่องที่น่าสนใจและต้องขอขอบคุณสมาชิกในกลุ่มทุกคนที่ให้ความช่วยเหลือมาโดยตลอดผู้จัดทำหวังว่ารายงานฉบับนี้จะให้ความรู้ และเป็นประโยชน์แก่ผู้อ่านทุก ๆ ท่าน

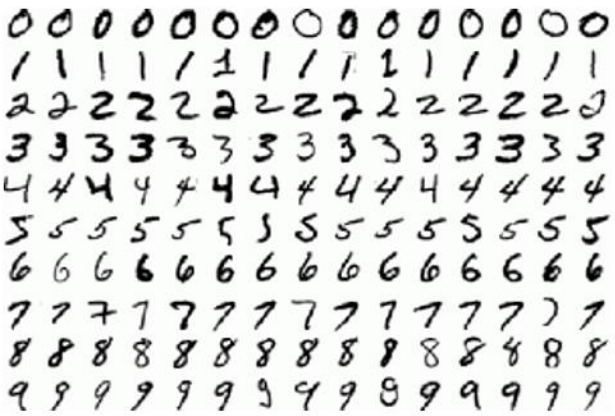
คณะผู้จัดทำ

สารบัญ

Machine learning บน MNIST dataset

MNIST Data เป็น Dataset ยอดฮิตอีกตัวใช้สำหรับเรียนรู้การทำ Machine Learningที่เกี่ยวกับรูปภาพปูทางไปสู่ CNN (Convolutional Neural Networks) ต่อไป
สำหรับในตอนนี้จะลองนำข้อมูลนี้มาใช้ทดสอบการแยกแยะตัวเลขดู โดยใช้วิธีที่พื้นฐานที่สุด นั่นคือการถดถอยโลจิสติกแบบมัลติโนเมียล (การถดถอยซอฟต์แม็กซ์)

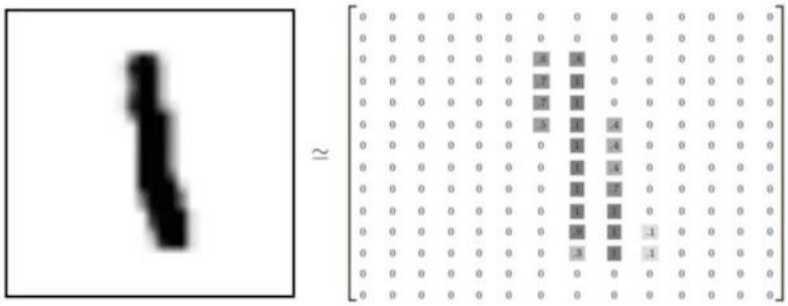
มารู้จัก MNIST DataSet กันก่อน ถ้าเราไม่รู้จักข้อมูลเราจะแก้ไขปัญหาไม่ได้แน่นอน ข้อมูล MNIST เป็นข้อมูลรูปภาพตัวเลข 0–9 ที่เขียนด้วยมือ เพื่อใช้ทำการฝึกทำนายว่ารูปตัวเลขดังกล่าวเป็นเลขอะไร



รูปที่ 1

ด้วยความที่เข้าใจได้ทั่วโลก TensorFlow เลือก Data Set ชุดนี้มาไว้ใน API เลย โดยมีจำนวนรูปสำหรับ Training 55,000 รูป รูปสำหรับ Test 10,000 รูป รูปสำหรับ Validation 5,000 รูป

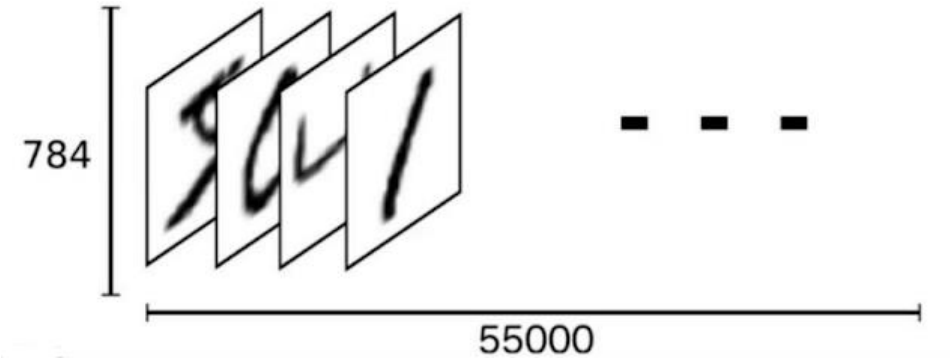
รูปตัวเลขแต่ละรูปจะถูกนำมาแปลงเป็น 2D array ของ pixels ขนาด 28x28 และแทนด้วยตัวเลขตั้งแต่ 0–1 โดยสีขาวจะแทนด้วยเลข 0 และ สีดำแทนด้วยเลข 1 ส่วนในขั้นตอนการนำไปใช้ เราจะทำการ reshape array มาเป็น 28x28



รูปที่ 2

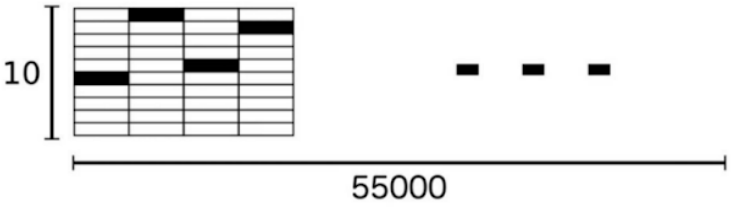
จาก 1 รูป 28x28 Data set ทำการ แปลงให้เป็น 784x1 หรือ 1*784 แล้วแต่สะดวก ที่มาของเลข 784 มาจาก 28x28 = 784

เพราะต้องการที่จะจัดเรียงชุดข้อมูลให้อยู่ในรูปแบบที่สามารถอ้างอิงทีละรูปได้สะดวก 55,000 รูป จะกลายเป็น Array ขนาด (784,55000) ดังนั้นพอนึกภาพการ array ที่จะเอาไปใช้เทรนจะได้ตามรูปข้างล่างนี้ ถ้าอยากหยิบรูปลำดับที่ 0 มาใช้ก็แค่อ้างอิง array[0]



รูปที่ 3

มาดูการจัดการ Array ของ Label ที่ใช้เก็บรูปตัวเลขนี้กันบ้าง สมมุติถ้าเป็นเลข 4 เราจะ label มันด้วย array : [0,0,0,0,1,0,0,0,0,0] จะเห็นว่าเลข 1 อยู่ในตำแหน่งที่ 5 เพราะเราเริ่มต้นที่เลข 0 ไล่ไปจนถึง 9 นั่นเอง ดังนั้นสุดท้ายแล้ว lable ของข้อมูล Test ชุดนี้จะแทนด้วย array ขนาด (10,55000) ถ้าจะให้เห็นภาพก็ตามด้านล่างนี้เลย



รูปที่ 4

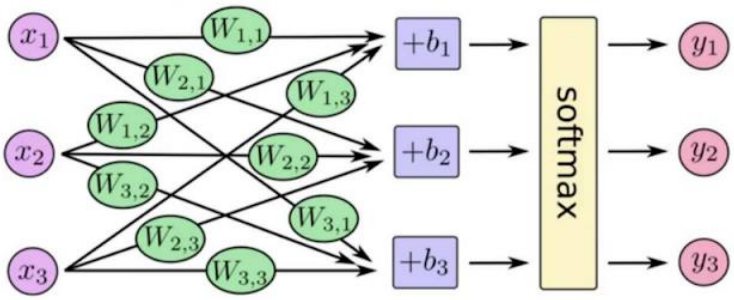
เราจะใช้ Softmax มาเป็น Activate Function เพราะ Function นี้จะให้ค่าออกมาตั้งแต่ 0 ถึง 1

$$z_i = \sum_j W_{i,j} x_j + b_i$$

$$y = \text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

สมการที่ 1

Network ที่เรากำลังจะทำ หน้าตาจะคล้ายๆด้านล่าง



รูปที่ 5

จากรูป Network ได้ความว่า

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{matrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{matrix} \right)$$

รูปที่ 6

จัดรูปแบบ Vectorize ซะใหม่ ให้ W และ x ในรูป vector การคูณ แล้วค่อยบวกด้วย vector b กลายร่างเป็น

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

รูปที่ 7

ร่างสุดท้ายที่จะเอาไปใช้ใน TensorFlow

$$y = \text{softmax}(Wx + b)$$

สมการที่ 2

หลักการของอัลกอริทึม k-Means และ SVM

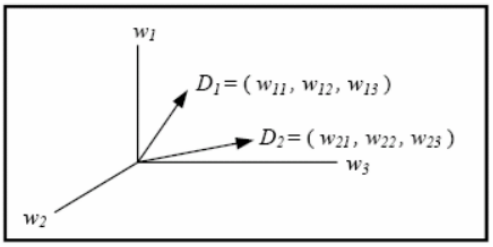
K-means คือ วิธีการหนึ่งใน Data mining อยู่ในกลุ่มของ Unsupervised Learning หรือแปลตรงๆคือการเรียนรู้แบบไม่ต้องสอน (Supervised Learning ต้องสอนก่อนต้องจับ Train และต้อง Test เป็นต้น) โดยหน้าที่หลักของ K-means คือการแบ่งกลุ่มแบบ Clustering ซึ่งการแบ่งกลุ่มในลักษณะนี้จะใช้พื้นฐานทางสถิติ ซึ่งแน่นอนว่าต้องมีตัวเลขประกอบ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

K-mean จะง่ายและทำงานในเวลาเชิงเส้นแต่เมื่อใช้ในการ แก้ปัญหาเวกเตอร์แบบหลายมิติขนาดใหญ่ก็จะใช่การทำงานที่มากและซับซ้อน มีการพัฒนาให้ใช้กับข้อมูลปริมาณมากได้ รวมทั้งงานวิจัยของ Sanpawat มีการนำเสนอวิธีการปรับปรุงอัลกอริทึม เคมีนโดยการนำการคำนวณแบบขนานเข้ามาร่วมด้วย ซึ่งสามารถทำงานในเครื่องหลายๆเครื่องที่มีหน่วยความจำแบบสะสมได้ขนาดใหญ่

งานวิจัยของ ฆนาศัย กรึงไกร และ ชุลีรัตน์จรัสกุลชัย ได้ทำการวิจัยเรื่อง การจัดกลุ่มเอกสารข้อความภาษาไทยด้วยขั้นตอนวิธี Spherical K-Means แบบขนานบนพิรุณลีนุกซ์คลัสเตอร์ โดยพยายามคิดค้นวีธีที่จะจัดกลุ่มเอกสารปริมาณมากๆแบบอัตโนมัติซึ่งจัดกลุ่มแบบโกลบอล(global) เอกสารถูกจัดกลุ่มตามที่ปรากฏอยู่ในชุดเอกสารทั้งหมดจุดประสงค์เพื่อลดเวลาการประมวลผลกับชุดเอกสารข้อความเต็ม(full text) จำนวนมากๆ แนวคิดคือให้เอกสารทั้งหมดถูกแบงออกไปประมวลผลเป็นชุดย่อยๆ และสามารถถูกจัดกลุ่มในเวลาเดียวกันได้อย่างอิสระ หลังจากนั้นแยกรายงานออกเป็นกลุ่มตามเนื้อหาที่คล้ายคลึงกัน ในงานวิจัยนี้มีการแทนเอกสารที่ไม่มีโครงสร้าง(unstructured text document) ด้วย Vector Space Model(VSM) ใน VSM เอกสารแต่ละฉบับ เปรียบเสมือนกับเวกเตอร์คำ โดยที่ ขนาดของเวกเตอร์ขึ้นอยู่กับจำนวนของคำที่ปรากฏอยู่ในเอกสารฉบับนั้น กำหนดให้ wik คือค่าน้ำหนักของคำ k ที่ปรากฏในเอกสาร i

เวกเตอร์สำหรับเอกสาร Di สามารถเขียนแทนด้วย

Di = (wi1, wi2, ..., wit) ซึ่ง คือจำนวนของคำที่ไม่ซ้ำกันในชุดเอกสารทั้งหมด



รูปที่ 8

โดยคุณสมบัติของเวคเตอร์ทำให้สามารถคำนวณค่าความคล้ายคลึงของ เอกสารคู่หนึ่งๆได้จากค่าสัมประสิทธิ์ cosine ของมุมระหว่างคู่ของเวคเตอร์ซึ่งจะมีค่าอยู่ ระหว่าง 0 ถึง 1 มีสูตรดังนี้

$$\text{sim}(D_i, D_j) = \frac{D_i \bullet D_j}{\|D_i\| \times \|D_j\|}$$

สมการที่ 3

ต่อจากนั้นนำมาให้น้ำหนักของคำวิธีการให้น้ำหนักคำที่ใช้กันอย่างมากในการสืบค้นข้อมูลคือ tf * idf (term frequency * inverse document frequency) โดยที่ค่า idf คำนวณจากค่า log(N/df) ซึ่ง N คือจำนวน เอกสารในชุดเอกสารทั้งหมด และ df คือจำนวนเอกสารที่มีคำนั้นปรากฏอยู่ วิธีการให้น้ำหนักของคำมีการ normalization ทำให้เวคเตอร์มีขนาด 1 หน่วยมีสูตรดังนี้

$$w_{ik} = \frac{tf_{ik} \times \log(N / df_k)}{\sqrt{\sum_{j=1}^{t} (tf_{ij})^2 \times (\log(N / df_j))^2}}$$

สมการที่ 4

โดยที่ tf_{ik} คือความถี่ของคำ k ในเอกสาร i

N คือจำนวนของเอกสารในชุดเอกสาร

df_k คือจำนวนของเอกสารในชุดเอกสารซึ่งบรรจุคำ i

สามารถหาค่าความคล้ายคลึงกันของคู่เอกสารได้จาก inner product เนื่องจาก $\|D_i\| = \|D_j\| = 1$ ดังนั้น

$$\text{sim}(D_i, D_j) = D_i \bullet D_j = \sum_{k=1}^{t} (w_{i,k} \times w_{j,k})$$

สมการที่ 5

k-mean ไม่เหมาะกับข้อมูลที่มีความสัมพันธ์กัน(correlation) เนื่องจากข้อมูลมีโอกาสเป็นสมาชิกเพียงกลุ่มใดกลุ่มหนึ่งเท่านั้น การจัดกลุ่มแบบฟัซซี่นั้น สมาชิกของกลุ่มมีโอกาสหรือค่าการเป็นสมาชิกของข้อมูลระดับต่างๆในทุกๆกลุ่ม สำหรับการแบ่งกลุ่มแบบฟัซซี่ (fuzzy clustering) Dunn ได้มีการปรับปรุงโดย Bezdek

Value Stream Mapping เป็นเครื่องมือและเทคนิคที่สนับสนุนการพัฒนากลยุทธ์การผลิตแบบลีน (Lean Manufacturing Strategy) ด้วยการแสดงลำดับขั้นตอนของกิจกรรมต่างๆ ที่มุ่งส่งมอบคุณค่าให้กับลูกค้า โดยแนวคิด การสร้างสายธารแห่งคุณค่า Value Stream Mapping
จะทำให้สามารถเข้าใจภาพรวมของกระบวนการ (Overall Process)
จากมุมมองลูกค้าโดยมุ่งแนวทางปรับปรุงการไหลของทรัพยากรและสารสนเทศ การสร้างสายธารแห่งคุณค่า VSM จึงเป็น
แนวทางที่ใช้จำแนกกิจกรรมที่สร้างคุณค่าเพิ่มและกิจกรรมที่เกิดความสูญเปล่าโดยนำข้อมูลผลลัพธ์จากการวิเคราะห์สถานะปัจจุบัน (Current State) ที่ถูกแสดงด้วยเอกสารสำหรับกำหนดสถานะในอนาคต (Future State) หลังจากการปรับปรุง

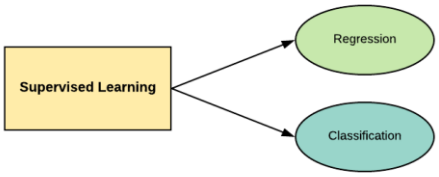
Supervised Learning

Supervised Learning หรือการเรียนรู้แบบมีผู้สอน เป็นศาสตร์แขนงหนึ่งใน AI ภายใต้หัวข้อ Machine Learning ที่กำลังเป็นที่นิยมศึกษาและวิจัยกันในปัจจุบันซึ่งทำได้ง่ายและต้นทุนต่ำ

Supervised Learning

เป็นการสร้าง model จากชุดข้อมูลสำหรับฝึกฝน (training dataset) ซึ่งประกอบไปด้วย ตารางข้อมูลทดสอบ (feature matrix) และเป้าหมาย (target vector) เมื่อได้ model ที่มีความแม่นยำพอแล้วก็จะสามารถนำ model ไปใช้คำนวณ เป้าหมาย ของข้อมูลที่ผู้ใช้กรอก โดยเป้าหมายที่ทำการคำนวณมี 2 แบบได้แก่

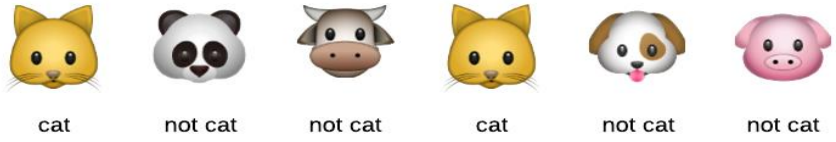
- หมวดหมู่ไหน? **classification**
- **A or B?** - two-class classification
- **A or B or C?** - multi-class classification
- เป็นค่าเท่าใหร่? **regression**
- **How much/many?**



รูปที่ 9 หลักการ Supervised Learning สามารถนำไปประยุกต์ใช้แก้ปัญหาได้ 2 รูปแบบ

Classification

โดยเราจะเปรียบเทียบเหมือนการสอนเด็ก โดยเราจะชี้ภาพสัตว์ให้เด็กที่ไม่เคยเห็น แล้วบอกว่าสัตว์ตัวไหนคือแมว ตัวไหนไม่ใช่แมว ชี้ไป 2–3 วัน และให้เด็กได้เจอสัตว์หลายประเภท จนเริ่มเข้าใจวันที่ 4–5 เราอาจจะลองเอาแมวตัวที่เด็กไม่เคยเห็นมาให้ดูสัก 10 ตัว รวมกับสัตว์อื่นๆโดยคราวนี้เราไม่บอกว่าสัตว์ตัวไหนคือแมว ตัวไหนไม่ใช่แมวถ้าเด็กตอบถูกก็แปลว่าการสอนของเรามีประสิทธิภาพ หากเราสอนเด็กไปเลยว่า สัตว์ที่เด็กเห็นนั้นเป็น แมว หมา หรือหมู เด็กก็อาจจะตอบได้มากกว่าแค่ แมว หรือไม่ใช่แมว วิธีนี้อาจจะต้องใช้กระบวนการสอนที่มีความซับซ้อนมากขึ้นไปอีก ซึ่งเรียกวิธีการสอนเด็กทั้ง 2 แบบนี้ว่า Classification

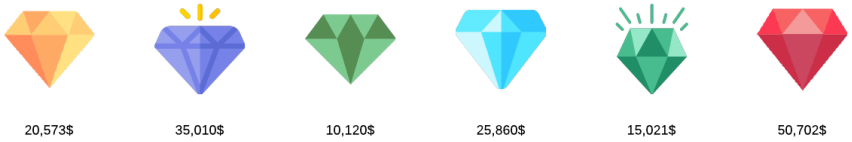


รูปที่ 10 ผลลัพธ์ที่ได้จากการสอนเด็กแบบ Classification ที่ไม่ซับซ้อน

รูปที่ 11 ผลลัพธ์ที่ได้จากการสอนเด็กแบบ Classification ที่ซับซ้อน

Regression

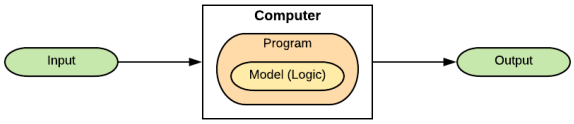
ต่อมาเราเรียกเด็กอีกคนมาสอนเรื่องราคาเพชร (diamond) เราหยิบเพชรอันนึง ขนาด 2 กะรัต สีเหลือง ระดับความสะอาด VS2 แล้วบอกเด็กว่า เนี่ยราคา 2 ล้านบาท หยิบอีกเม็ดขนาด 3 กะรัต สีฟ้า ระดับความสะอาด VS1 แล้วบอกเด็ก 3 ล้านบาท ทำแบบนี้ไปหลายๆ เม็ดจนเด็กเกิด model หรือ logic ในการคาดเดาราคาของเพชรขึ้นในหัว จนวันนึงสุ่มหยิบเพชรเม็ดใหม่ขึ้นมา ก็อาจให้เด็กคาดเดาราคาได้เลย เราเรียกกระบวนการสอนเด็กแบบนี้ว่า Regression



รูปที่ 12 ผลลัพธ์ที่ได้จากการสอนเด็กแบบ Regression

การเขียนโปรแกรมแบบดั้งเดิม

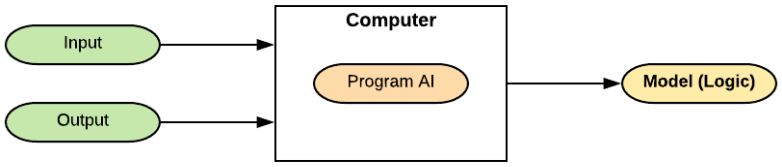
โดยวิธีการทำคือเราจะนำ logic หรือ model ที่เราคิดขึ้นมา ใช้เขียนโปรแกรมเพื่อให้ได้ output จาก input ที่รับเข้ามาหากนำไปเปรียบเทียบกับตัวอย่างการสอนเด็กข้างต้น input ของเราก็คือภาพสัตว์ชนิดต่างๆ ส่วน output ก็คือคำตอบว่าภาพที่รับเข้าไปเป็นภาพสัตว์ชนิดอะไร การเขียนโปรแกรมแบบนี้ยากและแทบเป็นไปไม่ได้เลย เนื่องจากความซับซ้อนของ model หรือ logic ที่เราต้องเป็นคนคิดขึ้นมาใช้ในโปรแกรม เพื่อแยกแยะภาพสัตว์ชนิดต่างๆ



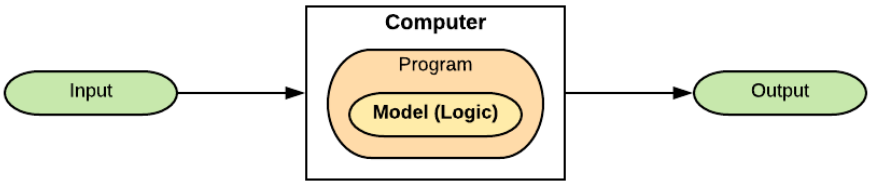
รูปที่ 13 Traditional Programming

การใช้ Supervised Learning

โดยเราจะเขียนโปรแกรมให้คอมพิวเตอร์สร้าง model หรือ logic ของโปรแกรมขึ้นมาเองจาก input (ภาพสัตว์) และ output (คำตอบ) ที่เราต้องการเช่นเดียวกับการสอนเด็กจากนั้นเราจึงนำ model มาใช้ ดังนั้นยิ่งเรามี input และ output ที่มีความหลากหลายและจำนวนมากเท่าไร เราก็มี“โอกาส”ได้ model ที่มีความแม่นยำมากขึ้นกระบวนการสร้าง model แบบนี้เราเรียกว่าการ “**เทรน**” ซึ่งสามารถกินเวลาได้ตั้งแต่หลักวินาทีจนถึงหลายๆวัน แล้วแต่ความซับซ้อนของโจทย์ที่เราต้องการแก้ และพลังในการประมวลผลของเครื่องคอมพิวเตอร์ที่เราใช้เทรน



รูปที่ 14 ภาพแสดงกระบวนการเทรน เพื่อให้ได้ model ที่เราต้องการ



รูปที่ 15 เมื่อเราได้ model ที่เราต้องการแล้ว เราจึงนำมาประยุกต์ใช้ กับโปรแกรมของเรา

จะเห็นว่ามีความซับซ้อนมากกว่าการเขียนโปรแกรมแบบดั้งเดิมแต่ข้อดีของมันก็คือสามารถทำสิ่งที่เป็นไปไม่ได้ให้เป็นไปได้

scikit-learn (supervised) ต่างจาก python list อย่างไร?

Scikit Learn นั้นจะเน้นในส่วนของการสร้างโมเดลเพื่อทำนายและพยากรณ์ต่างๆ สามารถทำ Spam Detection, Image Recognition, Clustering หรือ Regression จุดที่ต้องระวังคือ หากต้องการผลที่ถูกต้องหรือมีประสิทธิภาพสูง จำเป็นจะต้องมี Input Data ที่ดี ฉะนั้น NumPy และ pandas มักเป็น 2 เครื่องมือที่ถูกเลือกใช้ก่อนการป้อนข้อมูลลงใน Model

python list คือ ชุดข้อมูลแบบคอลเล็คชันที่มีการเรียงลำดับ และสามารถแก้ไขได้ โดยการสร้างข้อมูลแบบ List ในภาษาไพธอน จะใช้เครื่องหมาย square brackets [] ครอบชุดข้อมูล

Unsupervised Learning

การวิเคราะห์การจัดกลุ่มข้อมูล (Cluster Analysis) หรือการจัดคลัสเตอร์ เป็นเทคนิคในกลุ่มของการเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning) กล่าวคือ จะทำการจัดกลุ่มข้อมูลที่ไม่เคยมีการจัดกลุ่มมาก่อนหน้า แต่จะแบ่งกลุ่มข้อมูลโดยพิจารณาจากลักษณะที่คล้ายกันของข้อมูล โดยจะนำข้อมูลที่มีลักษณะคล้ายกันมาอยู่กลุ่มเดียวกัน ส่วนข้อมูลที่มีลักษณะต่างออกไปก็ให้ไปอยู่อีกกลุ่มหนึ่ง แนวทางของการนำเทคนิค นี้ไปใช้ คือ จะไม่ใช่การหาผลลัพธ์ที่ต้องการวัดค่าความแม่นยำ หากแต่ต้องการหาความสัมพันธ์ของข้อมูลอีก รูปแบบหนึ่ง เช่น การจัดกลุ่มลูกค้าจากพฤติกรรมการซื้อสินค้า การจัดกลุ่มนักท่องเที่ยวจากพฤติกรรมการไปยัง สถานที่ต่าง ๆ

ขั้นตอนในการสร้างโมเดลประเภทแบบไม่มีผู้สอนนั้นจะมีความต่างจากการสร้างโมเดลแบบมีผู้สอน คือการสร้างโมเดลนั้นจะไม่มีการแบ่งชุดข้อมูลออกเป็นชุดสำหรับฝึกสอนและทดสอบเนื่องจากเป็นชุดข้อมูลที่ไม่มีคลาสคำตอบจึงทำให้ไม่จำเป็นต้องแบ่งชุดข้อมูลออกเป็น2ส่วนเหมือนกับเทคนิคที่ใช้สร้างโมเดลแบบมีผู้สอน (แต่จะแบ่งก็ได้) ซึ่งมีขั้นตอนของการสร้างโมเดลเป็น 4 ขั้นตอน ดังนี้

- **กำหนดวิธีวัดความคล้ายหรือความต่างของข้อมูล** ตัวอย่างวิธีที่นิยมใช้ เช่น ยูคลิเดียน (Euclidean Distance) โคไซน์ (Cosine) และ แมนฮัชตัน (Manhattan Distance) จะขออธิบายวิธีการวัดความคล้ายของข้อมูลด้วยวิธีวัดระยะแบบยูคลิเดียน (Euclidean Distance) ซึ่งสามารถคำนวณหาได้จาก สมการ

$$D = \sqrt{\sum_{i=1}^{d} \left(x_1^i - x_2^i\right)^2}$$

สมการที่ 6

โดยที่ D คือ ระยะห่างระหว่างข้อมูลที่ 1 และ 2 ซึ่งถ้าค่าเท่ากับ 0 แสดงว่าข้อมูลมีความเหมือนกัน

x_1^i คือ ค่าของแอทริบิวต์ที่ i ของข้อมูลที่ 1

x_2^i คือ ค่าของแอทริบิวต์ที่ i ของข้อมูลที่ 2

- **เลือกอัลกอริทึมที่ใช้ในการจัดกลุ่มข้อมูล** ซึ่งหลักๆ แบ่งได้เป็น 2 ประเภท ได้แก่

1. ประเภทที่มีการแบ่งกลุ่มอย่างชัดเจน (Hard clustering)
   เป็นเทคนิคที่มีการแบ่งข้อมูลออกจากกันเป็นกลุ่มๆ อย่างสิ้นเชิง
   โดยแต่ละข้อมูลนั้นจะถูกจัดให้อยู่ในกลุ่มใดกลุ่มหนึ่งเท่านั้น เทคนิคประเภทนี้ ได้แก่
   เทคนิคการจัดกลุ่มข้อมูลแบบเคมีน (K-means clustering algorithm)

2. ประเภทที่มีการแบ่งกลุ่มแบบไม่ชัดเจน (Soft clustering)
   เป็นเทคนิคการแบ่งที่ข้อมูลสามารถอยู่ในหลายๆ กลุ่มได้ โดยขึ้นอยู่กับความน่าจะเป็นของตัวข้อมูล เทคนิคประเภทนี้ ได้แก่ เทคนิคการจัดกลุ่มแบ่งกลุ่มข้อมูลแบบลำดับชั้น (Hierarchical clustering methods)

- **กำหนดจำนวนกลุ่มที่ต้องการ** ซึ่งในอัลกอริทึมประเภทที่มีการแบ่งกลุ่มอย่างชัดเจน เราจำเป็นต้องกำหนดจำนวนกลุ่มที่ต้องการแต่ถ้าเป็นอัลกอริทึมประเภทที่มีการแบ่งกลุ่มแบบไม่ชัดเจนเราไม่จำต้องกำหนดจำนวนกลุ่มก็ได้อัลกอริทึม จะทำการหาจำนวนกลุ่มที่เหมาะสมให้กับชุดข้อมูลเอง
- **ประเมินผลการวิเคราะห์การจัดกลุ่ม**
  เนื่องจากเทคนิคการวิเคราะห์จัดกลุ่มเป็นประเภทโมเดลแบบไม่มีผู้สอนซึ่งจะมีวิธีการวัดผลที่แตกต่างไปจากเทคนิคประเภทแบบมีผู้สอน คือ จะไม่สามารถวัดผลได้จากเปรียบเทียบหาความแม่นยำ เนื่องจากไม่มีผลลัพธ์ตั้งต้นให้เปรียบเทียบ แต่จะสามารถวัดผลได้ด้วยวิธีอื่น เช่น วัดจากความพึงใจในตัวโมเดลที่ได้ หรือวัดผลด้วยวิธีอื่นๆ เช่น การใช้วิธีการวิเคราะห์จัดกลุ่มข้อมูลเพื่อระบุตำแหน่งในการสร้างศูนย์กระจายสินค้าโดยต้องการให้ได้ศูนย์กระจายสินค้าที่สามารถลดค่าใช้จ่ายในการขนส่งได้มากที่สุดจากตัวอย่างวิธีการวัดผลสามารถทำได้ด้วยการนำโมเดลการจัดกลุ่มที่ได้ ไปสร้างโมเดลในการหาค่าที่เหมาะสมอีกครั้ง (Optimization Model)
  เพื่อหาจุดที่ให้ค่าขนส่งโดยรวมน้อยที่สุด

อธิบายการทำงานของอัลกอริทึมที่เป็นที่รู้จักกันดีและถูกใช้บ่อยในงานการจัดกลุ่มข้อมูล นั่นคือ การจัดกลุ่มข้อมูลแบบเคมีน

การจัดกลุ่มข้อมูลแบบเคมีน (K-mean clustering Algorithm)

การจัดกลุ่มข้อมูลแบบเคมีน (K-mean clustering) นับเป็นวิธีการที่ถูกนำใช้บ่อยมากที่สุดเนื่องจากมีขั้นตอนการทำงานที่ไม่ซับซ้อน และเข้าใจได้ง่าย โดยมีขั้นตอนการทำงานดังนี้

ขั้นตอนที่ 1 เริ่มต้นจากการกำหนดค่า K หรือจำนวนกลุ่มข้อมูลที่ต้องการ

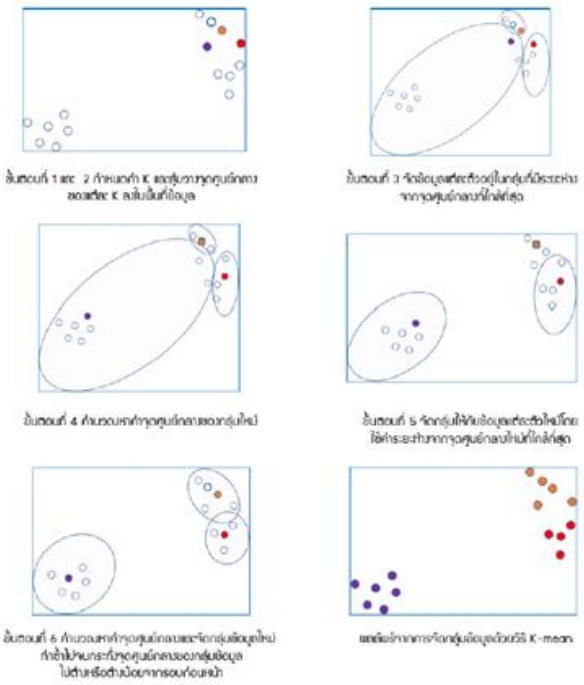
ขั้นตอนที่ 2 สุ่มวางตำแหน่งจุดศูนย์กลางของข้อมูลแต่ละกลุ่ม หรือเซ็นทรอยด์ (Centroid)

ขั้นตอนที่ 3 จากจุดศูนย์กลางแต่ละจุดจะมีการคำนวณระยะทางกับทุกข้อมูลในชุดข้อมูล
ซึ่งคำนวณโดยวิธียูคลิเดียนดังได้อธิบายไว้แล้วในข้างต้น
จากนั้นแต่ละข้อมูลจะถูกจัดอยู่ในกลุ่มของจุดศูนย์กลางที่มีระยะทางใกล้ที่สุดเท่านั้น

ขั้นตอนที่ 4 หลังจากทำการจัดกลุ่มข้อมูลใหม่แล้วทำการคำนวณค่าเฉลี่ยของสมาชิกในกลุ่ม
เพื่อกำหนดเป็นจุดศูนย์กลางของกลุ่มข้อมูลใหม่

ขั้นตอนที่ 5 ทำซ้ำข้อ 3 ถึง 4

จนกระทั่งค่าจุดศูนย์กลางของกลุ่มข้อมูลใหม่ได้ค่าไม่ต่างหรือต่างเพียงเล็กน้อยจากค่าจุดศูนย์กลางรอบก่อนหน้าดังรูปที่ 1



รูปที่ 16 ขั้นตอนการจัดกลุ่มข้อมูลด้วยวิธี K-means

การจัดกลุ่มแบบโครงสร้างลำดับชั้น (Hierarchical clustering)

เป็นการจัดกลุ่มโดยไม่ต้องมีการกำหนดจำนวนกลุ่มที่ต้องการจัดกลุ่มข้อมูลก่อน เป็นการวิเคราะห์แบบเป็นขั้นตอน วิธีการที่นิยมคือวิธีการ “Agglomerative Hierarchical Cluster” การจัดกลุ่มแบบโครงสร้างลำดับชั้นนั้นจะประกอบด้วย 4 ขั้นตอนดังต่อไปนี้

อาจารย์ต้องการแบ่งกลุ่มนักเรียนในห้องโดยพิจารณาจากคะแนนของนักเรียนแต่ละคน โดยผลคะแนน

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

รูปที่ 17 คะแนนของนักเรียนในชั้นเรียนเพื่อพิจารณาแบ่งกลุ่ม

ขั้นตอนที่ 1 กำหนดให้ข้อมูลแต่ละข้อมูลคือแต่ละคลัสเตอร์ ดังนั้นจากรูปที่ 17 เรามีนักเรียน 5 คนจึงทำให้เรามีคลัสเตอร์ทั้งหมด 5 คลัสเตอร์

รูปที่ 18 จำนวนคลัสเตอร์เริ่มต้น

ขั้นตอนที่ 2 ทำการสร้างเมตริกซ์ขนาด $N \times N$ จากชุดข้อมูล คำนวณระยะห่างของแต่ละจุดข้อมูล โดยใช้สูตรเดียวกับที่ใช้คำนวณใน K-Mean แสดงดังรูปที่ 19 เช่น ระยะห่างระหว่าง (1, 2) = $\sqrt{10-7^2} = \sqrt{9} = 3$

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

รูปที่ 19 เมตริกซ์ขนาด$N \times N$ จากชุดข้อมูล

ขั้นตอนที่ 3 มองหาค่าระยะห่างระหว่างข้อมูลที่น้อยที่สุดและทำการรวมข้อมูลเหล่านั้นเป็นคลัสเตอร์ เดียวกัน และทำการปรับปรุงเมตริกซ์ จากรูปที่ 19 ค่าที่น้อยที่สุดคือ (1, 2) = 3 แสดงรูปที่20
และเมื่อทำการรวมคลัสเตอร์ (1,2) แสดงดังรูป

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

รูปที่ 20 ค่าที่น้อยที่สุดระหว่าง (1, 2)

รูปที่ 21 ทำการรวมคลัสเตอร์ (1,2)

ขั้นตอนที่ 4 ทำการปรับปรุงตารางชุดข้อมูล (รูปที่ 16) เมื่อทำการรวมคลัสเตอร์แล้ว จะเหลือค่าที่มากที่สุดของทั้ง 2 คลัสเตอร์เอาไว้ จากตัวอย่างนี้ คะแนนคนที่ 1 = 10 และคนที่ 2=7 เมื่อรวมคลัสเตอร์ (1,2) เข้าด้วยกัน คะแนนสอบที่จะเป็นตัวแทนของกลุ่มใหม่คือ 10 นั่นคือ (1,2)=10 ดังรูปที่ 22 และปรับปรุงเมตริกซ์ดังรูปที่ 23

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

รูปที่ 22 ผลลัพธ์การปรับปรุงตารางชุดข้อมูลเมื่อรวมคลัสเตอร์ (1,2)

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

รูปที่ 23 ปรับปรุงค่าภายในเมตริกซ์เมื่อรวมคลัสเตอร์ (1,2)

ขั้นตอนที่ 5 ทำซ้ำขั้นที่ 3-4 จนกว่าจะเหลือเพียงคลัสเตอร์เดียว

- ขั้นที่ 5.1 ทำการรวมคลัสเตอร์ 3,5 และได้ผลลัพธ์ดังรูปที่ 24-26

รูปที่ 24 ทำการรวมคลัสเตอร์ (3,5)

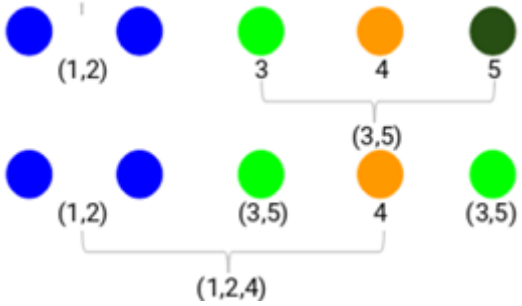
StudentID	Marks
(1,2)	10
(3,5)	35
4	20

รูปที่ 25 ผลลัพธ์การปรับปรุงตารางชุดข้อมูลเมื่อรวมคลัสเตอร์ (3,5)

ID	(1,2)	(3,5)	4
(1,2)	0	25	10
(3,5)	25	0	15
4	10	15	0

รูปที่ 26 ปรับปรุงค่าภายในเมตริกซ์เมื่อรวมคลัสเตอร์ (3,5)

- ขั้นที่ 5.2 ทำการรวมคลัสเตอร์ (1,2,4) และได้ผลลัพธ์ดังรูปที่ 27-29



รูปที่ 27 ทำการรวมคลัสเตอร์ (1,2,4)

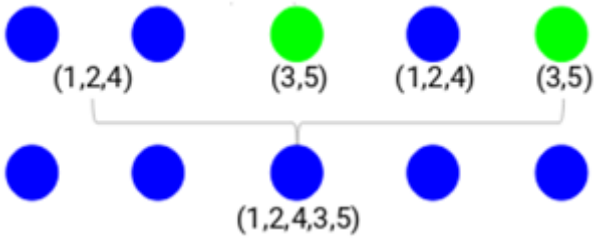
StudentID	Marks
(1,2,4)	20
(3,5)	35

รูปที่ 28 ผลลัพธ์การปรับปรุงตารางชุดข้อมูลเมื่อรวมคลัสเตอร์ (1,2,4)

ID	(1,2,4)	(3,5)
(1,2,4)	0	15
(3,5)	15	0

รูปที่ 29 ปรับปรุงค่าภายในเมตริกซ์เมื่อรวมคลัสเตอร์ (1,2,4)

- ขั้นที่ 5.3 ทำการรวมคลัสเตอร์ (1,2,3,4,5) และได้ผลลัพธ์ดังรูปที่ 30-31

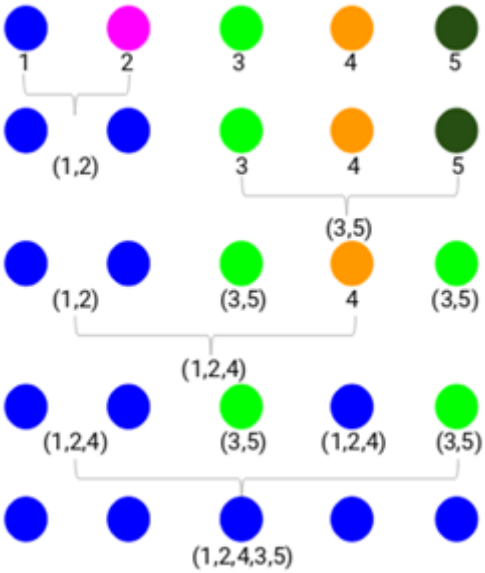


รูปที่ 30 ทำการรวมคลัสเตอร์ (1,2,3,4,5)

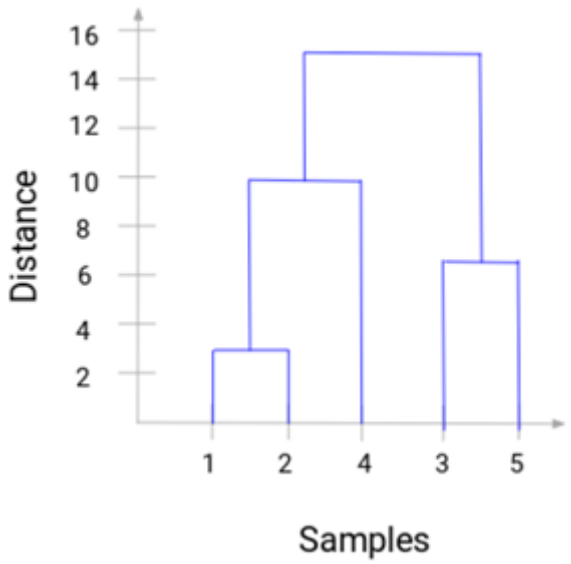
StudentID	Marks
(1,2,4,3,5)	35

รูปที่ 31 ผลลัพธ์การปรับปรุงตารางชุดข้อมูลเมื่อรวมคลัสเตอร์ (1,2,3,4,5)

จากทุกขั้นตอนที่อธิบายมานี้สามารถสรุปได้ดังรูปที่ 32 และสร้างกราฟ Dendrogram ดังรูปที่ 33



รูปที่ 32 สรุปขั้นตอนการรวมคลัสเตอร์ในแต่ละขั้นตอน



รูปที่ 33 กราฟเด็นโดรแกรมที่ได้จากวิธีการ Agglomerative Hierarchical Cluster

จากการแบ่งกลุ่มข้อมูลโดยใช้ Hierarchical Algorithm จะแสดงให้เห็นถึงความสัมพันธ์ของข้อมูลว่ามีข้อมูลใดมีความสัมพันธ์กันอย่างไร เหมาะกับการนำไปใช้วิเคราะห์ข้อมูลในกรณีที่ไม่ทราบจำนวนกลุ่มของข้อมูลที่ต้องการแบ่ง แต่เนื่องจากเป็นอัลกอริทึมที่ใช้เวลาในการทำคลัสเตอร์แต่ละกลุ่มนาน จึงไม่เหมาะกับข้อมูลที่มีปริมาณมาก สำหรับข้อมูลปริมาณมากจึงอาจต้องใช้ K-Means ในการจัดกลุ่มแทน

Numpy

ศึกษา Python Library สำหรับ Data

มีชื่อเต็มว่า “Numerical Python” ซึ่งแน่นอนว่า ชื่อมาขนาดนี้แล้ว NumPy ต้องโดดเด่นในในด้านการคำนวณ และการทำงานกับตัวเลขอย่างแน่นอน

นอกจากนี้ NumPy ยังมีความสามารถสำคัญในการสร้าง Array (โครงสร้างข้อมูล) และ Multidimensional Array ได้ ทำให้การคำนวณบน Python มีความรวดเร็วมากขึ้น ซึ่งแม้ Python พื้นฐานเอง จะมี Python list ที่มีความคล้ายคลึงกับ Array แต่ NumPy สามารถจัดการข้อมูลเหล่านี้ได้เร็วกว่าการใช้ Python list ธรรมดาๆ

NumPy จะถูกนำไปใช้พัฒนา Library อื่นอีกด้วย เช่น Matplotlib และ pandas

Numpy ใช้ทำอะไร?

หลักของ NumPy คืออาร์เรย์หลายมิติที่เป็นเนื้อเดียวกัน มันเป็นตารางขององค์ประกอบ (โดยปกติจะเป็นตัวเลข) ซึ่งเป็นชนิดเดียวกันทั้งหมดจัดทำดัชนีโดย tuple ของจำนวนเต็มที่ไม่ใช่ค่าลบ ในมิติ NumPy เรียกว่า*แกน*

ตัวอย่างเช่นอาร์เรย์สำหรับพิกัดของจุดในพื้นที่ 3 มิติมีแกนเดียว แกนนั้นมี 3 องค์ประกอบอยู่ในนั้น ดังนั้นเราจึงบอกว่ามันมีความยาว 3 ในตัวอย่างภาพด้านล่างอาร์เรย์มี 2 แกน แกนแรกมีความยาว 2 แกนที่สองมีความยาว 3[1, 2, 1]

```
[[1., 0., 0.],
 [0., 1., 2.]]
```

การสร้างอาร์เรย์

มีหลายวิธีในการสร้างอาร์เรย์

ตัวอย่างเช่น คุณสามารถสร้างอาร์เรย์จากรายการ Python ปกติหรือ tuple โดยใช้ฟังก์ชัน ชนิดของอาร์เรย์ผลลัพธ์จะถูกอนุมานจากชนิดขององค์ประกอบในลำดับarray

```
>>> import numpy as np
>>> a = np.array([2, 3, 4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

ฟังก์ชั่นสร้างอาร์เรย์ที่เต็มไปด้วยศูนย์ฟังก์ชันจะสร้างอาร์เรย์ที่เต็มไปด้วยอาร์เรย์และฟังก์ชันจะสร้างอาร์เรย์ที่มีเนื้อหาเริ่มต้นเป็นแบบสุ่มและขึ้นอยู่กับสถานะของหน่วยความจำ โดยค่าเริ่มต้น dtype ของอาร์เรย์ที่สร้างขึ้นคือ แต่สามารถระบุได้ผ่านทางอาร์กิวเมนต์คำสำคัญ .zerosonesemptyfloat64dtype

```
>>> np.zeros((3, 4))
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
>>> np.ones((2, 3, 4), dtype=np.int16)
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
>>> np.empty((2, 3))
array([[3.73603959e-262, 6.02658058e-154, 6.55490914e-260],  # may vary
       [5.30498948e-313, 3.14673309e-307, 1.00000000e+000]])
```

ในการสร้างลำดับของตัวเลข NumPy มีฟังก์ชันที่คล้ายกับ Python ในตัว แต่ส่งกลับอาร์เรย์arangerange และเมื่อใช้กับอาร์กิวเมนต์จุดลอยตัวโดยทั่วไปจะไม่สามารถทำนายจำนวนองค์ประกอบที่ได้รับได้เนื่องจากความแม่นยำของจุดลอยตัวที่ จำกัด
ด้วยเหตุนี้จึงเป็นการดีกว่าที่จะใช้ฟังก์ชันที่ได้รับเป็นอาร์กิวเมนต์จำนวนองค์ประกอบที่เราต้องการแทนที่จะเป็นขั้นตอน: arangelinspace

```
>>> np.arange(10, 30, 5)
array([10, 15, 20, 25])
>>> np.arange(0, 2, 0.3)  # it accepts float arguments
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])
```

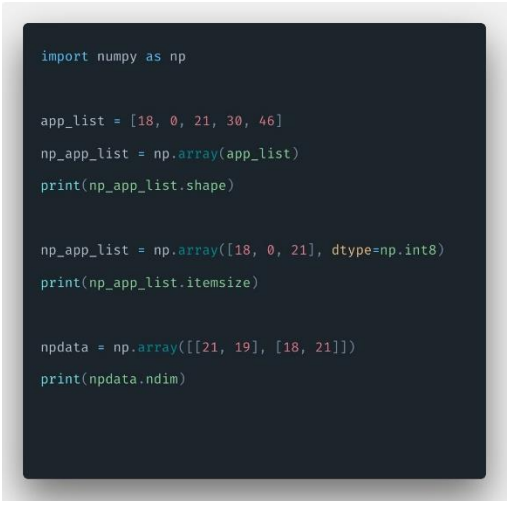
```
>>> from numpy import pi
>>> np.linspace(0, 2, 9)                 # 9 numbers from 0 to 2
array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
>>> x = np.linspace(0, 2 * pi, 100)      # useful to evaluate function at lots of point
>>> f = np.sin(x)
```

NumPy Arrays เปรียบเทียบกับ 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

```
listA = ['Game Of Thrones', 8, 'Stranger Things', 3, 'Friends', 10]
listB = ['Emilia Clarke', 'Millie Bobby Brown', 'Jennifer Aniston']
print(listA)
print(listB)

listA = ['Game Of Thrones', 8, 'Stranger Things', 3, 'Friends', 10]
listB = ['Emilia Clarke', 'Millie Bobby Brown', 'Jennifer Aniston']
print("listA[2]: ", listA[2])
print("listB[1:2]: ", listB[1:2])

listB = ['Emilia Clarke', 'Millie Bobby Brown', 'Jennifer Aniston']
print('Before change', listB)
listB[2] = 'Emma Mackey'
print('After change', listB)
```

Python Lists



```
import numpy as np

app_list = [18, 0, 21, 30, 46]
np_app_list = np.array(app_list)
print(np_app_list.shape)

np_app_list = np.array([18, 0, 21], dtype=np.int8)
print(np_app_list.itemsize)

npdata = np.array([[21, 19], [18, 21]])
print(npdata.ndim)
```

NumPy Arrays

NumPy Arrays มีประสิทธิภาพที่ดีกว่า Python Lists ทั้งการเข้าถึงข้อมูล , การเปลี่ยนแปลงข้อมูล , การดำเนินการทางคณิตศาสตร์ต่างๆ , การประหยัด Memory และ NumPy Arrays มีขนาดคงที่เมื่อสร้าง ซึ่งแตกต่างจาก Python Lists (ซึ่งสามารถขยายได้แบบไดนามิก) ทำให้ Numpy Arrays เป็นที่นิยมในการใช้งานใน Data Science

ตัวอย่าง Numpy

ตัวอย่าง เวลาจะใช้งานก็ import เข้ามาใช้งานได้เหมือนตัวอย่างด้านล่างนี้เลย

```
import numpy as np
a = np.array([1,2,3])
print(a)        # [1 2 3]
print(type(a))  # <class 'numpy.ndarray'>
```

ตัวอย่าง Array Numpy นั้นสามารถสร้างและจัดการกับ array หลายมิติได้ง่ายๆและหลายวิธี

```
import numpy as np

a = np.array([1, 2, 3])   # สร้าง array 1 มิติ
print(a)                  # [1 2 3]
print(a[0], a[1], a[2])   # 1 2 3
a[2] = 5                  # เปลี่ยนค่าใน array
print(a)                  # [1 2 5]

b = np.array([[1,2,3],[4,5,6]])    # สร้าง array 2 มิติ
print(b)                           # [[1 2 3]
                                   #  [4 5 6]]

# เราสามารถดูขนาดรูปร่างของ array ได้
print(a.shape)            # (3,)
print(b.shape)            # (2, 3)
```

ตัวอย่าง สร้าง Array ที่มีข้อภายใน

```
import numpy as np

zero = np.zeros((2,2))   # สร้าง array ที่ทุกค่าเป็น 0
print(zero)              # [[0. 0.]
                         #  [0. 0.]]

one = np.ones((3,2))     # สร้าง array ที่ทุกค่าเป็น 1
print(one)               # [[1. 1.]
                         #  [1. 1.]
                         #  [1. 1.]]

five = np.full((3,3), 5) # สร้าง array ที่ทุกค่าเป็นเลขที่ระบุ (5)
print(five)              # [[5 5 5]
                         #  [5 5 5]
                         #  [5 5 5]]

identity = np.eye(5)     # สร้างเมทริกซ์เอกลักษณ์ขนาด 5 x 5
print(identity)          # [[1. 0. 0. 0. 0.]
                         #  [0. 1. 0. 0. 0.]
                         #  [0. 0. 1. 0. 0.]
                         #  [0. 0. 0. 1. 0.]
                         #  [0. 0. 0. 0. 1.]]

random = np.random.random((3,3))  # สร้าง array ที่สุ่มค่าด้านใน
print(random)                     # [[0.06410201 0.14923082 0.32935854]
                                  #  [0.4784618  0.70607096 0.32578298]
                                  #  [0.90210072 0.51487876 0.98004843]]
```

ตัวอย่าง การทำให้ค่า Array อยู่ในช่วงที่กำหนด

```
import numpy as np

max_value = 4
min_value = 1
a = [1, 22, 99, 0, 6, 8, -2, 3, 4, 3, 1]
print(np.clip(a, min_value, max_value))    # [1 4 4 1 4 4 1 3 4 3 1]
```

ตัวอย่าง การค้นหาตำแหน่งของของของข้อมูลที่ผ่านเงื่อนไข

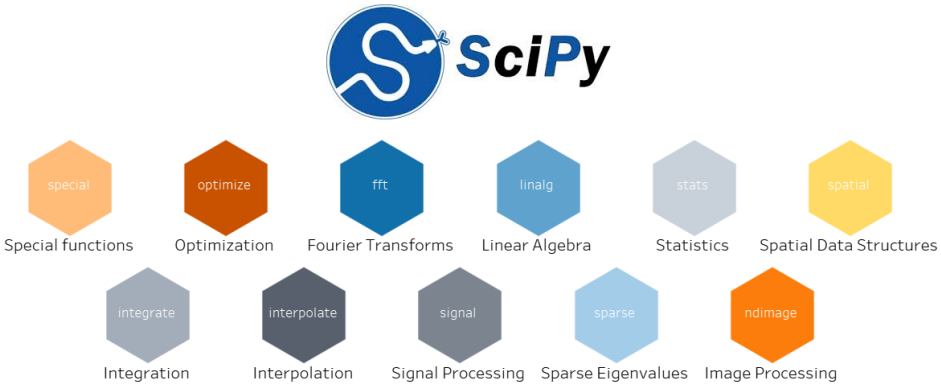
```
import numpy as np

max_value = 4
min_value = 1
a = [1, 22, 99, 0, 6, 8, -2, 3, 4, 3, 1]
print(np.clip(a, min_value, max_value))    # [1 4 4 1 4 4 1 3 4 3 1]
```

ตัวอย่าง การหา Percentile

```
import numpy as np

a = np.array([1,2,3,4,5,6,7,8,9,10])
print(np.percentile(a, 50, axis =0))   # 5.5

b = np.array([5, 1, 3, 7, 9])
print(np.percentile(b, 25, axis =0))   # 3.0
```

SciPy

SciPy เป็น library ที่ใช้ใน python สำหรับการคำนวณทางวิทยาศาสตร์ ซึ่งมีฟังก์ชันหลากหลายไม่ว่าจะเป็น linear algebra, calculus หรือ optimization SciPy สามารถนำมาใช้ควบคู่และเติมเต็ม NumPy ได้อย่างดีเยี่ยม



ฟังก์ชัน curve_fit จาก SciPy

ฟังก์ชันที่เราจะใช้กันก็คือ curve_fit จาก library scipy.optimize
ที่สามารถใช้ในการปรับสมการให้เข้ากับข้อมูลที่เรามีมากที่สุด
โดยเราจะต้องกำหนดหน้าตาของสมการตัวนั้นมาก่อน แล้วฟังก์ชัน curve_fit จะพยายามหาค่า parameter ที่จะทำให้สมการของเรานั้นใกล้เคียงกับข้อมูลจริงที่สุด

เริ่มต้นจากการ import library จะใช้ matplotlib มาสำหรับการสร้างกราฟและ SciPy ในการ fit curve

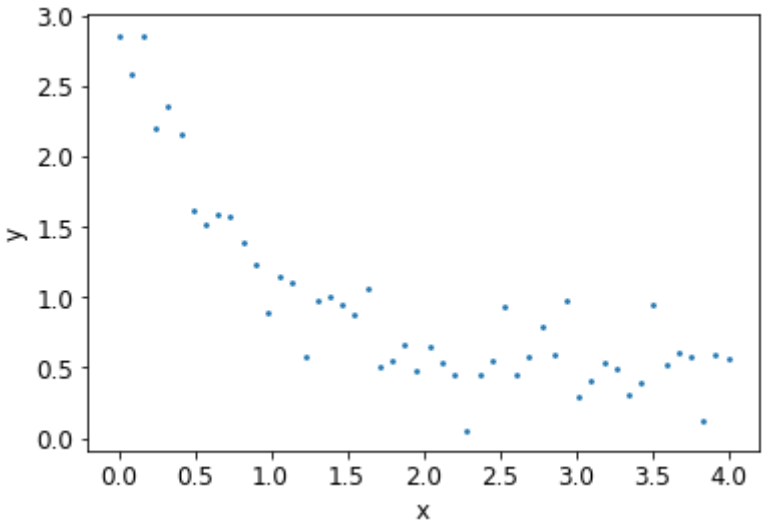
```
import numpy as np

import matplotlib.pyplot as plt from scipy.optimize

import curve_fit
```

สมมติฟังก์ชันขึ้นมาตัวนึงสำหรับสมการที่จะใช้ เป็นสมการเอกซ์โพเนนเชียล(Exponential) : $f(x) = (ae^{-bx} + c)$ ที่มี parameter ที่ไม่รู้ค่าคือ a, b และ c โดยใช้ฟังก์ชัน np.exp จาก NumPy

```
def func(x, a, b, c)

return a * np.exp(-b * x) + c
```

ยังไม่มีข้อมูลจริงมาใช้ ดังนั้นก็สร้างมันขึ้นมาก่อน โดยตั้งให้ คือ a = 2.5, b = 1.3 และ c = 0.5 แต่ใส่ noise ขึ้นมาเพื่อให้ข้อมูลกระจัดกระจาย

```
xdata = np.linspace(0, 4, 50)

y = func(xdata, 2.5, 1.3, 0.5)

np.random.seed(1729)

y_noise = 0.2 * np.random.normal(size=xdata.size)

ydata = y + y_noise

plt.scatter(xdata, ydata, s = 2)
```



มีตัวฟังก์ชันและข้อมูลแล้ว ก็สามารถเรียก curve_fit ได้

```
popt, pcov = curve_fit(func, xdata, ydata)
```

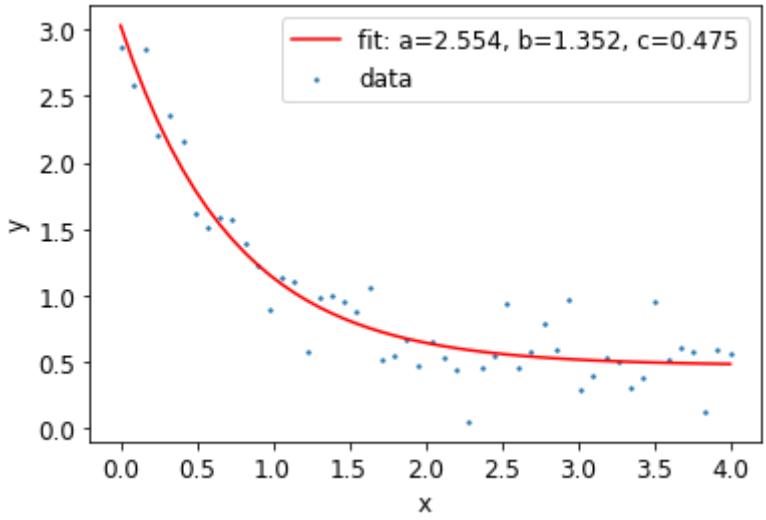
output ของ curve_fit นั้นมีสองตัวคือ

- popt เป็นค่าที่ดีที่สุดสำหรับ parameter จากสมการของเรา
- pcov เป็นค่าทางสถิติที่บอกถึงการกระจายตัวของข้อมูลและความคลาดเคลื่อนของค่า parameter ที่หามาได้

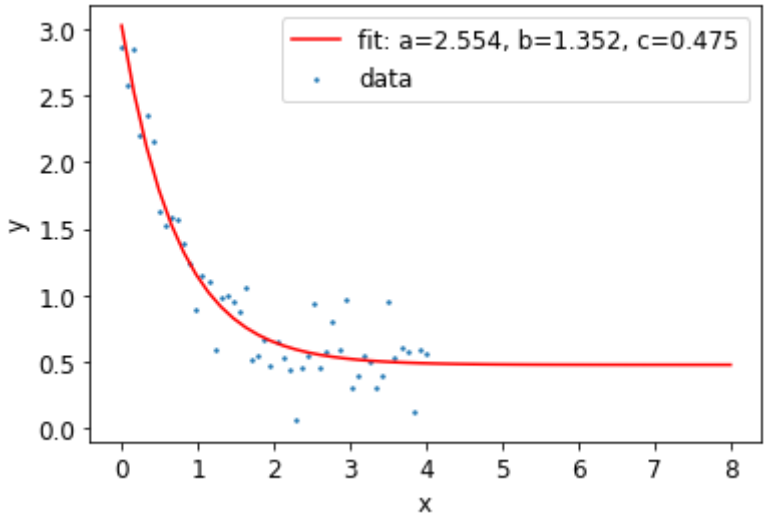
เราจะสนใจแค่ popt ซึ่งจะบอกค่า a, b และ c ที่เราหามาได้ และ นำมาสร้างกราฟเทียบกับข้อมูลจริง

```
 plt.scatter(xdata, ydata, label='data', s = 2)

plt.plot(xdata, func(xdata, *popt), 'r-',

label='fit: a=%5.3f, b=%5.3f, c=%5.3f' % tuple(popt))
```

plt.legend()



ก็ได้สมการและเส้นโค้งสำหรับข้อมูลที่มีและก็ยังสามารถใช้พยากรณ์ข้อมูลต่อไปได้อีก



อย่างไรก็ตามข้อจำกัดของ curve_fit ก็คือต้องรู้รูปแบบของสมการที่จะใช้ก่อน ไม่งั้นจะไม่สามารถทำอะไรได้ แต่ข้อดีคือรูปแบบของสมการนั้นเป็นอะไรก็ได้ที่เป็นฟังก์ชันทางคณิตศาสตร์ที่สามารถเขียนเป็นฟังก์ชันใน python ได้ และจะมี parameter กี่ตัวก็ได้

Matplotlib

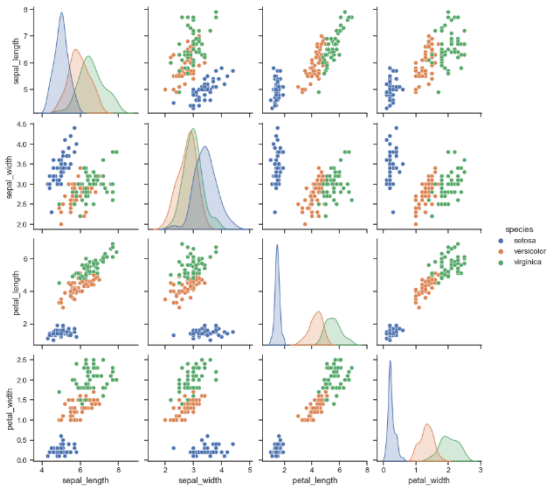
Matplotlib เป็น Module พื้นฐานในการสร้างกราฟใน Python โดยที่เราจะนำไปต่อยอดด้วย Modulbe อื่นที่ทำกราฟได้สวยขึ้นเช่น Seaborn ด้วย การสร้างกราฟด้วยการเรียกใช้ python script ใน Power BI ได้

การทำงาน Matplotlibการสร้างกราฟใน Python ด้วย Matplotlib นั้นจะต้องเขียน Code
เพื่อสั่งว่ากราฟของเราจะมีส่วนประกอบอะไร หน้าตายังไงบ้าง
โดยที่หลังจากที่บอกมันว่าส่วนประกอบเป็นอะไรบ้าง ถ้าเป็นการทำ python ใน script
ปกติเราจะยังคงมองไม่เห็นผลลัพธ์ทันที ถ้าอยากจะเห็นว่าเป็นยังไง ต้องสั่งให้มัน show กราฟออกมา

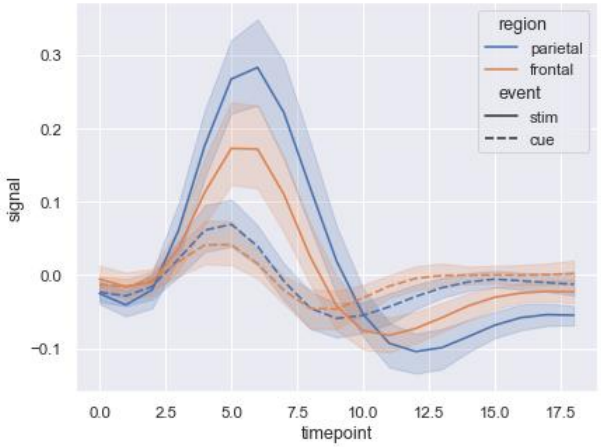
ซึ่งคำสั่ง plt.show() จะไปไล่หาว่ามี object กราฟอะไรถูกสร้างขึ้นมาบ้างจากนั้นจึงจะแสดงออกมาทุกอัน

Seaborn

Seaborn เป็นไลบรารีสำหรับสร้างกราฟิกทางสถิติในภาษา Python ถูกสร้างขึ้นบน Matplotlib และรวมเข้ากับโครงสร้างข้อมูลของ Pandas เป็นเครื่องมือที่แตกต่างจาก Matplotlib ตรงที่มีความเชี่ยวชาญในการแสดงภาพสถิติ ใช้ไวยากรณ์น้อยกว่า และมีธีมเริ่มต้นที่เข้าใจง่าย



ตัวอย่าง



seaborn components used: `set_theme()`, `load_dataset()`, `lineplot()`

```
import seaborn as sns
sns.set_theme(style="darkgrid")

# Load an example dataset with long-form data
fmri = sns.load_dataset("fmri")

# Plot the responses for different events and regions
sns.lineplot(x="timepoint", y="signal",
             hue="region", style="event",
             data=fmri)
```

Pandas

Pandas เป็นโครงสร้างข้อมูลที่ใช้งานง่ายและประสิทธิภาพสูง มันสร้างขึ้นบน Numpy Scipy Matplotlib

โดยโครงสร้างข้อมูลเป็นการจัดการตารางตัวเลขและชุดข้อมูล

Pandas Series คือโครงสร้างข้อมูลมีลักษณะคล้ายnumpy 1-d array

แต่มีความสามารถมากกว่าประกอบด้วยอาร์เรย์สองอาร์เรย์คือvalues, index แต่ access

ข้อมูลจะใช้ค่าตำแหน่ง(iloc)(สามารถกำหนดได้หลายตัว)และใช้ค่าของindex (loc)

(สามารถกำหนดได้หลายตัว)และBoolean Array

Pandas Dataframe คือโครงสร้างข้อมูลแบบheterogenous data typeมีลักษณะคล้ายExcel

Spreadsheet

```
import pymysql
import pandas as pd

dbcon = pymysql.connect("localhost", "root", "root", "laravel7crud")

try:
    SQL_Query = pd.read_sql_query(
        '''select
         symptoms,
         country_name,
         cases
         from coronas''', dbcon)

    df = pd.DataFrame(SQL_Query, columns=['symptoms', 'country_name', 'cases'])
    print(df)
    print('The data type of df is: ', type(df))
except:
    print("Error: unable to convert the data")

dbcon.close()
```

Team	Position	Name	Number	Age	Height	Weight	College	Salary
Atlanta Hawks	C	Al Horford	15.0	30.0	6-10	245.0	Florida	12000000.0
	PF	Kris Humphries	43.0	31.0	6-9	235.0	Minnesota	1000000.0
	PG	Dennis Schroder	17.0	22.0	6-1	172.0	Wake Forest	1763400.0
	SF	Kent Bazemore	24.0	26.0	6-5	201.0	Old Dominion	2000000.0
	SG	Tim Hardaway Jr.	10.0	24.0	6-6	205.0	Michigan	1304520.0
Boston Celtics	C	Kelly Olynyk	41.0	25.0	7-0	238.0	Gonzaga	2165160.0
	PF	Jonas Jerebko	8.0	29.0	6-10	231.0	LSU	5000000.0
	PG	Avery Bradley	0.0	25.0	6-2	180.0	Texas	7730337.0
	SF	Jae Crowder	99.0	25.0	6-6	235.0	Marquette	6796117.0
	SG	John Holland	30.0	27.0	6-5	205.0	Boston University	1148640.0
Brooklyn Nets	C	Brook Lopez	11.0	28.0	7-0	275.0	Stanford	19689000.0
	PF	Chris McCullough	1.0	21.0	6-11	200.0	Syracuse	1140240.0
	PG	Jarrett Jack	2.0	32.0	6-3	200.0	Georgia Tech	6300000.0
	SG	Bojan Bogdanovic	44.0	27.0	6-8	216.0	Oklahoma State	3425510.0
Charlotte Hornets	C	Al Jefferson	25.0	31.0	6-10	289.0	Wisconsin	13500000.0
	PF	Tyler Hansbrough	50.0	30.0	6-9	250.0	North Carolina	947276.0
	PG	Jorge Gutierrez	12.0	27.0	6-3	189.0	California	189455.0
	SF	Michael Kidd-Gilchrist	14.0	22.0	6-7	232.0	Kentucky	6331404.0