บทที่ 3 หลักการพื้นฐานของการจำแนกประเภท

หัวข้อ

3.1 คำนิยามและแนวคิดพื้นฐาน

3.2 การสร้างโมเดลการจำแนกประเภท

3.3 ตัวจำแนกประเภทต้นไม้ตัดสินใจ

3.4 Model Overfitting

3.5 การคัดเลือกโมเดล (Model Selection)

3.6 การประเมินประสิทธิภาพโมเดล (Model Evaluation)

3.7 การประเมินประสิทธิภาพของโมเดลกับการเลือกค่าไฮเปอร์พารามิเตอร์

3.1 คำนิยามและแนวคิดพื้นฐาน

คำนิยาม 3.1 การจำแนกประเภท (Classification)

กำหนดชุดของเรคอร์ดข้อมูล training set แต่ละเรคอร์ดประกอบด้วย ทูเพิล (tuple) (**x**, y) โดยที่ **x** คือแอทริบิวต์เซต (attribute set) และ y คือ คลาสเลเบิล (class label) งานการจำแนกประเภท (classification task) คือการสร้างโมเดลซึ่งแมปจาก attribute set x ไปยังคลาสเลเบิล y ดังในรูปที่ 3.1

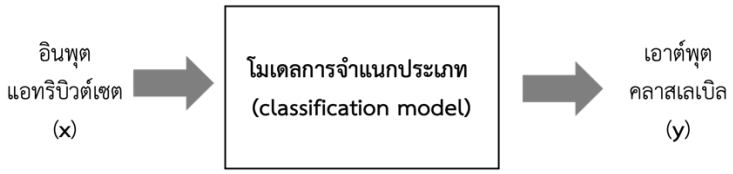
ตัวแปร x มีชื่อเรียกได้หลายอย่าง ได้แก่ attribute, predictor, independent variable, input

ตัวแปร y มีชื่อเรียกได้หลายอย่าง ได้แก่ class, response, depdendent variable, output

ตัวอย่างของงานการจำแนกประเภท มีดังนี้คือ

- การกรองอีเมล์ขยะ (spam filtering) คือการสร้างโมเดลที่แมปจากแอทริบิวต์เซต x ซึ่งประกอบด้วยฟีเจอร์ต่าง ๆ เกี่ยวกับอีเมล์ (เช่น อีเมล์ผู้ส่ง อีเมล์ผู้รับ หัวข้อเรื่อง ข้อความในอีเมล์) ไปเป็นคลาสเลเบิลที่เป็นไปได้สองค่าคือ spam และ non-spam
- การจำแนกชนิดเนื้องอก คือการสร้างโมเดลที่แมปจากแอทริบิวต์เซต x ซึ่งประกอบด้วยฟีเจอร์ต่างๆ เกี่ยวกับก้อนเนื้องอกที่ได้มาจากผลการเอกซเรย์ (X-Rays) หรือผลการตรวจด้วยคลื่นแม่เหล็กไฟฟ้า (Magnetic Resonance Imaging : MRI) ไปเป็นคลาสเลเบิลที่เป็นไปได้สองค่าคือ malignant (เนื้อร้าย) และ benign (เนื้องอกชนิดไม่อันตราย)
- การจำแนกสายพันธุ์ดอกไอริส คือการสร้างโมเดลที่แมปจากแอทริบิวต์ x ซึ่งประกอบด้วยฟีเจอร์ของดอกไอริส (ความกว้างกลีบเลี้ยง ความยาวกลีบเลี้ยง ความกว้างกลีบดอก ความยาวกลีบดอก) ไปเป็นคลาสเลเบิลที่เป็นไปได้ 3 ค่าคือ virginica setosa และ versicolor

จากตัวอย่างข้างต้น เราจะเรียกการกรองอีเมล์ขยะและการจำแนกชนิดเนื้องอก ว่าเป็นการจำแนกประเภทแบบสองคลาส (binary classification) เนื่องจากค่าที่เป็นไปได้ของคลาสเลเบิลมีจำนวนสองคลาส ส่วนการจำแนกสายพันธุ์ดอกไอริสถือเป็นการจำแนกประเภทแบบหลายคลาส (multi-class classification)



รูปที่ 3.1 งานการจำแนกประเภท คือการสร้างโมเดลที่แมปจาก แอทริบิวต์เซต x ไปเป็นคลาสเลเบิล y

โมเดลการจำแนกประเภท ถูกนำไปใช้ในสองรูปแบบ คือ ใช้เพื่อการทำนายป้ายกำกับ (predictive model) และใช้เพื่อการระบุคุณลักษณะเฉพาะของคลาสแต่ละชนิด (descriptive model) ตัวอย่างเช่น โมเดลการจำแนกประเภทดอกไอริส สามารถใช้เป็น predictive model เพื่อทำนายป้ายกำกับ (สายพันธุ์) ของดอกไอริสที่เราทราบขนาดความกว้างกลีบเลี้ยง ความยาวกลีบเลี้ยง ความกว้างกลีบดอก ความยาวกลีบดอก แต่ไม่ทราบสายพันธุ์ ในขณะเดียวกันโมเดลการจำแนกประเภทดอกไอริสนี้ก็สามารถใช้เป็น descriptive model เพื่ออธิบายคุณสมบัติเฉพาะตัวของดอกไอริสแต่ละสายพันธุ์ได้

แอทริบิวต์เซตซึ่งเป็นอินพุตของโมเดลการจำแนกประเภท สามารถเป็นได้ทั้งข้อมูลเชิงคุณภาพและเชิงปริมาณ แต่คลาสเลเบิล (ตัวแปรเป้าหมาย) จะต้องมีชนิดเป็นแบบ nominal เท่านั้น หากตัวแปรเป้าหมายไม่ใช่ข้อมูลชนิด nominal แล้วโมเดลการทำนายนั้นจะไม่ใช่โมเดลการจำแนกประเภท แต่จะเรียกว่า รีเกรสชันโมเดล (regression model)

3.2 การสร้างโมเดลการจำแนกประเภท

กระบวนการสร้างโมเดลการจำแนกประเภท แบ่งออกเป็น 2 ขั้นตอนหลัก คือ

1) **Induction Phase (ช่วงการอุปมานหรือการสร้างโมเดล)** คือกระบวนการสร้างโมเดลการจำแนกประเภท โดยใช้อัลกอริทึมการเรียนรู้ (**learning algorithm**) สกัดรูปแบบที่ซ่อนอยู่ในชุดข้อมูลฝึกฝน (**training set**)

2) **Deduction Phase (ช่วงการอนุมานหรือการใช้โมเดลจำแนกประเภท)** คือการนำโมเดลที่ได้จาก induction phase ไปใช้ในการจำแนกประเภทให้กับข้อมูลที่ไม่มีป้ายกำกับของชุดข้อมูลทดสอบ (**testing set**) และทดสอบประสิทธิภาพของโมเดลหรือตัวจำแนกประเภท (**classifier**) หากตัวจำแนกประเภทมีประสิทธิภาพอยู่ไม่อยู่เกณฑ์ที่ต้องการก็จะต้องกลับไปเริ่มกระบวนการเรียนรู้โมเดลในขั้นตอน induction phase ใหม่ แต่หากโมเดลมีประสิทธิภาพอยู่ในระดับที่ต้องการก็สามารถนำตัวจำแนกประเภทดังกล่าวไปใช้งานจริงได้

เทคนิคการจำแนกประเภท (**classifcation techniques**) หมายถึง รูปแบบวิธีการสำหรับการจำแนกประเภทซึ่งมีอยู่หลายรูปแบบ เช่น เทคนิคต้นไม้ตัดสินใจ (decision tree based techniques) ซึ่งจะกล่าวถึงในบทนี้ และเทคนิคอื่น ๆ ที่จะได้อธิบายในบทต่อไป เช่น เทคนิคกฎการจำแนก (rule-based techniques), เทคนิคเพื่อนบ้านใกล้เคียง (neartest neighbor), เทคนิคนาอีฟเบย์ (naive Bayes) และเครือข่ายความเชื่อแบบเบย์ (Bayesian belief network), ซัพพอร์ตเวกเตอร์แมชชีน (support vector machines), เครือข่ายประสาทเทียม (neural network), เทคนิคอองซอมเบิล (ensemble based techniques) เป็นต้น โมเดลการจำแนกประเภทที่มีประสิทธิภาพดี จะต้องสามารถทำนายประเภทของข้อมูลที่ไม่เคยพบมาก่อน (unseen data) ได้อย่างมีประสิทธิภาพ (ในทางเทคนิคเรียกว่ามี generalization performance) การประเมินประสิทธิภาพของตัวจำแนกประเภทแบบสองคลาส (binary classifier) ทำได้โดยการบันทึกจำนวนครั้งที่โมเดลทำนายถูกต้องและทำนายผิดพลาดของแต่ละคลาส ลงใน confusion matrix ดังเช่นตารางที่ 3.1 จากนั้นจึงคำนวณประสิทธิภาพโดยใช้วิธีการวัดประสิทธิภาพ (evaluation metrics) รูปแบบต่าง ๆ ตามความเหมาะสม เช่น ความถูกต้อง (accuary), อัตราส่วนความผิดพลาด (error rate) เป็นต้น

ตารางที่ 3.1 Confusion matrix สำหรับการจำแนกประเภทแบบสองคลาส

		คลาสที่โมเดลทำนาย (Predicted Class)	
		Class = 1	**Class = 0**
คลาสที่แท้จริง (Actual Class)	**Class = 1 (Positive)**	True Positive (TP) f_{11}	False Negative (FN) f_{10}
	Class = 0 (Negative)	False Positive (FP) f_{01}	True Negative (TN) f_{00}

จากตารางที่ 3.1 สามารถประเมินประสิทธิภาพของตัวจำแนกประเภทแบบสองคลาส ได้จาก evaluation metrics ต่างๆ ดังนี้คือ

Accuracy คือจำนวนครั้งที่ทำนายถูกต้องหารด้วยจำนวนครั้งทั้งหมดที่ทำนาย

$$Accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Error rate คือจำนวนครั้งที่ทำนายผิดพลาดหารด้วยจำนวนครั้งทั้งหมดที่ทำนาย

$$Error\ rate = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Recall หรือ Sensitivity คือจำนวนครั้งที่ทำนายคลาสบวกถูกต้องหารด้วยจำนวนคลาสบวกทั้งหมด

$$Recall = \frac{f_{11}}{f_{11} + f_{10}}$$

Precision คือจำนวนครั้งที่ทำนายคลาสบวกได้ถูกต้องหารด้วยจำนวนครั้งที่โมเดลลทำนายว่าเป็นคลาสบวก

$$Precision = \frac{f_{11}}{f_{11} + f_{01}}$$

True Negative Rate หรือ Specificity คือจำนวนครั้งที่ทำนายคลาสลบได้ถูกต้องหารด้วยจำนวนคลาสลบทั้งหมด

$$TNR = \frac{f_{00}}{f_{01} + f_{00}}$$

False Positive Rate คือจำนวนครั้งที่ทำนายคลาสลบผิดพลาดหารด้วยจำนวนคลาสลบทั้งหมด

$$FPR = \frac{f_{01}}{f_{01} + f_{00}} = 1 - TNR$$

F1-Score คือค่าเฉลี่ยฮาร์มอนิก (harmonic mean) ของ precision และ recall

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

ตัวอย่างที่ 3.1 กำหนดผลการทดสอบประสิทธิภาพของโมเดลการจำแนกประเภทแบบสองคลาสสองโมเดล ดัง confusion matrix ในข้อ (ก) และ (ข) จงประเมินประสิทธิภาพของโมเดลทั้งสองโดยคำนวณหาค่า Accuracy, Error rate, Recall, Precision, Specificity, False Positivie Rate และ F1-score

(ก)

Model 1		Predicted Class	
		Positive Class	Negative Class
Actual Class	Positive Class	130	20
	Negative Class	10	290

(ข)

Model 2		Predicted Class	
		Positive Class	Negative Class
Actual Class	Positive Class	140	10
	Negative Class	20	280

	Accuracy	Error rate	Recall	Precision	Specificity	FPR	F1-Score
Model 1	$\frac{130+290}{130+20+10+290}$	$\frac{20+10}{130+20+10+290}$	$\frac{130}{130+20}$	$\frac{130}{130+10}$	$\frac{290}{10+290}$	$\frac{10}{10+290}$	$\frac{2*0.929*0.867}{0.929+0.867}$
	0.933	**0.067**	**0.867**	**0.929**	**0.967**	**0.033**	**0.897**
Model 2	$\frac{140+280}{140+10+20+280}$	$\frac{10+20}{140+10+20+280}$	$\frac{140}{140+10}$	$\frac{140}{140+20}$	$\frac{280}{20+280}$	$\frac{20}{20+280}$	$\frac{2*0.875*0.933}{0.875+0.933}$
	0.933	**0.067**	**0.933**	**0.875**	**0.933**	**0.067**	**0.903**

จากผลการคำนวณประสิทธิภาพข้างต้น จะเห็นได้ว่า Model 1 มีค่า Precision, Specificity, และ FPR สูงกว่า Model 2 ในขณะที่ Model 2 มีค่า Recall และ F1-score ที่สูงกว่า ในการเลือกโมเดลเพื่อนำไปใช้งานจริงจะต้องคำนึงถึงเป้าหมายของงานเป็นสำคัญ ตัวอย่างเช่น หากงานจำแนกประเภทที่จะนำโมเดลไปใช้ต้องการตัวจำแนกประเภทที่มีประสิทธิภาพในตรวจจับคลาสบวกได้สูง (เช่น การจำแนกประเภทเนื้องอก) ควรเลือกใช้ Model 1 แต่หากงานจำแนกประเภทเป็นงานที่ต้องการลดความผิดพลาดของการทำนายผลเป็นบวก (เช่น การจำแนกประเภทเว็บเพจ) ควรเลือกใช้ Model 2

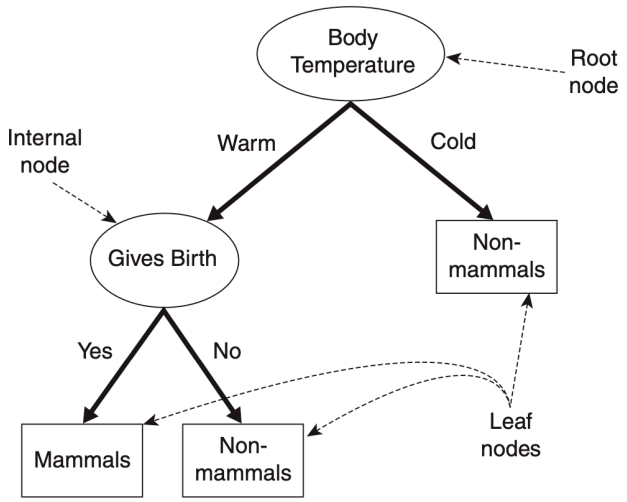
3.3 ตัวจำแนกประเภทต้นไม้ตัดสินใจ (Decision Tree Classifier)

ต้นไม้ตัดสินใจ (decision tree) เป็นตัวจำแนกประเภทที่ใช้โครงสร้างข้อมูลต้นไม้เพื่อบันทึกกฎเกณฑ์การจำแนกประเภทที่เรียนรู้ได้จากชุดข้อมูลฝึกฝน ต้นไม้ตัดสินใจประกอบด้วยโหนด 3 ชนิด คือ

- **root node จำนวนหนึ่งโหนด** คือโหนดที่ไม่มีลิงก์ขาเข้า (incoming links) และอาจไม่มีหรือมีลิงก์ขาออก (outgoing links) จำนวนเท่าใดก็ได้
- **Internal nodes** คือโหนดที่มีลิงก์ขาเข้าจำนวนหนึ่งลิงก์ และมีลิงก์ขาออกตั้งแต่สองลิงก์ขึ้นไป
- **Leaf nodes หรือ terminal nodes** คือโหนดที่มีลิงก์ขาเข้าจำนวนหนึ่งลิงก์ และไม่มีลิงก์ขาออกเลย

Vertebrate Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

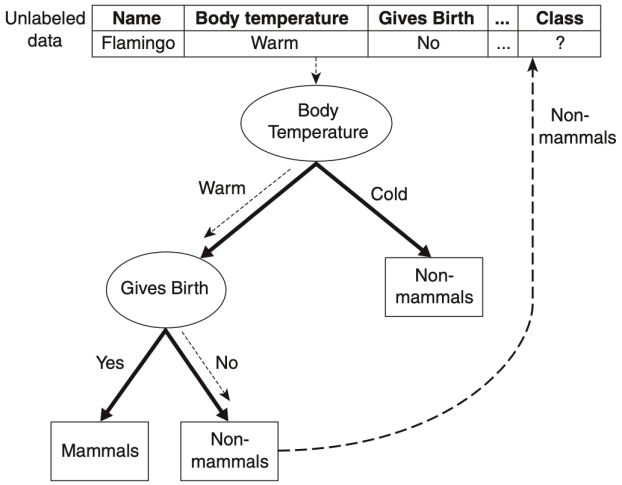
รูปที่ 3.2 Vertebrate dataset



รูปที่ 3.3 Decision tree classifier สำหรับการจำแนกประเภทสัตว์เลี้ยงลูกด้วยม (mammal classifier)

Leaf node แต่ละโหนดจะถูกเชื่อมโยงไว้กับคลาสเลเบิล ส่วน non-terminal nodes (root node และ internal nodes) จะถูกเชื่อมโยงไว้กับเงื่อนไขการทดสอบแอทริบิวต์ แต่ละผลลัพธ์ที่เป็นไปได้ของเงื่อนไขการทดสอบแอทริบิวต์จะสัมพันธ์กับโหนดลูก (child node) เพียงโหนดเดียวของโหนดนั้น ๆ เช่น ต้นไม้ตัดสินใจสำหรับจำแนกประเภทสัตว์เลี้ยงลูกด้วยนม (mammal decision tree classifier) ในรูปที่ 3.3 root node มีเงื่อนไขการทดสอบแอทริบิวต์ Body Temperature ที่มีผลลัพธ์ที่เป็นไปได้สองค่า คือ Warm และ Cold ซึ่งถูกเชื่อมโยงกับโหนดลูกทางด้านซ้ายและด้านขวาตามลำดับ โหนดในระดับถัดไปของต้นไม้ตัดสินใจมีสองโหนด โหนดแรกทางด้านซ้ายคือ internal node ที่มีเงื่อนไขการทดสอบแอทริบิวต์ Give Birth ที่มีผลลัพธ์ที่เป็นไปได้สองค่า คือ Yes และ No ซึ่งถูกเชื่อมโยงไปยัง leaf node ทางด้านซ้าย (คลาสเลเบิล Mammals) และ leaf node ทางด้านขวา (คลาสเลเบิล Non-mammals) ตามลำดับ ส่วนโหนดที่สองทางด้านขวาคือ leaf node ที่มีคลาสเลเบิลเป็น Non-mammals

การจำแนกประเภทโดยใช้ต้นไม้ตัดสินใจ เริ่มต้นจากการทดสอบเงื่อนไขแอทริบิวต์ของ root node แล้วตามลิงก์ขาออกไปยังโหนดลูกในระดับถัดไปที่เชื่อมโยงกับผลลัพธ์ของการทดสอบที่ได้ ถ้าโหนดลำดับถัดไปเป็น internal node ก็ทำการทดสอบเงื่อนไขแอทริบิวต์ของโหนดลูกนั้น แล้วตามลิงก์ขาออกไปยังโหนดลูกในระดับถัดไป แต่ถ้าโหนดลำดับถัดไปเป็น leaf node กระบวนการจำแนกประเภทจะจบลงโดยมีคำทำนายคือคลาสเลเบิลของ leaf node ที่ไปถึง ดังแสดงตัวอย่างในรูปที่ 3.4 เป็นกระบวนการจำแนกประเภทของข้อมูลไม่ทราบคลาสเลเบิลที่มีแอทริบิวต์ Body termperature เท่ากับ Warm และ Gives Birth เท่ากับ No โดยเริ่มจากการเงื่อนไขการทดสอบแอทริบิวต์ Body temperature ที่ root node ได้ผลลัพธ์เท่ากับ Warm ดังนั้นตัวจำแนกประเภทจะตามลิงก์ขาออกทางด้านซ้ายไปยังโหนดลูกในระดับถัดไปซึ่งเป็น internal node ที่มีเงื่อนไขการทดสอบแอทริบิวต์ Give Birth ที่ให้ผลลัพธ์เป็น No ทำให้ตัวจำแนกประเภทตามลิงก์ขาออกไปทางด้านขวาซึ่งนำไปสู่ leaf node ที่มีคลาสเลเบิลเท่ากับ Non-mammals เพราะฉะนั้นคำทำนายของตัวจำแนกประเภทสำหรับข้อมูลตัวอย่างนี้จึงมีค่าเท่ากับ Non-mammals



รูปที่ 3.4 ตัวอย่างวิธีการจำแนกประเภทของต้นไม้ตัดสินใจ

3.3.1 อัลกอริทึมพื้นฐานสำหรับสร้างต้นไม้ตัดสินใจ

อัลกอริทึมที่มีประสิทธิภาพสำหรับสร้างต้นไม้ตัดสินใจมีอยู่หลากหลายตัวในปัจจุบัน โดยส่วนใหญ่อัลกอริทึมเหล่านี้จะให้ผลลัพธ์เป็นต้นไม้ตัดสินใจที่ให้คำตอบได้ดีพอสมควร แต่อาจจะไม่ใช่โครงสร้างต้นไม้ที่ดีที่สุด (suboptimal decision tree) โดยรูปแบบในการสร้างต้นไม้ของอัลกอริทึมส่วนใหญ่จะใช้แนวทางการสร้างต้นไม้จากบนลงล่าง (top-down คือจาก root node ไปยัง leaf node) และใช้วิธีการขยายกิ่งแบบละโมบ (greedy strategy) กล่าวคือจะเลือกแอทริบิวต์เพื่อแบ่งชุดข้อมูลและขยายกิ่งต้นไม้ตัดสินใจโดยการตัดสินใจจากค่าเหมาะสมที่สุดเฉพาะที่ (locally optiomal decisions) หนึ่งในอัลกอริทึมสร้างต้นไม้ตัดสินใจที่ได้ถูกพัฒนาขึ้นในยุคแรกคือ อัลกอริทึมของฮันท์ (Hunt's algorithm) ซึ่งเป็นพื้นฐานสำหรับการพัฒนาอัลกอริทึมสร้างต้นไม้ตัดสินใจที่นิยมใช้กันในปัจจุบัน เช่น ID3, C4.5, CART อัลกอริทึมของฮันท์สรุปเป็น pseudo-code ได้ดังอัลกอริทึม 3.1 ตัวอย่างการสร้างต้นไม้ตัดสินใจสำหรับชุดข้อมูล vertebrate ในรูปที่ 3.2 แสดงดังรูปที่ 3.5

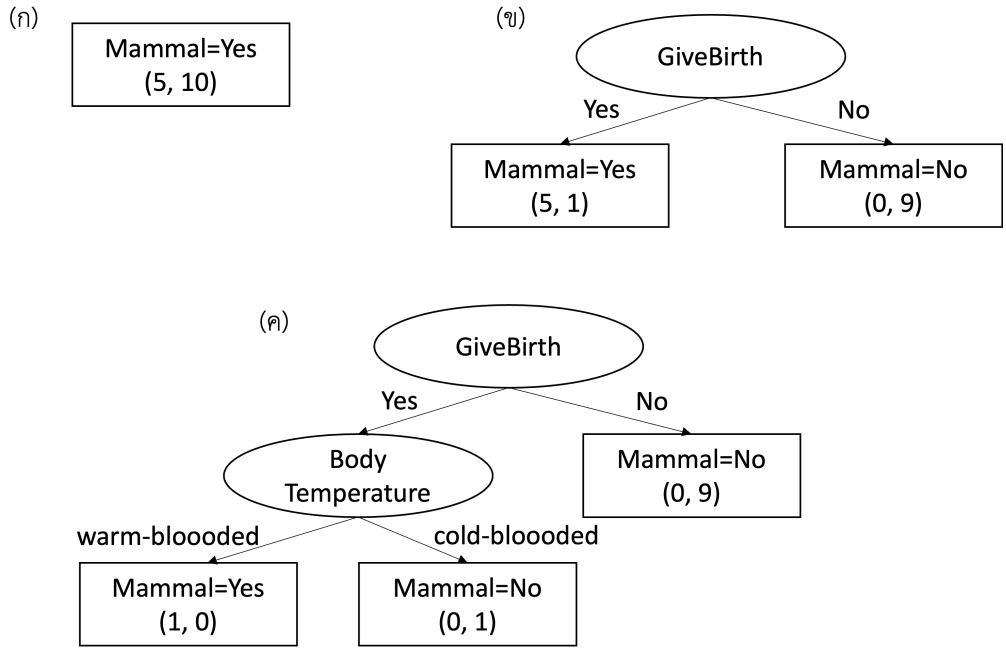
อัลกอริทึมของฮันท์เป็นรูปแบบทั่วไปสำหรับการสร้างต้นไม้ตัดสินใจโดยใช้เทคนิคการออกแบบอัลกอริทึมแบบละโมบ ในการเขียนโปรแกรมเพื่อนำอัลกอริทึมของฮันท์ไปใช้งานจริง ยังมีจุดที่จำเป็นต้องเพิ่มเติมให้สมบูรณ์สองส่วน คือ

(1) จะใช้อะไรเป็นเกณฑ์ในการคัดเลือกแอทริบิวต์สำหรับแตกกิ่ง (Splitting crriterion S ในอัลกอริทึม 3.1)

(2) จะใช้อะไรเป็นเงื่อนไขในการหยุดการขยายกิ่งต้นไม้ (stopping criterion) อัลกอริทึม 3.1 หยุดขยายกิ่งต้นไม้ตัดสินใจเมื่อเรคอร์ดที่สัมพันธ์กับโหนดเป็นคลาสเดียวกันทั้งหมด ในทางปฏิบัติเพื่อป้องกันปัญหา overfitting เรามักจำเป็นต้องหยุดการขยายกิ่งต้นไม้ก่อน (early termination) เงื่อนไขที่ใช้กำหนดให้หยุดการขยายกิ่งของโหนดนี้ เรียกว่า stopping condition

อัลกอริทึมที่ 3.1 Hunt's algorithm

BuildTree(Xt, Y, S):
อินพุต - Xt คือเซตเรคอร์ดข้อมูลฝึกฝนสำหรับโหนด t - Y={y1, ..., yc } คือ คลาสเลเบิล - S คือ เกณฑ์ในการคัดเลือกแอทริบิวต์สำหรับแตกกิ่งต้นไม้ (Splitting Criterion) **เอาต์พุต** - ต้นไม้ตัดสินใจ
ถ้าเรคอร์ดทุกเรคอร์ดใน Xt มีคลาสเลเบิลคือ yt เหมือนกันทั้งหมดทุกเรคอร์ด: *โหนด t จะเป็น leaf node ที่มีคลาสเลเบิลเท่ากับ yt* ***ถ้าไม่เช่นนั้น Xt ประกอบด้วยเรคอร์ดของคลาสมากกว่าหนึ่งคลาส:*** *เลือกเงื่อนไขทดสอบแอทริบิวต์ที่เหมาะสมที่สุดโดยใช้ฟังก์ชัน S เพื่อแบ่งเรคอร์ดใน Xt เป็นซับเซตที่มีขนาดเล็กลง* *สร้างโหนดลูกสำหรับผลลัพธ์ทุกผลลัพธ์ที่เป็นไปได้ของเงื่อนไขการทดสอบแอทริบิวต์* *เรียกฟังก์ชัน BuildTree แบบเวียนบังเกิดกับทุกโหนดลูกที่ถูกสร้างขึ้นมาใหม่*



รูปที่ 3.5 การสร้างต้นไม้ตัดสินใจด้วยอัลกอริทึมของฮันท์

3.3.2 หน่วยการวัดสำหรับการเลือกเงื่อนไขทดสอบแอทริบิวต์

ในการสร้างต้นไม้ตัดสินใจ เราต้องการแบ่งข้อมูลตัวอย่างแต่ละโหนดออกเป็นซับเซตใน child node ที่เป็นเนื้อเดียวกันมากขึ้น (คือข้อมูลส่วนใหญ่มีคลาสเลเบิลที่เหมือนกัน; more homogenous, purer) การมีโหนดในต้นไม้ที่ตัดสินใจที่เป็นเนื้อเดียวกันมากกว่าส่งผลดีเนื่องจากทำให้ต้นไม้ตัดสินใจที่ได้มีจำนวนกิ่งหรือความลึกลดลง ในทางตรงกันข้ามถ้าโหนดในต้นไม้ตัดสินใจมีความเป็นเนื้อเดียวกันต่ำ จะทำให้เกิดผลเสียคือต้นไม้จะมีขนาดใหญ่และมีความลึกสูงซึ่งทำให้ง่ายต่อการเกิด model overfitting ซึ่งส่งผลให้ประสิทธิภาพของการจำแนกประเภทลดลง นอกจากนี้ต้นไม้ตัดสินใจที่ใหญ่และซับซ้อนยังยากต่อการตีความ และใช้เวลาในการเทรนและการทดสอบมากกว่าต้นไม้ตัดสินใจที่มีขนาดเล็ก

การวัดความไม่บริสุทธิ์ของโหนดหนึ่งโหนดในต้นไม้ตัดสินใจ ทำได้หลายวิธี เช่น

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

$$Gini\ index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

$$Classification\ error = 1 - max_i p_i(t)$$

เมื่อ $p_i(t)$ คือความถี่สัมพันธ์ (relative frequency) ของตัวอย่างฝึกฝนของคลาส i ที่โหนด t, c คือจำนวนคลาสลาเบิลทั้งหมด และในการคำนวนค่า entropy 0 $\log_2$ 0 = 0 วิธีการวัดความไม่บริสุทธิ์ของโหนดทั้งสามวิธีจะให้ค่าไม่บริสุทธิ์เท่ากับศูนย์ เมื่อเรคอร์ดข้อมูลของโหนดเป็นคลาสเดียวกันทั้งหมด และค่าความไม่บริสุทธิ์จะสูงสุดเมื่อโหนดมีจำนวนเรคอร์ดของแต่ละคลาสเท่ากันทั้งหมด กราฟในรูปที่ 3.6 เปรียบเทียบค่าของ Entropy, Gini index, และ Classification error ณ ค่า $p_i(t)$ ต่าง ๆ ของโหนดต้นไม้ตัดสินใจสำหรับการจำแนกประเภทแบบไบนารี จะเห็นได้ว่า ค่า Entropy, Gini index, และ Classification Error จะมีค่าเท่ากับศูนย์เมื่อความถี่สัมพัทธ์มีค่าเท่ากับ 0 และ 1 ซึ่งจะเกิดเมื่อเรคอร์ดทุกเรคอร์ดของโหนดเป็นคลาสเดียวกันทั้งหมด ในขณะที่ค่า Entropy, Gini index, และ Classification error จะมีค่าสูงสุดเมื่อความถี่สัมพัทธ์มีค่าเท่ากับ 0.5 ซึ่งจะเกิดเมื่อโหนดมีจำนวนเรคอร์ดแต่ละคลาสเท่ากัน

ตัวอย่าง 3.2 จงคำนวณค่า Entropy, Gini index, และ Classification error ของโหนดแต่ละโหนดดังต่อไปนี้

(ก) Node 1: จำนวนเรคอร์ดของ Class=Yes เท่ากับ 0, จำนวนเรคอร์ดของ Class=No เท่ากับ 6

(ข) Node 2: จำนวนเรคอร์ดของ Class=Yes เท่ากับ 1, จำนวนเรคอร์ดของ Class=No เท่ากับ 5

(ค) Node 3: จำนวนเรคอร์ดของ Class=Yes เท่ากับ 3, จำนวนเรคอร์ดของ Class=No เท่ากับ 3

(ก)

Entropy = - (0/6) $\log_2$ (0/6) - (6/6) $\log_2$ (6/6) = 0

Gini = 1 - $(0/6)^2$ - $(6/6)^2$ = 0

Error = 1 - max(0/6, 6/6) = 1 - 1 = 0

(ข)

Entropy = = - (1/6) $\log_2$ (1/6) - (5/6) $\log_2$ (5/6) = 0.650

Gini = 1 - $(1/6)^2$ - $(5/6)^2$ = 0.278

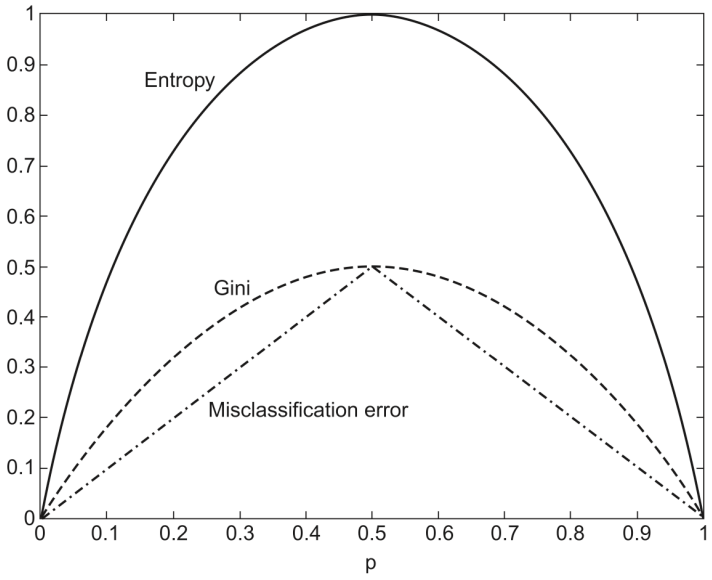
Error = 1 - max(1/6, 5/6) = 1 - 5/6 = 0.167

(ค)

Entropy = = - (3/6) $\log_2$ (3/6) - (3/6) $\log_2$ (3/6) = 1

Gini = 1 - $(3/6)^2$ - $(3/6)^2$ = 0.5

Error = 1 - max(3/6, 3/6) = 1 - 3/6 = 0.5



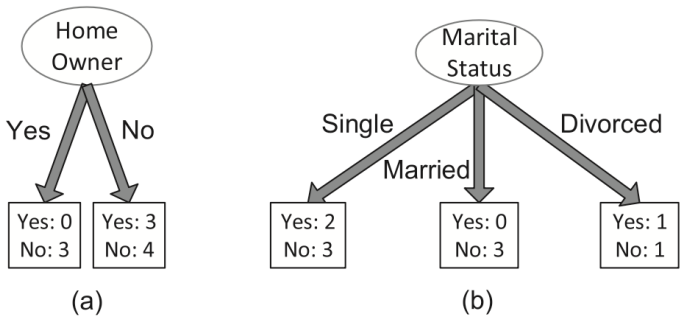
รูปที่ 3.6 ค่าของ Entropy, Gini Index, Misclassification Error สำหรับการจำแนกประเภทแบบไบนารี

การคำนวณความไม่บริสุทธิ์ของโหนดลูก

พิจารณาเงื่อนไขทดสอบแอทริบิวต์ที่แตกกิ่งต้นไม้ออกเป็นโหนดจำนวน k โหนด { v_1, v_2, ..., v_k } กำหนดให้ $N(v_j)$ คือจำนวนข้อมูลฝึกฝนที่สัมพันธ์อยู่กับโหนดลูก v_j ซึ่งมีค่าความไม่บริสุทธิ์เท่ากับ $I(v_j)$ แล้ว ความไม่บริสุทธิ์รวม (collective impurity) ของโหนดลูกทุกโหนดสามารถคำนวณได้จาก

$$I(children) = \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

ตัวอย่าง 3.3 จงคำนวณความไม่บริสุทธิ์ของโหนดลูกในแต่ละกรณีดังต่อไปนี้



(a) I (Home Owner = yes) = - (0/3) $\log_2$ (0/3) - (3/3) $\log_2$ (3/3) = 0

I (Home Owner = no) = - (3/7) $\log_2$ (3/7) - (4/7) $\log_2$ (4/7) = 0.985

I (Home Owner) = (3/10) x 0 + (7/10) x 0.985 = 0.690

(b) I (Marital Status = Single) = - (2/5) $\log_2$ (2/5) - (3/5) $\log_2$ (3/5) = 0.971

I (Home Owner = Married) = - (0/3) $\log_2$ (0/3) - (3/3) $\log_2$ (3/3) = 0

I (Home Owner = Divorced) = - (1/2) $\log_2$ (1/2) - (1/2) $\log_2$ (1/2) = 1.000

I (Marital Status) = (5/10) x 0.971 + (3/10) x 0 + (2/10) x 1 = 0.686

การเลือกเงื่อนไขทดสอบแอทริบิวต์ที่ดีที่สุด

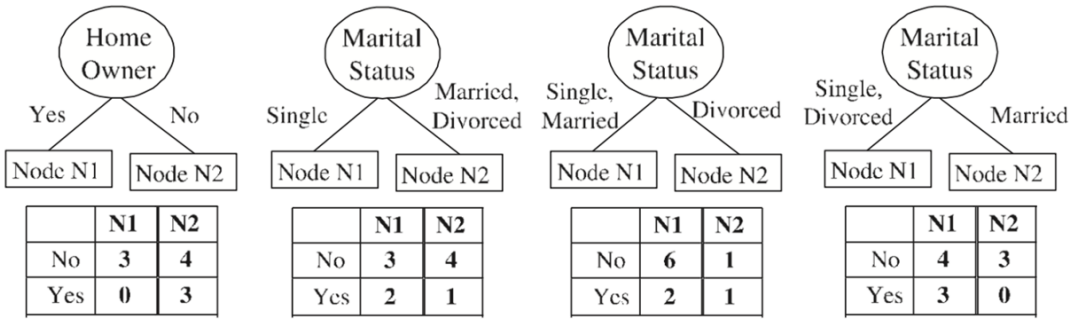
การเลือกเงื่อนไขที่ดีที่สุดสำหรับการทดสอบแอทริบิวต์ทำได้โดยหาผลต่างของระดับความไม่บริสุทธิ์ของโหนดก่อนการแตกกิ่ง (parent node) กับระดับความไม่บริสุทธิ์รวมของโหนดลูกที่เกิดจากการแตกกิ่ง ผลต่างนี้เรียกว่า **gain :**

$$\Delta = I(parent) - I(children)$$

ยิ่งค่า gain สูง ความบริสุทธิ์ของคลาสใน child node เมื่อเทียบกับ parent node ก็จะยิ่งสูงไปด้วย ดังนั้นจึงสรุปได้ว่า เกณฑ์การแตกกิ่ง (splitting criterion) ของอัลกอริทึมสร้างต้นไม้ตัดสินใจคือการเลือกเงื่อนไขทดสอบแอทริบิวต์ที่ให้ค่า gain สูงที่สุดนั่นเอง

ตัวอย่าง 3.4 จงคำนวณ gain ของเงื่อนไขทดสอบแอทริบิวต์แต่ละกรณีในรูปภาพข้างล่าง

	Parent
No	7
Yes	3



	N1	N2
No	3	4
Yes	0	3

	N1	N2
No	3	4
Yes	2	1

	N1	N2
No	6	1
Yes	2	1

	N1	N2
No	4	3
Yes	3	0

Entropy(Parent)

= - [(7/10 $\log_2$ (7/10) + (3/10) $\log_2$ (3/10)] = 0.8813

Entropy(Home Owner)

= (3/10)[-3/3 $\log_2$ (3/3) -0/3 $\log_2$ (0/3)] + (7/10)[-4/7 $\log_2$ (4/7) - 3/7 $\log_2$ (3/7)]

= (3/10)(0) + (7/10)(0.985)

= 0.6895

Entropy(Marital Status_single)

= (5/10)[-3/5 $\log_2$ (3/5) - 2/5 $\log_2$ (2/5)] + (5/10)[-4/5 $\log_2$ (4/5) - 1/5 $\log_2$ (1/5)]

= 0.5 * 0.9710 + 0.5 * 0.7220

= 0.8465

Entropy(Marital Status_single, married)

= (8/10)[-6/8 $\log_2$ (6/8) - 2/8 $\log_2$ (2/8)] + (2/10)[-1/2 $\log_2$ (1/2) - 1/2 $\log_2$ (1/2)]

= 0.8 * 0.8113 + 0.2 * 1.0

= 0.8490

Entropy(Marital Status_single, divorced)

= (7/10)[-4/7 $\log_2$ (4/7) - 3/7 $\log_2$ (3/7)] + (3/10)[-3/3 $\log_2$ 3/3 - 0/3 $\log_2$ 0/3]

= 0.7 * 0.9852 + 0.3 * 0

= 0.6896

Gain (Home Owner) = Entropy(Parent) - Entropy(Home Owner)

= 0.8813 - 0.6895 = **0.1918**

Gain (Marital Status_single) = Entropy(Parent) - Entropy(Marital Status_single)

= 0.8813 - 0.8465 = **0.0348**

Gain (Marital Status_single, married) = Entropy(Parent) - Entropy(Marital Status_single, married)

= 0.8813 - 0.8490 = **0.0323**

Gain (Marital Status_single, divorced) = Entropy(Parent) - Entropy(Marital Status_single, divorced)

= 0.8813 - 0.6896 = **0.1917**

ปัญหาที่มักเกิดขึ้นกับการใช้ entropy และ Gini Index ในการคำนวณ information gain คือ การที่ entropy และ Gini index มักจะเป็นผลดีกับแอทริบิวต์ที่มีจำนวนค่าที่เป็นไปได้จำนวนมาก เช่น รหัสประจำตัว การแก้ปัญหานี้ อาจทำได้โดยการใช้ Gain ratio แทนการใช้ information gain โดย Gain ratio นี้จะคำนึงถึงจำนวน split ทั้งหมดด้วย

$$Gain\ ratio = \frac{\Delta}{split\ info} = \frac{Entropy(parent) - \sum_{i=1}^{k} \frac{N(v_i)}{N} Entropy(v_i)}{-\sum_{i=1}^{k} \frac{N(v_i)}{N} log_2 \frac{N(v_i)}{N}}$$

ตัวอย่าง 3.5 จงคำนวณ gain ของเงื่อนไขทดสอบแอทริบิวต์แต่ละกรณีในตัวอย่างที่ 3.4

Gain ratio (Home Owner)

= Gain (Home Owner) / (-[3/10 log (3/10) + 7/10 log (7/10)])

= 0.1918/0.8813 = 0.2176

Gain ratio (Marital Status_single)

= Gain (Marital Status_single) / (-[5/10 log (5/10) + 5/10 log (5/10)])

= 0.0348 / 1.0 = 0.0348

Gain ratio (Marital Status_single, married)

= Gain (Marital Status_single, married) / (-[8/10 log (8/10) + 2/10 log (2/10)])

= 0.0323 / 0.7219 = 0.0447

Gain ratio (Marital Status_single, divorced)

= Gain (Marital Status_single, divorced) / (-[7/10 log (7/10) + 3/10 log (3/10)])

= 0.1917 / 0.8813 = 0.2175

3.3.3 อัลกอริทึมสำหรับการอุปมานต้นไม้นตัดสินใจ

ดังที่อธิบายไปแล้ว อัลกอริทึมของฮันท์เป็นเพียงแนวทางในการสร้างต้นไม้ตัดสินใจ การนำอัลกอริทึมไปเขียนเป็นโปรแกรมจะต้องมีการกำหนดรายละเอียดเกี่ยวกับเงื่อนไขการหยุดและวิธีการเลือกเงื่อนไขการทดสอบแอทริบิวต์ที่ดีที่สุดให้ชัดเจน เช่น เงื่อนไขการหยุดอัลกอริทึมอาจใช้การตรวจสอบจำนวนเรคอร์ดที่สัมพันธ์กับโหนดปัจจุบันเป็นเกณฑ์ โดยหากจำนวนเรคอร์ดลดลงน้อยกว่าจำนวนที่กำหนดก็ให้หยุดอัลกอริทึม สำหรับเงื่อนไขการทดสอบแอทริบิวต์ที่ดีที่สุดอาจเลือกใช้ information gain หรือ gain ratio เป็นเกณฑ์ อัลกอริทึมสำหรับการสร้างต้นไม้ตัดสินใจที่ได้รับการปรับปรุงจากอัลกอริทึมของฮันท์ และมีรายละเอียดเพียงพอต่อการนำไปเขียนโปรแกรมแสดงดังอัลกอริทึมที่ 3.2 ซึ่งมีฟังก์ชันที่สำคัญดังนี้คือ

1. createNode() สร้างโหนดของต้นไม้ตัดสินใจ โดยโหนดจะประกอบด้วยฟิลด์คือ node.test_cond (สำหรับโหนดที่เป็น internal node หรือ node.label (สำหรับโหนดที่เป็น leaf node)

2. find_best_split() เป็นฟังก์ชันที่เลือกแอทริบิวต์ที่ดีที่สุดสำหรับใช้ในการแตกกิ่งต้นไม้ โดยการคำนวณหาค่า information gain ของแอทริบิวต์แต่ละตัว แล้วเลือกแอทริบิวต์ที่ให้ค่า information gain สูงที่สุด

3. Classify() เป็นฟังก์ชันที่กำหนดคลาสเลเบิลให้กับ leaf node ของต้นไม้ตัดสินใจ โดยใช้คลาสเลเบิลที่ปรากฏในชุดข้อมูลเป็นจำนวนมากที่สุด

4. stopping_cond() เป็นฟังก์ชันที่ทดสอบเงื่อนไขการหยุดสร้างต้นไม้ เช่น ทดสอบจำนวนเรคอร์ดข้อมูลที่เหลืออยู่ หรือ ทดสอบจากระดับความลึกของต้นไม้ เป็นต้น

อัลกอริทึมที่ 3.2 Decision Tree Induction Algorithm

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

3.3.4 คุณสมบัติที่สำคัญของตัวจำแนกประเภทต้นไม้ตัดสินใจ

1. Applicability and Interpretability. การสร้างต้นไม้ตัดสินใจไม่จำเป็นต้องมีการกำหนดสมมติฐานเกี่ยวกับการแจกแจงความน่าจะเป็น (probability distribution) ของข้อมูล ดังนั้นต้นไม้ตัดสินใจจึงสามารถนำไปใช้ได้กับชุดข้อมูลที่หลากหลาย ต้นไม้ตัดสินใจยังใช้ได้กับแอทริบิวต์ทั้งแบบ categorical และแบบ contiunous และใช้กับปัญหาแบบ multiclass ได้โดยไม่จำเป็นต้องแปลงให้อยู่ในรูปของปัญหาแบบ binary class หลายปัญหา นอกจากนี้ต้นไม้นตัดสินใจยังสามารถตีความได้ง่าย และสำหรับชุดข้อมูลขนาดไม่ใหญ่มากประสิทธิภาพการทำนายของต้นไม้ตัดสินใจก็ใกล้เคียงกับเทคนิคการจำแนกประเภทอื่น ๆ ที่ซับซ้อนมากกว่า

2. Computational Efficiency. แนวทางการสร้างต้นไม้แบบละโมบ จากบนลงล่าง และแบ่งแบบเวียนบังเกิด (greedy, top-down, recursive partitioning strategy) ช่วยให้ได้ต้นไม้ตัดสินใจที่มีประสิทธิภาพดีแม้ว่าชุดข้อมูลจะมีขนาดใหญ่มาก และการจำแนกประเภทโดยใช้ต้นไม้ตัดสินใจก็ตทำได้อย่างรวดเร็วมาก โดยมี worst-case complexity เท่ากับ O(w) เมื่อ w คือความลึกสูงสุดของต้นไม้

3. Choice of Impurity Measure. ตัวเลือกของวิธีการวัดความบริสุทธิ์ มีผลกระทบไม่มากต่อประสิทธิภาพของต้นไม้ตัดสินใจที่ได้ เนื่องจากวิธีการวัดความบริสุทธิ์ส่วนใหญ่มีคุณสมบัติที่สอดคล้องกัน

นอกจากนี้ ยังมีจุดแข็งอื่น ๆ เช่น จัดการกับแอทริบิวต์ที่ไม่เกี่ยวข้อง แอทริบิวต์ที่ซ้ำซ้อน และข้อมูลรบกวน (noise) ได้ดี สามารถปรับปรุงให้ใช้กับเงื่อนไขที่ประกอบด้วยแอทริบิวต์มากกว่าหนึ่งแอทริบิวต์ได้ เป็นต้น สำหรับจุดอ่อนที่สำคัญของ

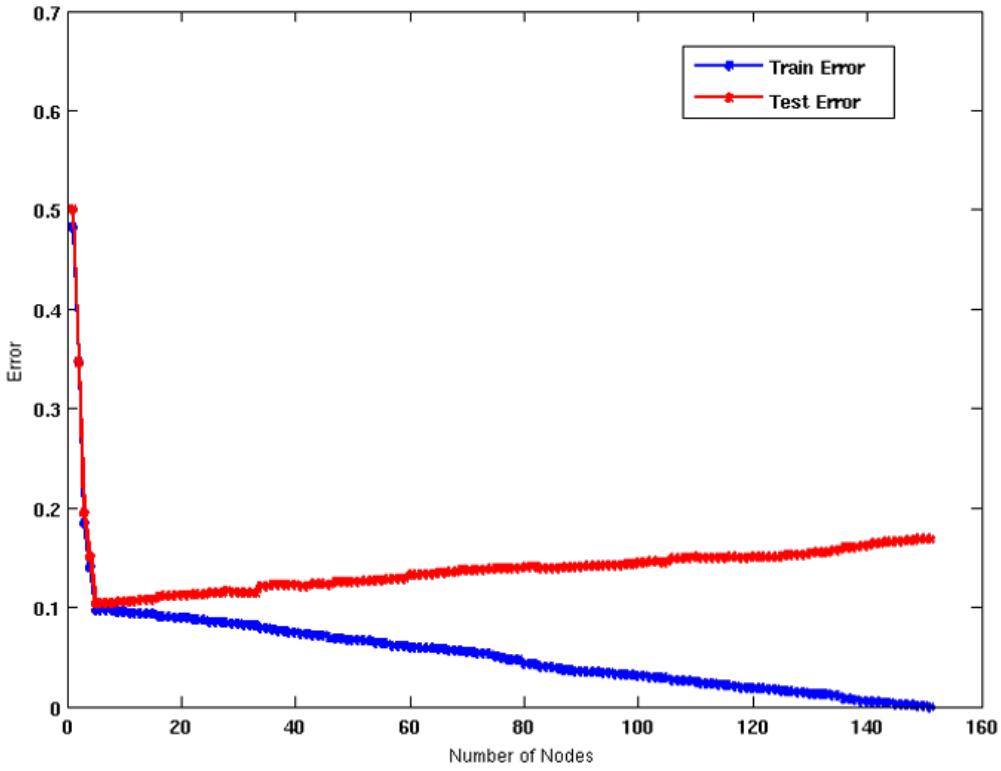
ต้นไม้ตัดสินใจคือ การเลือกเงื่อนไขทดสอบแอทริบิวต์ที่ใช้ได้เพียงแอทริบิวต์เดียว และมีแนวทางการเลือกแบบละโมบ อาจทำให้แอทริบิวต์ที่มีปฏิสัมพันธ์กัน (interacting attributes คือแอทริบิวต์ตั้งแต่สองแอทริบิวต์ที่ต้องใช้ร่วมกันในการจำแนกคลาส) ถูกละเลยไปได้

3.4 Model Overfitting

ความผิดพลาดสามารถแบ่งได้เป็น (1) **training error** คือความผิดพลาดที่เกิดขึ้นบนชุดข้อมูลฝึกฝน, (2) **test error** คือความผิดพลาดที่เกิดขึ้นบนชุดข้อมูลทดสอบ, (3) **genralization error** คือความผิดพลาดคาดหวัง (expected error) เมื่อนำโมเดลไปใช้กับข้อมูลที่สุ่มมาจากการแจกแจงความน่าจะเป็นที่เหมือนกันกับการแจกแจงความน่าจะเป็นในชุดฝึกฝนและชุดทดสอบ

Model underfitting เกิดเมื่อโมเดลมีความซับซ้อนไม่พอที่จะเรียนรู้รูปแบบที่แฝงอยู่ในข้อมูลได้ สามารถสังเกตได้จากการที่ทั้ง training error และ test error มีค่าสูง

Model overfitting เกิดเมื่อโมเดลมีความซับซ้อนมากเกินไปทำให้เรียนรู้รูปแบบปลอม (spurious patterns) แทนที่จะเป็นรูปแบบที่แท้จริงตามธรรมชาติของข้อมูล สามารถสังเกตได้จากการที่ training error มีค่าต่ำหรือลดลงในขณะที่ test error มีค่าสูงหรือเพิ่มขึ้น เช่นตัวอย่างในรูปที่ 3.7



รูปที่ 3.7 ค่าความผิดพลาดของต้นไม้ตัดสินใจที่มีขนาดต่าง ๆ กัน ตั้งแต่จำนวน 0-150 โหนด เมื่อต้นไม้ตัดสินใจมีความซับซ้อนไม่มากนัก (ขนาด 0-9 โหนด) ทั้ง training error และ test error จะลดลงอย่างรวดเร็ว แต่เมื่อต้นไม้ตัดสินใจมีความซับซ้อนมากเกินไป (จำนวนโหนดตั้งแต่ 10 โหนดเป็นต้นไป) จะเกิด model overfitting ขึ้น เนื่องจาก test errors มีค่าสูงขึ้นเรื่อย ๆ ในขณะที่ training errors มีค่าลดลงจนเป็นศูนย์

สาเหตุของ model overfitting ส่วนใหญ่เกิดจากการที่มีจำนวนข้อมูลในชุดฝึกฝนน้อยเกินไป และการที่โมเดลมีความซับซ้อน (จำนวนพารามิเตอร์ที่ต้องเรียนรู้) มากเกินไป ดังนั้น เพื่อป้องกันไม่ให้เกิด model overfitting ในกรณีของต้นไม้ตัดสินใจ อาจทำได้โดยการเพิ่มขนาดชุดข้อมูลฝึกฝน หรือการจำกัดความซับซ้อนของต้นไม้ตัดสินใจโดยการจำกัดความลึกของต้นไม้ การหยุดการสร้างต้นไม้เมื่อค่า information gain ลดลงต่ำกว่าค่าที่กำหนด เป็นต้น

3.5 การคัดเลือกโมเดล (Model Selection)

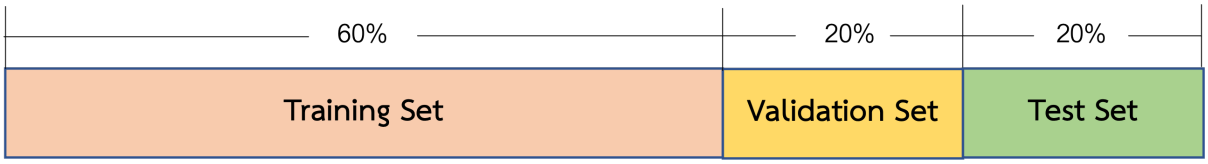
ในระหว่างการสร้างโมเดลการจำแนกประเภท เช่น ต้นไม้ตัดสินใจ มีโมเดลที่เป็นไปได้จำนวนมาก แต่ละโมเดลมีความซับซ้อนของโมเดล (model complexity) ที่แตกต่างกัน เป้าหมายของการสร้างโมเดลคือโมเดลที่มีอัตราความผิดพลาดนัยทั่วไปต่ำที่สุด (generalization error rates คืออัตราความผิดพลาดเมื่อนำโมเดลไปใช้กับข้อมูลอินพุตที่นอกเหนือไปจากข้อมูลที่ใช้ในการสร้างโมเดล) กระบวนการในการคัดเลือกโมเดลที่มีระดับความซับซ้อนที่เหมาะสม (ไม่มากและไม่น้อยเกินไป) และคาดว่าจะสามารถจำแนกประเภทได้อย่างมีประสิทธิภาพบน**ข้อมูลทดสอบ (test instances)** ที่นอกเหนือจากข้อมูลฝึกฝน เรียกว่า การคัดเลือกโมเดล (model selction)

หลักการคัดเลือกโมเดลก็คือการประมาณค่า genralization errors ให้ใกล้เคียงกับความเป็นจริงที่สุด ซึ่งมีวิธีการที่นิยมใช้ 2 วิธีการ คือ การใช้ชุดตรวจสอบความถูกต้อง (validation set) และการรวมค่าความซับซ้อนของโมเดลเข้าไปในการคำนวณอัตราความผิดพลาด

3.5.1 การใช้ชุดตรวจสอบความถูกต้อง (validation set)

การใช้ชุดตรวจสอบความถูกต้องเพื่อคัดเลือกโมเดล เป็นเทคนิคที่ช่วยประมาณค่า generalization errors ได้วิธีหนึ่ง ทำได้โดยการแบ่งส่วนหนึ่งของชุดข้อมูลฝึกฝนออกมาเป็น validation set เพื่อเปรียบเทียบอัตราความผิดพลาดของโมเดลแต่ละตัว โมเดลที่มีอัตราความผิดพลาดบน validation set ต่ำที่สุดจะเป็นโมเดลที่ถูกเลือก ตัวอย่างการแบ่งชุดข้อมูลในทางปฏิบัติแสดงดังแผนภาพในรูปที่ 3.8 ข้อมูลทั้งหมดจะถูกแบ่งเป็น 3 ส่วนคือ ชุดฝึกฝน (training set) 60% ชุดตรวจสอบความถูกต้อง (validation set) 20% และชุดทดสอบ 20%

แม้ว่าการใช้ validation set จะช่วยให้การคัดเลือกโมเดลทำได้อย่างยุติธรรมขึ้น แต่มีข้อควรระวังคือ วิธีการนี้จะอ่อนไหวต่อขนาดของ training set และ validation set หากมีขนาดเล็กเกินไป จะทำให้อัตราความผิดพลาดขาดความน่าเชื่อถือได้



รูปที่ 3.8 การแบ่งชุดข้อมูลออกเป็น training set, validation set, test set สำหรับการคัดเลือกโมเดล (model selection)

3.5.2 การรวมความซับซ้อนของโมเดลเข้าไว้ในการคำนวณอัตราความผิดพลาด

จากหลักการ **Occam's razor** หรือ หลักของความตระหนี่ (**Principle of Parsimony**) ที่กล่าวไว้ว่า ถ้ามีโมเดลสองโมเดลที่มีอัตราความผิดพลาดเท่าๆ กัน ควรเลือกใช้โมเดลที่มีความซับซ้อนน้อยกว่า ดังนั้นการเลือกโมเดลควรคำนึงถึงทั้งขนาดของอัตราความผิดพลาดและความซับซ้อนของโมเดล ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

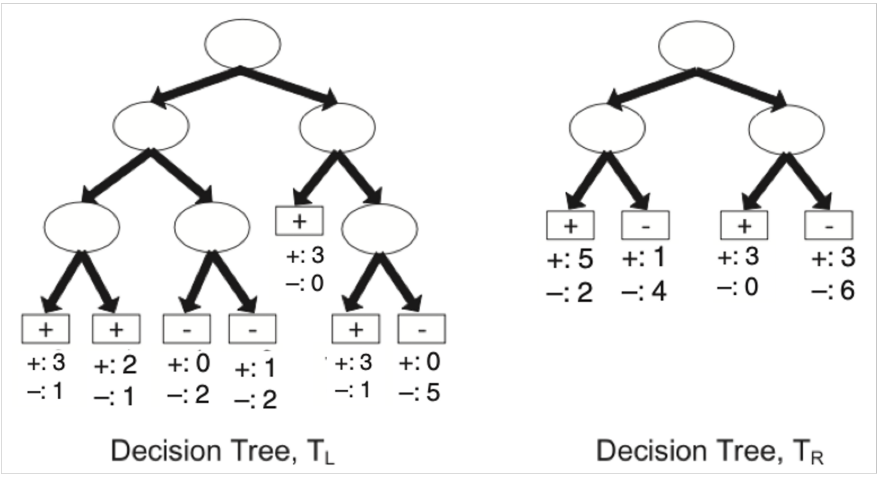
$$gen.error(m) = train.error(m, D.train) + \alpha \; x \; complexity(M)$$

เมื่อ **gen.error(m)** คือ generalization error ของโมเดล m, **train.error(m, D.train)** คือ training error ของโมเดล m บนชุดข้อมูลทดสอบ D.train, α คือไฮเปอร์พารามิเตอร์ (hyper-parameter) สำหรับปรับระดับความสำคัญของความซับซ้อนของโมเดล (ค่า α สูง จะหมายถึงความสำคัญของความซับซ้อนของโมเดลต่อ generalization error ที่มีมากขึ้น) และ complexity(M) คือความซับซ้อนของโมเดลชนิด M เช่น หากต้องการคำนวณหา generalization error สำหรับต้นไม้ตัดสินใจ T เราสามารถคำนวณได้โดยสมการต่อไปนี้

$$gen.error(T) = train.error(T, D.train) + \Omega \frac{k}{N_{train}}$$

เมื่อ Ω คือ ต้นทุนสัมพัทธ์ (relative cost) ที่เกิดจากการเพิ่ม leaf node เข้าในต้นไม้ตัดสินใจเทียบกับอัตราความผิดพลาดบนชุดฝึกฝนที่จะเพิ่มขึ้น, k คือ จำนวน leaf nodes บนต้นไม้ตัดสินใจ และ N_{train} คือจำนวนตัวอย่างในชุดฝึกฝน

ตัวอย่าง 3.6 กำหนดต้นไม้ตัดสินใจ T_L และ T_R สองต้นดังรูป จงคำนวณหา generalization error ของต้นไม้ทั้งสอง เมื่อกำหนดให้ค่าต้นทุนสัมพัทธ์ของการเพิ่ม leaf node เมื่อเทียบกับการเพิ่มขึ้นของอัตราผิดพลาดบนชุดฝึกฝนมีค่าเท่ากับ (ก) 0.5 และ (ข) 1 ตามลำดับ



train.error(T_L) = 4/24 = 0.1667, k = 7/24 = 0.2917, train.error(T_R) = 6/24 = 0.25, k = 4/24 = 0.1667

(ก) **gen.error(T_L)** = 0.1667 + 0.5 x 0.2917 = **0.3126** , **gen.error(T_R)** = 0.25 + 0.5 x 0.1667 = **0.3334**

ดังนั้นในกรณีนี้ เราควรเลือกโมเดล T_L เนื่องจากมีค่า generalization error ต่ำกว่า

(ข) **gen.error(T_L)** = 0.1667 + 1 x 0.2917 = **0.4584** , **gen.error(T_R)** = 0.25 + 1 x 0.1667 = **0.4167**

ดังนั้นในกรณีนี้ เราควรเลือกโมเดล T_R เนื่องจากมีค่า generalization error ต่ำกว่า

การใช้หลักการคำอธิบายที่สั้นที่สุด (Minimum Description Length Principle)

หลักการคำอธิบายที่สั้นที่สุด เป็นวิธีการเชิงทฤษฎีสารสนเทศ (information theoretic approach) ซึ่งประมาณการค่าความซับซ้อนของโมเดลจากจำนวนบิตที่จำเป็นต้องใช้ในการส่งข้อมูลเกี่ยวกับโมเดลจากผู้ส่งไปยังผู้รับ (Total Description Length) ค่า total description length ของโมเดล M เมื่อกำหนดข้อมูล D ให้ คำนวณได้จากสมการคือ

$$Cost(M, D) = Cost(D|M) + \alpha \ x \ Cost(M)$$

เมื่อ Cost(D|M) คือ จำนวนบิตที่ต้องใช้ในการเข้ารหัสข้อมูลที่โมเดล M ทำนายผิดพลาด ส่วน Cost(M) คือจำนวนบิตที่ต้องใช้ในการเข้ารหัสโมเดล M และไฮเปอร์พารามิเตอร์ α คือค่าถ่วงน้ำหนักที่ใช้กำหนดผลกระทบของ Cost(D|M) และ Cost(M) ที่มีต่อค่า MDL

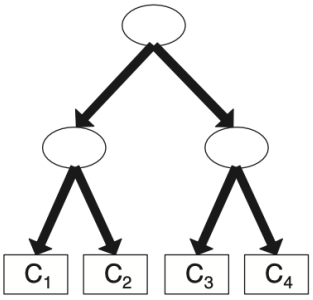
ตัวอย่าง 3.7 กำหนดต้นไม้ตัดสินใจ 2 ต้นดังรูป โดยชุดข้อมูลที่ใช้สร้างต้นไม้ตัดสินใจประกอบด้วยจำนวนแอทริบิวต์ไบนารีทั้งหมด 32 แอทริบิวต์ และคลาสจำนวน 4 คลาส จงคำนวณค่า total description length ของต้นไม้ตัดสินใจแต่ละต้นเพื่อคัดเลือกต้นไม้ตัดสินใจที่ดีกว่าตามหลักการคำอธิบายที่สั้นที่สุด (MDL) เมื่อ

- แต่ละ internal node ของต้นไม้ตัดสินใจเข้ารหัสโดยใช้รหัสของแอทริบิวต์ ถ้ามีแอทริบิวต์จำนวน m แอทริบิวต์ cost ของการเข้ารหัสของแต่ละแอทริบิวต์จะเท่ากับ log m
- แต่ละ leaf node เข้ารหัสโดยใช้รหัสของคลาส ถ้ามีจำนวนคลาสทั้งหมด k คลาส cost ของการเข้ารหัสคลาสจะเท่ากับ log k
- Cost(tree) เท่ากับ cost ของการเข้ารหัสทุกโหนดในต้นไม้ตัดสินใจ tree
- Cost(D|tree) ถูกเข้ารหัสโดยใช้จำนวนความผิดพลาดของการจำแนกประเภทของต้นไม้ตัดสินใจ แต่ละความผิดพลาดจะต้องใช้บิตจำนวน log n บิต เมื่อ n คือจำนวนข้อมูลในชุดฝึกฝน

Cost(internal node) = log m = log 32 = 5

Cost(leaf node) = log k = log 4 = 2

(ก) ต้นไม้ตัดสินใจ tree1 จำนวนครั้งที่ผิดพลาดเท่ากับ 5

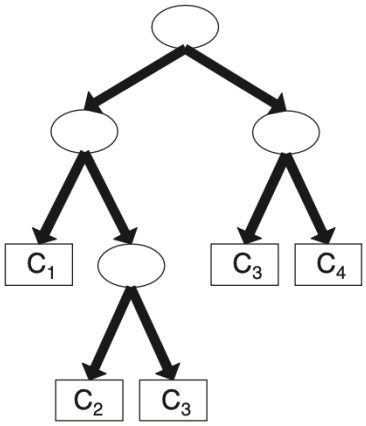


Cost(tree1) = 3 * 5 + 4 * 2 = 23

Cost(D | tree1) = 5 * log n

Cost(tree1, D) = 5log n + 23

(ข) ต้นไม้ตัดสินใจ tree2 จำนวนครั้งที่ผิดพลาดเท่ากับ 4



Cost(tree2) = 4 * 5 + 5 * 2 = 30

Cost(D | tree2) = 4 * log n

Cost(tree2, D) = 4log n + 30

ต้นไม้ตัดสินใจ tree1 มีค่า total description length ต่ำกว่า เมื่อ n <= 128 ดังนั้นถ้าจำนวนชุดข้อมูลฝึกฝนมีจำนวนน้อยกว่า 128 เราควรเลือก โมเดล tree1 แต่ถ้าจำนวนชุดข้อมูลฝึกฝนมีมากกว่า 128 เราควรเลือก โมเดล tree2 #

วิธีการคัดเลือกโมเดลที่ใช้สำหรับต้นไม้ตัดสินใจโดยเฉพาะ ที่นิยมใช้ในทางปฏิบัติ ได้แก่ การตัดแต่งกิ่งต้นไม้ก่อนเติบโตเต็มที่ (Prepruning หรือ early stopping rule) และการตัดแต่งกิ่งต้นไม้หลังจากเติบโตเต็มที่ (Post-pruning)

3.6 การประเมินประสิทธิภาพของโมเดล (Model Evaluation)

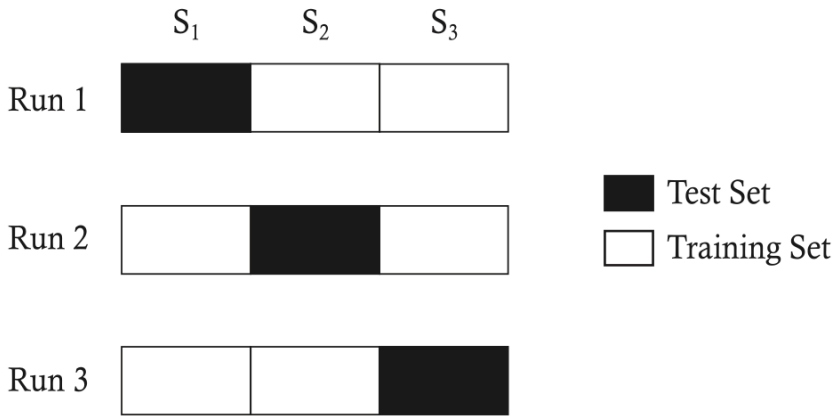
การประเมินประสิทธิภาพของโมเดล (model evaluation) คือการประเมินประสิทธิภาพโดยนัยทั่วไป (generalization performance) ของโมเดลบนชุดข้อมูลทดสอบที่ไม่ซ้ำซ้อนกันกับชุดข้อมูลฝึกฝน (training set) และชุดข้อมูลตรวจสอบ (validation) วิธีการประเมินประสิทธิภาพของโมเดลที่นิยมใช้มี 2 วิธี คือ Holdout method และ cross-validation method

3.6.1 Holdout Method

วิธี Holdout method ชุดข้อมูลที่มีป้ายกำกับ จะถูกแบ่งออกเป็นสองส่วนที่ไม่ซ้ำซ้อนกันคือ ชุดฝึกฝน (เลือกตัวอย่างแบบสุ่มประมาณ 2/3 ส่วนของข้อมูลทั้งหมด) และชุดทดสอบ (เลือกตัวอย่างแบบสุ่มประมาณ 1/3 ส่วนของข้อมูลทั้งหมด) โมเดลจำแนกประเภทจะถูกสร้างขึ้นโดยการเรียนรู้จากชุดข้อมูลฝึกฝน ด้วยวิธีการคัดเลือกโมเดล เช่น การใช้ชุดตรวจสอบ (validation set) ดังที่อธิบายในหัวข้อที่ 3.5 จากนั้นโมเดลที่ผ่านการคัดเลือกจะถูกนำมาทดสอบประสิทธิภาพโดยนัยทั่วไปบนชุดข้อมูลทดสอบ จำนวนข้อมูล สัดส่วนของข้อมูล และการสุ่มข้อมูลเพื่อเลือกชุดฝึกฝนและชุดทดสอบ มีผลต่อประสิทธิภาพของโมเดลที่ได้ หากข้อมูลที่ใช้ฝึกฝนมีจำนวนน้อยเกินไปจะทำให้โมเดลที่ได้มีประสิทธิภาพต่ำ หากข้อมูลในชุดทดสอบมีน้อยเกินไปจะทำให้ประสิทธิภาพโดยนัยทั่วไปที่วัดได้ขาดความน่าเชื่อถือ นอกจากนี้ประสิทธิภาพบนชุดข้อมูลทดสอบยังอาจมีความแปรปรวนสูงซึ่งเป็นผลจากการเลือกข้อมูลด้วยการสุ่ม ในกรณีนี้ผู้วิเคราะห์ข้อมูลอาจทำการประเมินประสิทธภาพซ้ำหลายๆ ครั้ง (repated holdout method) เพื่อให้สามารถเข้าใจความแปรปรวนของประสิทธิภาพบนชุดทดสอบและสรุปผลการประเมินประสิทธิภาพได้อย่างน่าเชื่อถือมากขึ้น

3.6.2 k-fold Cross-Validation

cross-validation เป็นวิธีการประเมินประสิทธิภาพของโมเดลที่มีเป้าหมายเพื่อการใช้ประโยชน์จากข้อมูลที่มีป้ายกำกับที่มีอยู่อย่างมีประสิทธิผลมากที่สุด การทำ k-fold cross-validation สามารถอธิบายได้ดังรูปที่ 3.9



รูปที่ 3.9 การประเมินประสิทธิภาพโมเดลด้วยวิธีการ 3-fold cross-validation

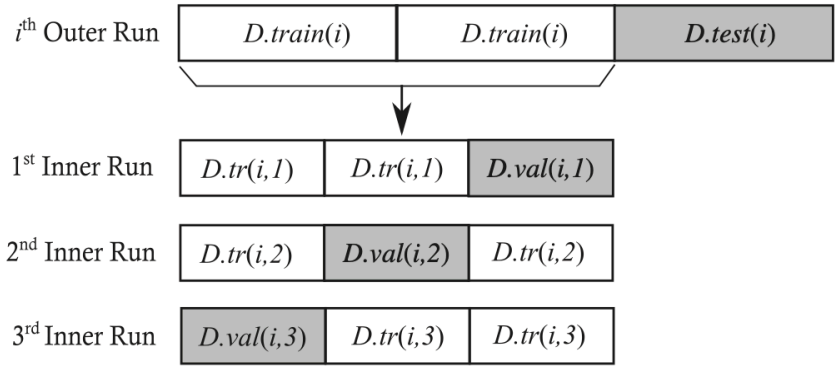
จากรูปที่ 3.9 เป็นตัวอย่างการทำ k-fold cross validation โดยกำหนดค่า k=3 ซึ่งอธิบายได้ดังนี้คือ ขั้นแรกแบ่งชุดข้อมูลออกเป็น k=3 ส่วน (3 folds) ที่มีขนาดเท่ากัน โดยวิธีการสุ่ม เรียกแต่ละซับเซตที่ได้ว่า S_1, S_2, S_3 จากนั้นให้ทำการเรียนรู้และทดสอบโมเดลบน k=3 ซับเซต จำนวน k=3 รอบ โดยที่ในแต่ละรอบให้กันข้อมูลไว้หนึ่งซับเซตเพื่อใช้สำหรับทดสอบโมเดล ดังรูปตัวอย่างซึ่งจะเห็นได้ว่า ในรอบที่ 1 (Run 1) เราเทรนโมเดลบนซับเซต S_2, S_3 และใช้ซับเซต S_1 เป็นชุดข้อมูลทดสอบซึ่งจะได้อัตราความผิดพลาด err(S_1) จากนั้น ในรอบที่ 2 (Run 2) เราเทรนโมเดลบนซับเซต S_1, S_3 และใช้ซับเซต S_2 เป็นชุดข้อมูลทดสอบซึ่งจะได้อัตราความผิดพลาด err(S_2) และรอบสุดท้ายรอบที่ 3 (Run 3) เราเทรนโมเดลบนซับเซต S_1, S_2 และใช้ซับเซต S_3 เป็นชุดข้อมูลทดสอบซึ่งจะได้อัตราความผิดพลาด err(S_3) อัตราความผิดพลาดรวมของโมเดลจะเท่ากับค่าเฉลี่ยของอัตราความผิดพลาดทั้งสามค่า ได้แก่ err(S_1) err(S_2) err(S_3)

การเลือกจำนวน fold หรือค่า k ของ k-fold cross-validation จะมีผลต่อประสิทธิภาพโดยนัยทั่วไปที่ได้ กล่าวคือ หากค่า k มีค่าน้อย จะทำให้จำนวนข้อมูลที่ใช้เทรนโมเดลมีน้อยซึ่งจะส่งผลให้อัตราความผิดพลาดที่ได้มีค่าสูงกว่าอัตราความผิดพลาดที่คาดหวังของโมเดลที่เทรนด้วยข้อมูลทั้งหมดที่มี แต่หากค่า k มีค่ามากจะทำให้ขนาดของข้อมูลที่ใช้เทรนโมเดลในแต่ละรอบมีขนาดใหญ่ขึ้นซึ่งจะช่วยลดความลำเอียง (bias) ของอัตราความผิดพลาดโดยนัยทั่วไป (generalized error rate) ได้

กรณีที่ค่า k มีค่าเท่ากับจำนวนข้อมูลในชุดข้อมูล (k=N) จะเรียกว่า **leave-one-out method** ซึ่งในแต่ละรอบเราจะใช้ข้อมูลจำนวน N-1 ตัวอย่างในการเทรนโมเดลและทดสอบประสิทธิภาพด้วยข้อมูลเพียง 1 ตัวอย่าง วิธีการนี้มีข้อดีคือทำให้เราใช้ประโยชน์จากข้อมูลที่มีป้ายกำกับที่มีอยู่ได้อย่างคุ้มค่ามากที่สุด แต่มีข้อเสียคือ ผลการประเมินที่ได้อาจเป็นเท็จ และในกรณีที่ข้อมูลมีขนาดใหญ่มากจะต้องมีการประมวลผลข้อมูลเป็นจำนวนมากซึ่งอาจเป็นไปไม่ได้ ในทางปฏิบัติ ค่า k ที่นิยมใช้จะอยู่ระหว่าง 5 ถึง 10 เนื่องจากจะทำให้มีข้อมูลประมาณ 80% ถึง 90% ถูกนำมาใช้ในการเทรนโมเดล

3.7 การประเมินประสิทธิภาพของโมเดลกับการเลือกค่าไฮเปอร์พารามิเตอร์

ไฮเปอร์พารามิเตอร์ (hyper-parameters) คือพารามิเตอร์ของอัลกอริทึมการเรียนรู้ที่ต้องถูกกำหนดก่อนการเรียนรู้ของโมเดล เช่น ระดับความลึกสูงสุดของต้นไม้ตัดสินใจ การเลือกค่าไฮเปอร์พารามิเตอร์ (hyper-parameter selection) จะต้องกระทำระหว่างการเทรนโมเดล สำหรับการประเมินประสิทธิภาพโมเดลด้วยวิธี cross-validation เราสามารถผนวกรวมกระบวนการคัดเลือกไฮเปอร์พารามิเตอร์เข้าในการทำ cross-validation อธิบายเป็นแผนภาพได้ดังรูปที่ 3.10 กระบวนการนี้ว่ามีชื่อเรียกว่า nested cross-validation เนื่องจากมีการทำ cross-validation สำหรับการคัดเลือกไฮเปอร์พารามิเตอร์ซ้อนอยู่ภายในขั้นตอนการเทรนของการทำ cross-validation



รูปที่ 3.10 การทำ 3-fold nested cross-validation

จากรูปที่ 3.10 ในแต่ละรอบของการทำ 3-fold cross-validation เราจะใช้ชุดข้อมูลฝึกฝนในการสร้างและคัดเลือกค่าไฮเปอร์พารามิเตอร์โดยการทำ cross-validation บนชุดข้อมูลฝึกฝน โดยมีขั้นตอนอธิบายได้ดังนี้

กำหนดให้ค่าเซต P คือเซตของค่าของไฮเปอร์พารามิเตอร์ที่ต้องการเลือก, k คือจำนวน fold ของการทำ cross-validation และ D.train คือชุดข้อมูลฝึกฝนสำหรับการทำ cross-validation ในรอบปัจจุบันแล้ว การเลือกค่าไฮเปอร์พารามิเตอร์ด้วย cross-validation ซ้อนมีขั้นตอนดังนี้

1. กำหนดให้ Ntrain คือขนาดของชุดข้อมูลฝึกฝน D.train

2. แบ่ง D.train ออกเป็น k ส่วน คือ D.train_1, D.train_2, D.train_3

3. สำหรับแต่ละรอบ i=1 ถึง i=k:

    D.val(i) = D.train_i

    D.tr(i) = D.train \ D.train_i // ชุดข้อมูลส่วนที่ไม่ใช่ D.train_i

    สำหรับไฮเปอร์พารามิเตอร์ p แต่ละค่าในเซต P:

        เทรนโมเดล m บนชุดข้อมูล D.tr(i) โดยกำหนดค่าไฮเปอร์พารามิเตอร์เท่ากับ p

        err_sum(p, i) = ทดสอบประสิทธิภาพของโมเดล m บนชุดข้อมูล D.val(i)

4. err_val(p) = err_sum(p, 1) + err_sum(p, 2) + ... + err_sum(p, k)

5. เลือกไฮเปอร์พารามิเตอร์ p* ที่มีค่า err_val(p) น้อยที่สุด

6. เทรนโมเดล m* บนชุดข้อมูล D.train โดยใช้ค่าไฮเปอร์พารามิเตอร์เท่ากับ p*

7. ไฮเปอร์พารามิเตอร์และโมเดลที่เป็นผลลัพธ์ของการรัน cross-validation ของรอบปัจจุบันคือ p* และ m*

8. ประเมินประสิทธิภาพของโมเดล m* บน validation set ของ cross-validation รอบปัจจุบัน

9. รัน cross-validation รอบถัดไป

สรุป

- ตัวจำแนกประเภท (classifiers) คือ โมเดลที่แมปอินพุตแอทริบิวต์ไปเป็นเอาต์พุต โดยเอาต์พุตมีชนิดข้อแบบ categorical
- กระบวนการสร้างโมเดลการจำแนกประเภทแบ่งเป็นสองขั้นตอนหลักคือ การสร้างโมเดลโดยการเรียนรู้จากชุดข้อมูลฝึกฝน และการประเมินประสิทธิภาพโมเดลโดยใช้ชุดข้อมูลทดสอบ โดยมีเป้าหมายคือโมเดลที่สามารถจำแนกประเภทเมื่อได้รับอินพุตที่ไม่เคยพบมาก่อนได้
- ต้นไม้ตัดสินใจ (decision tree) คือโมเดลการจำแนกประเภทที่ใช้โครงสร้างต้นไม้ในการอธิบายกฎเกณฑ์สำหรับการจำแนกประเภท โดย root node และ internal nodes ของต้นไม้ตัดสินใจจะประกอบด้วยเงื่อนไขการทดสอบแอทริบิวต์ ส่วน leaf nodes ของต้นไม้ตัดสินใจจะมีคลาสเลเบิลหนึ่งชนิดกำหนดไว้
- การนำต้นไม้ตัดสินใจไปใช้งาน ทำได้โดยการทดสอบเงื่อนไขแอทริบิวต์และเคลื่อนไปตามเส้นทางในต้นไม้ตัดสินใจที่ตรงกับผลลัพธ์การทดสอบ จนกระทั่งไปถึง leaf node ซึ่งเป็นโหนดสุดท้ายในเส้นทาง คลาสเลเบิลของ leaf node จะเป็นเอาต์พุตหรือคำตอบของต้นไม้ตัดสินใจ
- อัลกอริทึมสำหรับสร้างต้นไม้ตัดสินใจ ใช้แนวทางการแก้ปัญหาแบบละโมบโดยการสร้างต้นไม้จากบนลงล่างแบบเวียนบังเกิด (greedy, top-down recursive strategy) ตัวอย่างอัลกอริทึมสร้างต้นไม้ตัดสินใจที่เป็นที่นิยม เช่น CART, C4.5
- เกณฑ์ในการคัดเลือกเงื่อนไขทดสอบแอทริบิวต์ ที่นิยมใช้ ได้แก่ information gain, gain ratio ซึ่งสามารถหาได้จากการคำนวณผลต่างระหว่างค่าความบริสุทธิ์ (เช่น entropy, gini index) ของโหนดก่อนการแตกกิ่งและหลังการแตกกิ่ง
- ชุดข้อมูลตรวจสอบ (validation set) คือชุดข้อมูลที่ใช้สำหรับการคัดเลือกโมเดล และการเลือกค่าไฮเปอร์พารามิเตอร์
- การประเมินประสิทธิภาพของโมเดลทำได้สองวิธีคือ holdout method และ cross-validation

แบบฝึกหัด

1. กำหนดชุดข้อมูลดังตารางต่อไปนี้

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

(ก) จงคำนวณ Gini index ชุดข้อมูลในตาราง

(ข) จงคำนวณ Gini index และ information gain ของแอทริบิวต์ Customer ID

(ค) จงคำนวณ Gini index และ information gain ของแอทริบิวต์ Gender

(ง) จงคำนวณ Gini index และ information gain ของแอทริบิวต์ Car Type

(จ) จงคำนวณ Gini index และ information gain ของแอทริบิวต์ Shirt Size

(ฉ) แอทริบิวต์ใดเป็นแอทริบิวต์ที่ดีที่สุดสำหรับใช้เป็นเงื่อนไขทดสอบแอทริบิวต์ของโหนดในต้นไม้ตัดสินใจ

(ช) แอทริบิวต์ Customer ID เหมาะกับการใช้เป็นเงื่อนไขทดสอบแอทริบิวต์ของต้นไม้ตัดสินใจหรือไม่ เพราะเหตุใด

2. จงอธิบายความหมายของ underfitting และ overfitting

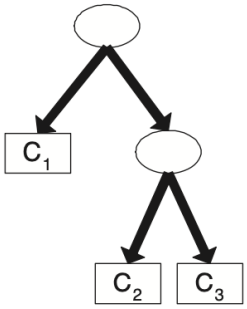
3. กำหนดผลการทดสอบประสิทธิภาพของโมเดลการจำแนกประเภทแบบสองคลาสดัง confusion matrix ต่อไปนี้ จงประเมินประสิทธิภาพของโมเดลดังกล่าวโดยคำนวณหาค่า Accuracy, Error rate, Recall, Precision, Specificity, False Positivie Rate และ F1-score

Model 1		Predicted Class	
		Positive Class	Negative Class
Actual Class	Positive Class	180	20
	Negative Class	60	340

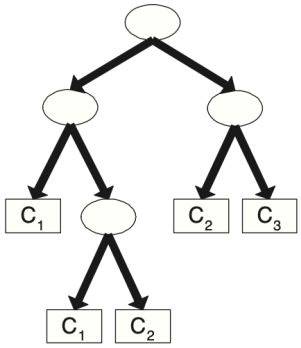
4. กำหนดต้นไม้ตัดสินใจให้ดังรูป จงคำนวณ total description length ของต้นไม้ตัดสินใจแต่ละต้น และสรุปว่าควรเลือกต้นไม้ตัดสินใจต้นใด เพราะเหตุใด เมื่อกำหนดข้อมูฃดังต่อไปนี้

- แต่ละ internal node ของต้นไม้ตัดสินใจเข้ารหัสโดยใช้รหัสของแอทริบิวต์ ถ้ามีแอทริบิวต์จำนวน m แอทริบิวต์ cost ของการเข้ารหัสของแต่ละแอทริบิวต์จะเท่ากับ log m
- แต่ละ leaf node เข้ารหัสโดยใช้รหัสของคลาส ถ้ามีจำนวนคลาสทั้งหมด k คลาส cost ของการเข้ารหัสคลาสจะเท่ากับ log k
- Cost(tree) เท่ากับ cost ของการเข้ารหัสทุกโหนดในต้นไม้ตัดสินใจ tree
- Cost(D|tree) ถูกเข้ารหัสโดยใช้จำนวนความผิดพลาดของการจำแนกประเภทของต้นไม้ตัดสินใจ แต่ละความผิดพลาดจะต้องใช้บิตจำนวน log n บิต เมื่อ n คือจำนวนข้อมูลในชุดฝึกฝน

(ก) ต้นไม้ตัดสินใจที่มีความผิดพลาดบนชุดฝึกฝนเท่ากับ 7



(ข) ต้นไม้ตัดสินใจที่มีความผิดพลาดบนชุดฝึกฝนเท่ากับ 4



5. จงอธิบายการประเมินประสิทธิภาพของโมเดลด้วยวิธีการ 10-fold cross validation