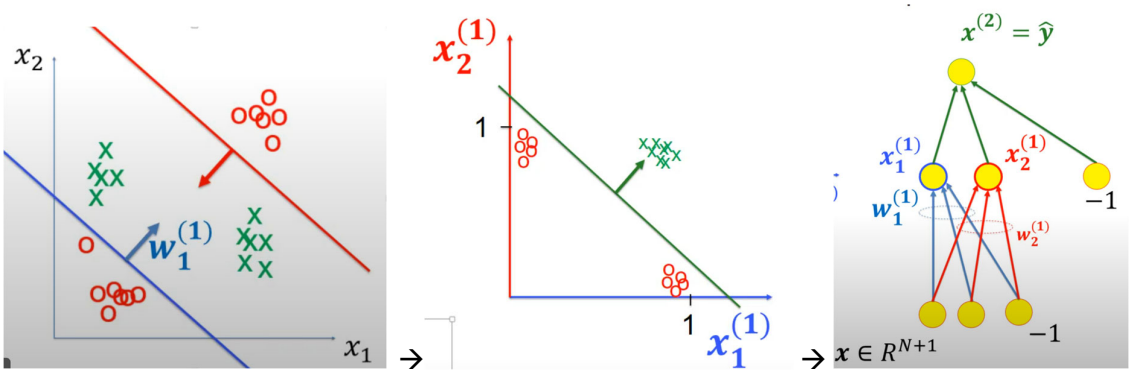
6. Non-linear Classifiers

6.1 ทำไมต้องใช้ **Non-linear Classifiers?**

บางปัญหา เช่น XOR Problem หรือข้อมูลที่มีลักษณะซับซ้อน จะไม่สามารถใช้เส้นตรงแบ่งกลุ่มได้อย่างถูกต้อง ตัวอย่างเช่น:

- ข้อมูลที่กระจายเป็นวงกลมซ้อนกัน
- ข้อมูลที่มีรูปแบบซับซ้อน เช่น รูปหัวใจ หรือเกลียว

XOR Problem



ในกรณีเหล่านี้ Linear Classifiers เช่น Linear SVM จะไม่สามารถจำแนกได้ดี จึงต้องใช้ Non-linear Classifiers ที่สามารถสร้างเส้นแบ่งที่โค้งงอหรือซับซ้อน

6.2 หลักการทำงานของ **Non-linear Classifiers**

1.การใช้โมเดลที่มีโครงสร้างไม่เชิงเส้น

- เช่น Decision Trees, Neural Networks, หรือ k-NN
- โมเดลเหล่านี้ไม่จำเป็นต้องใช้เส้นตรงในการแบ่งข้อมูล

2. การแปลงข้อมูล (Feature Transformation)

- ใช้เทคนิคเช่น Kernel Trick เพื่อแปลงข้อมูลให้อยู่ในมิติที่สูงขึ้น ซึ่งในมิติใหม่นั้นข้อมูลอาจสามารถแบ่งด้วยเส้นตรงได้
- ตัวอย่างเช่น SVM ที่ใช้ RBF Kernel หรือ Polynomial Kernel

6.3 K-Nearest Neighbors (KNN)

k-Nearest Neighbors (k-NN) เป็นอัลกอริธึมการเรียนรู้ของเครื่องแบบมีผู้สอน (Supervised Learning) ที่ใช้สำหรับการจำแนกประเภท (Classification) และการประมาณค่า (Regression) โดยหลักการคือ: เมื่อมีข้อมูลใหม่เข้ามา k-NN จะดูว่า "ข้อมูลใกล้เคียงที่สุด k ตัว" คืออะไร แล้วใช้ข้อมูลเหล่านั้นในการตัดสินใจ

6.3.1 หลักการทำงานของ k-NN Algorithm

1. กำหนดค่า k: จำนวนเพื่อนบ้านที่ใกล้ที่สุดที่เราจะพิจารณา
2. คำนวณระยะห่าง ระหว่างข้อมูลใหม่กับข้อมูลในชุดฝึก

   Euclidean Distance

   $$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

   Manhattan or city-block

   $$d(A,B) = \sqrt{|x_1 - x_2| + |y_1 - y_2|}$$
3. เลือก k ตัวที่ใกล้ที่สุด
4. โหวตหรือเฉลี่ยผล จากเพื่อนบ้านเหล่านั้น:
   - ถ้าเป็น classification → ใช้การโหวตหาคลาสที่พบมากที่สุด

Tutorial: kNN in the Iris data set

6.4 Decision Trees

Decision Tree เป็นอัลกอริธึมการเรียนรู้ของเครื่องแบบมีผู้สอน (Supervised Learning) ที่ใช้สำหรับทั้ง การจำแนกประเภท (Classification) และ การประมาณค่า (Regression) โดยมีโครงสร้างคล้ายกับ "ต้นไม้" ที่ประกอบด้วย:

- Node (โหนด): จุดตัดสินใจ
- Branch (กิ่ง): เส้นทางที่เลือกตามเงื่อนไข
- Leaf (ใบ): ผลลัพธ์สุดท้าย เช่น คลาสที่จำแนกได้

6.4.1 หลักการทำงาน

1. เริ่มจาก โหนดราก (Root Node) ที่มีข้อมูลทั้งหมด
2. เลือกคุณลักษณะ (Feature) ที่ดีที่สุดในการแบ่งข้อมูล โดยใช้เกณฑ์เช่น:
    - Gini Impurity วัดความน่าจะเป็นที่ข้อมูลที่เลือกมาแบบสุ่มจากชุดข้อมูล จะถูกจัดอยู่ในคลาสที่ผิด ถ้าเราจัดกลุ่มตามการกระจายของคลาสในโหนดนั้น

      Gini Impurity : 

      สำหรับชุดข้อมูลที่มี c คลาส:

      $$Gini = 1 - \sum_{i=1}^{c} p_i^2$$

      โดยที่ ***pi*** คือสัดส่วนของข้อมูลในคลาสที่ ***i***

    - เกณฑ์การแบ่งข้อมูล

      Entropy : 

      $$Entropy(S) = -\sum_{i=1}^{c} p_i log_2(p_i)$$

      Information Gain :

      $$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

3. แบ่งข้อมูลออกเป็นกลุ่มย่อยตามเงื่อนไข
4. ทำซ้ำขั้นตอนนี้จนกว่าจะถึงเงื่อนไขหยุด เช่น:
    - ข้อมูลในโหนดมีคลาสเดียวกันหมด
    - ความลึกของต้นไม้ถึงขีดจำกัด

 **Decision Tree | ID3 Algorithm**	 **Decision-Tree Classifier Tutorial**

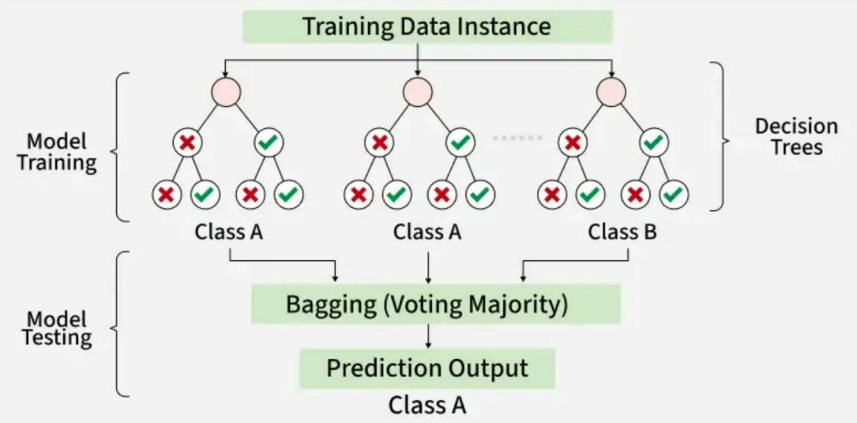
6.5 Random Forest

Random Forest คือการรวมหลาย ๆ ต้นไม้ตัดสินใจ (Decision Trees) เข้าด้วยกัน โดยให้แต่ละต้นไม้ทำนายผล แล้วใช้การโหวต (สำหรับ classification) หรือค่าเฉลี่ย (สำหรับ regression) เพื่อให้ได้ผลลัพธ์สุดท้าย

เป็นการรวมหลาย ๆ Decision Trees เข้าด้วยกัน เพิ่มความแม่นยำและลดการ overfitting

6.5.1 หลักการทำงาน

1. Bootstrap Sampling (Bagging):
    - สุ่มข้อมูลจากชุด training มาแบบมีการแทนที่ (sampling with replacement) เพื่อสร้างชุดข้อมูลย่อยหลายชุด
    - แต่ละชุดใช้สร้างต้นไม้ตัดสินใจของตัวเอง



รูป Bootstrap Sampling (Bagging)

2. Random Feature Selection:
    - ตอนสร้างแต่ละต้นไม้ จะสุ่มเลือกเฉพาะบาง features มาใช้ในการ split ที่แต่ละ node
    - ช่วยลดความสัมพันธ์ระหว่างต้นไม้แต่ละต้น (decorrelation)
3. Voting/Averaging:
    - สำหรับ classification: ใช้เสียงข้างมากจากต้นไม้ทั้งหมด
    - สำหรับ regression: ใช้ค่าเฉลี่ยของผลลัพธ์จากต้นไม้ทั้งหมด

Random Forest Algorithm

ข้อดี

- ลดปัญหา overfitting ที่พบใน decision tree เดี่ยว
- ทำงานได้ดีแม้ข้อมูลมี noise หรือ missing values
- สามารถวัดความสำคัญของ features ได้ (feature importance)

ข้อเสีย

- โมเดลใหญ่และซับซ้อน อาจใช้เวลาในการเทรนและทำนายมากขึ้น
- ยากต่อการตีความ (interpretability) เมื่อเทียบกับต้นไม้เดี่ยว

6.6 Kernel Support Vector Machines (Kernel SVM)

Kernel SVM ใช้ kernel trick link2 เพื่อแปลงข้อมูลไปยังมิติที่สูงขึ้น ทำให้สามารถแยกข้อมูลที่ไม่เชิงเส้นได้

Kernel Trick คือการแปลงข้อมูลจาก space เดิม ไปยัง space ที่สูงขึ้น (higher-dimensional space) โดยไม่ต้องคำนวณตำแหน่งจริง ใน space ใหม่ ข้อมูลอาจสามารถแยกด้วยเส้นตรงได้

ประเภทของ Kernel

- Linear เส้นตรง ข้อมูลที่แยกได้ด้วยเส้นตรง

$$K(x, y) = x^\top y$$

- Polynomial พหุนาม ข้อมูลที่มีความสัมพันธ์เชิงกำลัง

$$K(x, y) = (x^\top y + c)^d$$

c : ค่าคงที่ (bias term)

d : องศาของพหุนาม (degree)

- Radial Basis Function (RBF)  link2 หรือ Gaussian ข้อมูลที่มีลักษณะกระจายเป็นกลุ่ม

$K(x, y) = exp(-\gamma \| x - y \|^2)$

γ : พารามิเตอร์ที่ควบคุมความกว้างของ kernel (ยิ่งมาก ยิ่งแคบ)

- Sigmoidคล้ายกับ Neural Network ใช้ในบางกรณีเฉพาะ

$K(x, y) = tanh(\alpha x^\top y + c)$

α : พารามิเตอร์สเกล

c : bias term