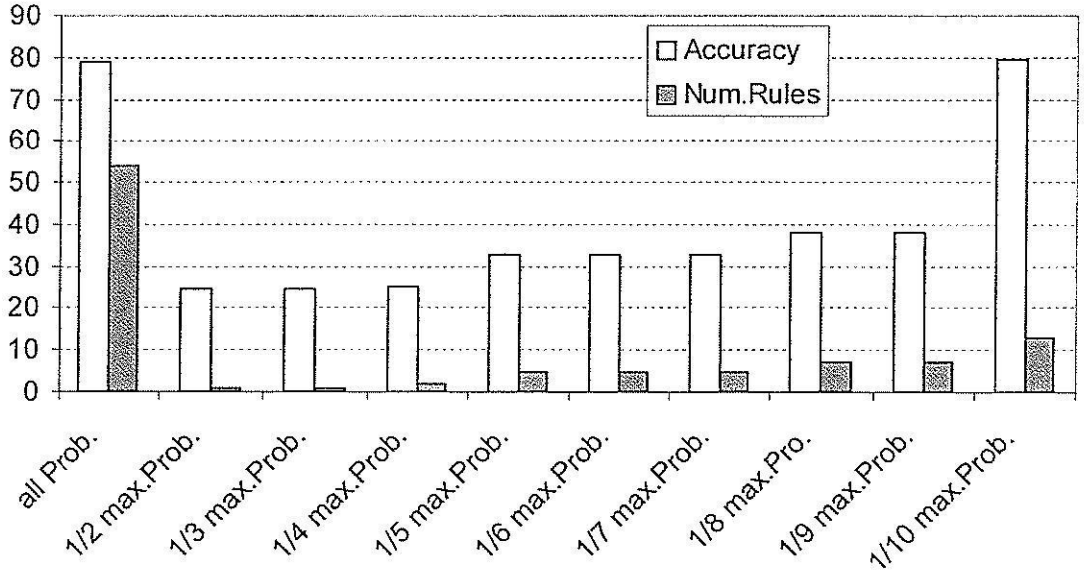
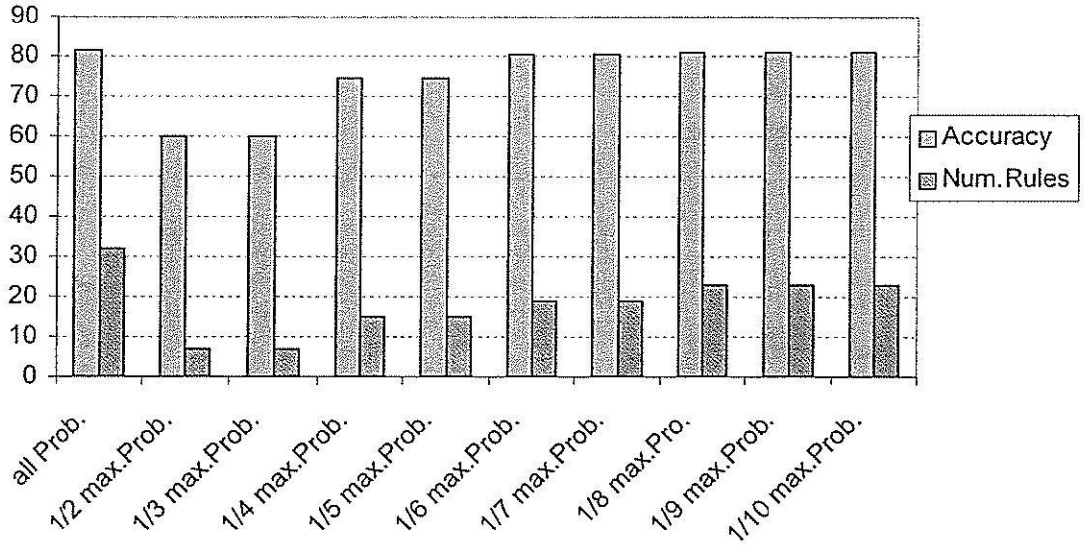
ตารางที่ 4.3 ความแม่นตรงของโมเดล Probabilistic decision rules เมื่อมีการลดขนาดของโมเดล

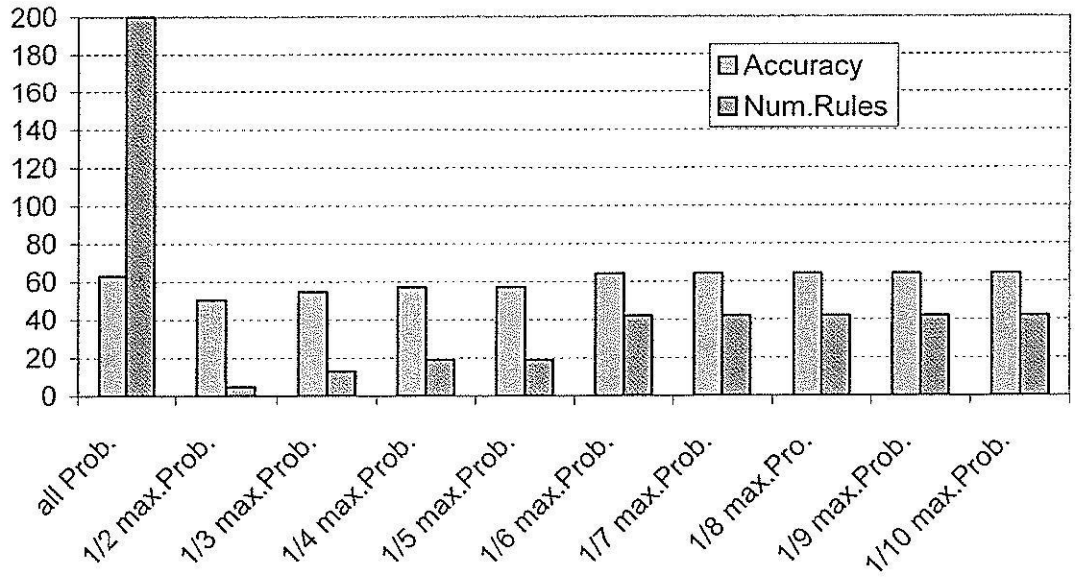
Monk		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	79.03	24.76	24.76	25	32.63	32.63	32.63	38.19	38.19	**79.62**
	# rules	54	1	1	2	5	5	5	7	7	**13**
Post-operative		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	81.65	60.09	60.09	74.65	74.65	**80.27**	80.27	80.95	80.95	80.95
	# rules	32	7	7	15	15	**19**	19	23	23	23
Breast cancer		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	63.15	50.52	54.73	57.12	57.12	**64.21**	64.21	64.21	64.21	64.21
	# rules	200	5	13	19	19	**42**	42	42	42	42
Vote		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	84.33	55.55	87.4	87.4	87.4	87.4	87.4	87.4	87.4	87.4
	# rules	47	1	**2**	2	2	2	2	2	2	2
Hepatitis		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	71.42	60.9	75	75	75	75	75	76.9	76.9	76.9
	# rules	16	3	4	4	4	4	4	5	5	5
Mushroom		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	100	91.48	91.48	96.72	96.72	**100**	100	100	100	100
	# rules	25	3	3	5	5	**8**	8	8	9	9



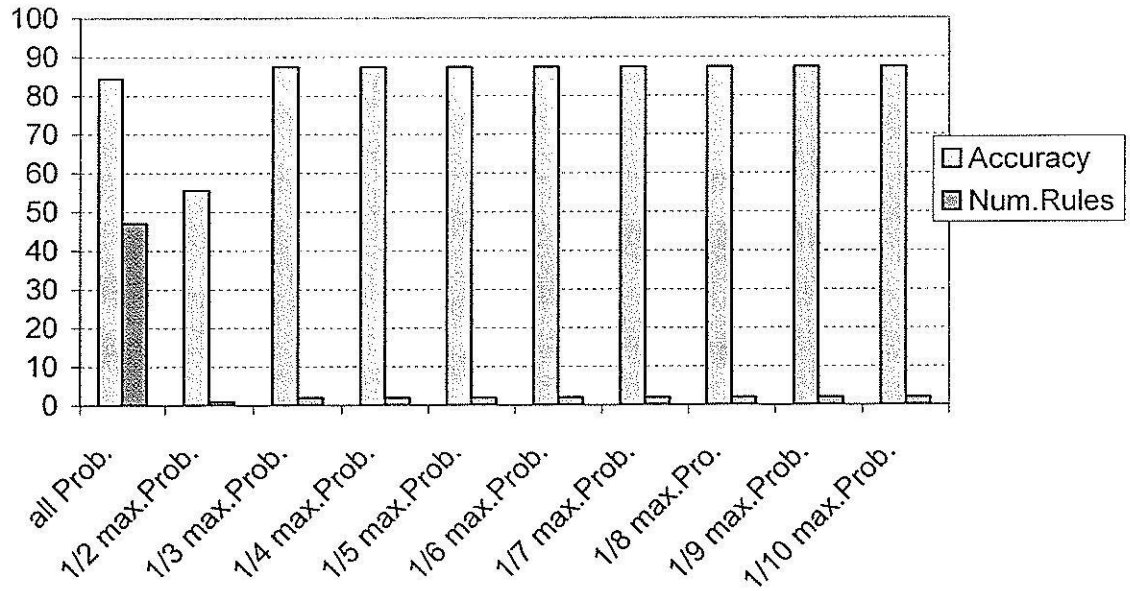
รูปที่ 4.8 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Monk



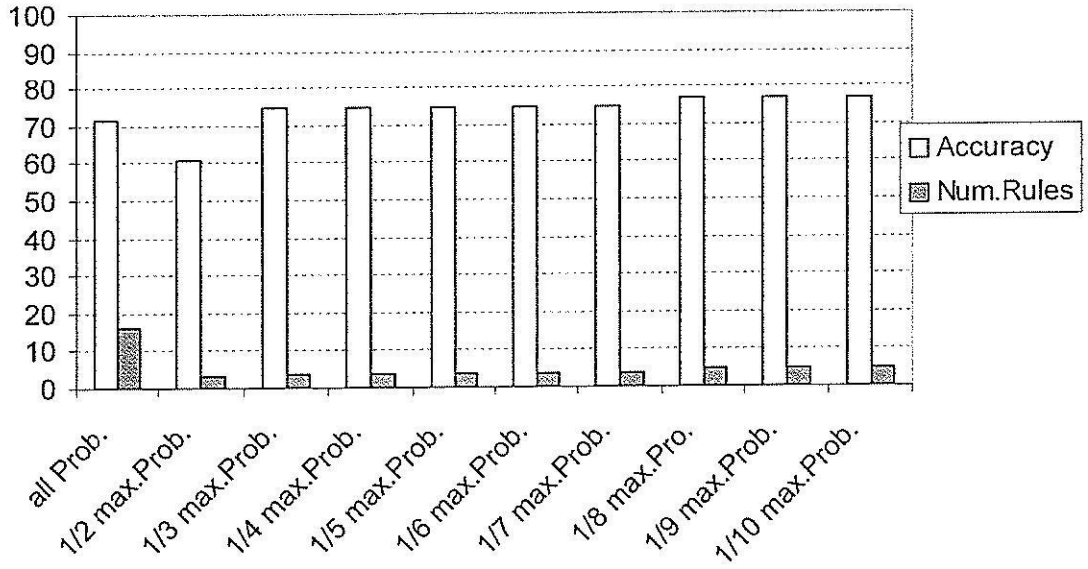
รูปที่ 4.9 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Post-operative



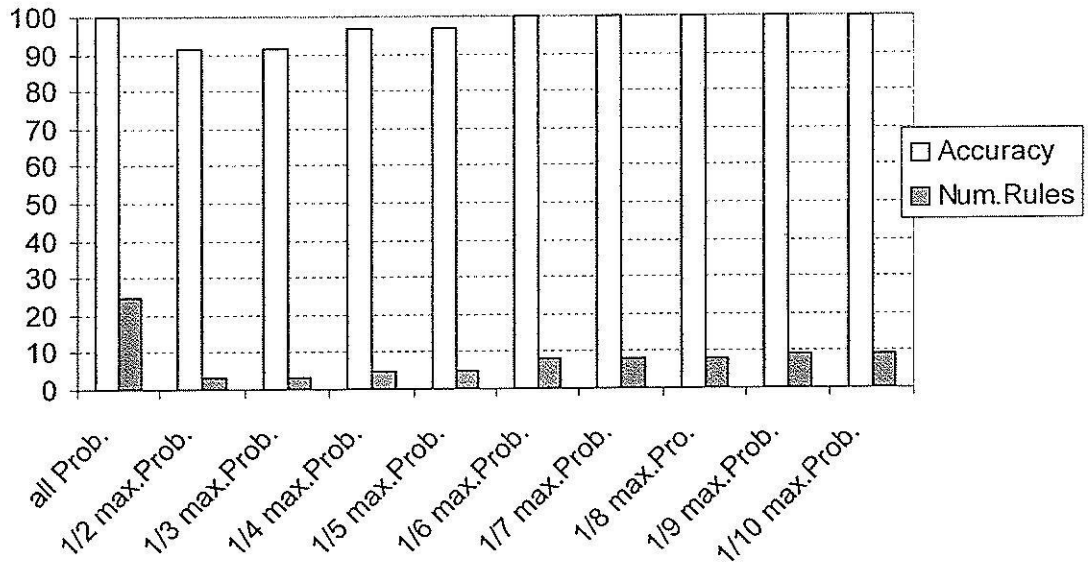
รูปที่ 4.10 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Breast cancer



รูปที่ 4.11 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Vote



รูปที่ 4.12 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Hepatitis



รูปที่ 4.13 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Mushroom

จากการทดสอบความถูกต้องในขั้นตอนการสร้างโมเดล ของโปรแกรมการทำเหมืองข้อมูลเพื่อการจำแนก ปรากฏผลว่าโมเดลที่ได้ในลักษณะของ probabilistic decision rules ให้ค่าความแม่นตรงเท่ากับโมเดลที่ได้จากโปรแกรม ID3 ดังแสดงในตารางที่ 4.2 และรูปที่ 4.7 ผลที่ได้นี้ตรงกับข้อสมมุติฐานเบื้องต้นก่อนทำการทดลอง เนื่องจากอัลกอริทึมที่พัฒนาขึ้นนี้ใช้วิธีการสร้างต้นไม้ตัดสินใจเช่นเดียวกับอัลกอริทึม ID3 แต่ได้เพิ่มเติมการคำนวณความน่าจะเป็นในเส้นทางจากโหนดรากถึงแต่ละโหนดใบที่เป็นค่าของการตัดสินใจ ความน่าจะเป็นนี้สามารถแปลความได้เป็นความครอบคลุมของโมเดลในเส้นทางจากโหนดรากถึงโหนดใบ

เมื่อการทดสอบความถูกต้องของการสร้างโมเดลได้ผลตรงตามความคาดหมายแล้ว ในขั้นตอนต่อไปเป็นการขยายขีดความสามารถของการปรับปรุงและการใช้งานโมเดล จากสมมุติฐานเบื้องต้นคือ เมื่อโปรแกรมแสดงผลลัพธ์เป็นโมเดลในรูปแบบกฎการตัดสินใจที่มีค่าความน่าจะเป็นกำกับ และผลลัพธ์มีการเรียงลำดับกฎตามค่าความน่าจะเป็น ดังนั้นผลลัพธ์นี้ควรจะสามารถลดจำนวนกฎการตัดสินใจลงได้ โดยคัดเลือกไว้เฉพาะกฎที่มีความน่าจะเป็นสูง เหตุผลที่สนับสนุนแนวคิดนี้คือ กรณีที่กฎการตัดสินใจมีความน่าจะเป็นต่ำ เช่น 0.001 ถ้าหากข้อมูลฝึกที่ใช้ในการสร้างโมเดลมีจำนวนข้อมูลทั้งหมด 1000 เรคคอร์ด กฎการตัดสินใจดังกล่าวจะถูกสร้างขึ้นเพื่ออธิบายหรือโมเดลข้อมูลเพียงหนึ่งเรคคอร์ดเท่านั้น ซึ่งการสร้างโมเดลเพื่อพยายามโมเดลข้อมูลให้ได้ครบถ้วนทั้งหมดมีโอกาสสูงมากที่จะได้โมเดลที่มีลักษณะเจาะจงมากเกินไป (overfitting)

การสร้างโมเดลที่มีลักษณะ overfitting จะทำให้เกิดผลเสียตามมาคือ เมื่อนำโมเดลไปใช้ทำนายคลาสของข้อมูลชุดอื่น เช่นข้อมูลทดสอบ หรือข้อมูลที่จะเกิดขึ้นใหม่ในอนาคต จะทำให้ผลการทำนายมีความถูกต้องลดลงกว่าการใช้โมเดลเดียวกันนั้นทำนายกับข้อมูลฝึก ซึ่งผลการทดลองตามตารางที่ 4.3 ยืนยันข้อสมมุติฐานนี้ นั่นคือ ในจำนวนข้อมูลที่ใช้ทดสอบหกชุด มีข้อมูลถึงสี่ชุด (ได้แก่ข้อมูล Monk, Breast cancer, Vote, Hepatitis) ที่ให้ผลการทำนายที่มีความแม่นตรงสูงขึ้นเมื่อคัดเลือกโมเดลให้มีเฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง

ในชุดข้อมูล Mushroom โมเดลที่มีเฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูงจะให้ผลการทดสอบความแม่นตรงสูงเทียบเท่ากับโมเดลปกติที่ยังไม่ได้มีการลดขนาด และในจำนวนหกชุดข้อมูลมีเพียงข้อมูลชุดเดียวคือข้อมูล Post-operative ที่โมเดลขนาดเล็กที่มีเฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง ให้ผลการทดสอบที่มีค่าความแม่นตรงลดลงจากโมเดลปกติเล็กน้อย (ประมาณ 1.38%)

จากผลการทดสอบดังกล่าวทำให้สามารถสรุปได้ว่าการลดขนาดของโมเดล โดยคัดเลือกไว้เฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง จะช่วยให้ได้โมเดลที่สามารถนำไปใช้ทำนายข้อมูลได้ดีขึ้น โมเดลที่ได้จะไม่มีลักษณะ overfitting ลักษณะการคัดเลือกกฎเช่นนี้เทียบเคียง

ได้กับการทำ post-pruning ของเทคนิค decision-tree induction แต่วิธีการทำจะง่ายกว่าและแนวคิดไม่ซับซ้อน

ในประเด็นของการพิจารณาว่าควรจะลดขนาดของโมเดลลงในปริมาณเท่าไรนั้น ผลการทดลองชี้ให้เห็นว่าเกณฑ์การตัดสินจะไม่คงที่ ทั้งนี้ลักษณะรูปแบบข้อมูลและการกระจายค่าของข้อมูลจะเป็นปัจจัยสำคัญที่ทำให้ไม่สามารถสร้างเกณฑ์ที่คงที่ได้ แต่อย่างไรก็ตามผลการทดลองในข้อมูลทั้งหกชุดชี้ให้เห็นว่า การลดขนาดของโมเดลสามารถพิจารณาจากกฎที่ให้ค่าความน่าจะเป็นสูงสุด จากนั้นใช้ค่าความน่าจะเป็นขั้นต่ำเป็นเกณฑ์คัดทิ้งกฎ โดยค่าความน่าจะเป็นขั้นต่ำนี้คำนวณได้จาก 1/4 *(maximum probability) จนถึงประมาณ 1/6 * (maximum probability)

4.3 สรุปผลโครงการวิจัยที่ 3

โครงการวิจัยเรื่องการพัฒนาการทำเหมืองข้อมูลแบบจำแนกนี้ มีวัตถุประสงค์หลักเพื่อพัฒนาซอฟต์แวร์ในลักษณะของโอเพนซอร์ส เผยแพร่เพื่อการใช้งานวิเคราะห์ข้อมูลในลักษณะของการทำเหมืองข้อมูล ที่มีข้อแตกต่างจากการวิเคราะห์ข้อมูลตามปกติ คือการทำเหมืองข้อมูลจะให้โปรแกรมทำหน้าที่วิเคราะห์หาความสัมพันธ์ระหว่างตัวแปรหรือแอททริบิวต์ที่จะสามารถใช้ในการทำนาย (predicting attributes) กับแอททริบิวต์ที่เป็นเป้าหมายหลักในการทำนาย (target attribute, goal attribute) ประเด็นในการวิเคราะห์คือต้องการทราบแอททริบิวต์ที่สามารถใช้ในการทำนายค่าของแอททริบิวต์เป้าหมายได้แม่นยำที่สุด การตรวจสอบหาแอททริบิวต์ที่เหมาะสมจะใช้ในการทำนายค่าเป้าหมายจะเป็นไปโดยอัตโนมัติ ในขณะที่กระบวนการวิเคราะห์ข้อมูลตามปกติจะตรวจสอบหาแอททริบิวต์ที่เหมาะสมแบบกึ่งอัตโนมัติ โดยนักวิเคราะห์จะคัดเลือกแอททริบิวต์ที่อาจใช้ในการทำนายได้มาพิจารณา correlation ของแอททริบิวต์ทำนายกับแอททริบิวต์เป้าหมายทีละแอททริบิวต์ และสุดท้ายจะคัดเลือกแอททริบิวต์ที่มี correlation ดีที่สุด การทำเหมืองข้อมูลจึงมีประโยชน์ที่ช่วยให้งานของนักวิเคราะห์ข้อมูลทำได้เร็วขึ้น

การทำเหมืองข้อมูลเป็นเทคโนโลยีที่ครอบคลุมงานวิเคราะห์ข้อมูลทุกประเภท เช่น การจำแนกข้อมูล (data classification), การวิเคราะห์กลุ่มข้อมูล (cluster analysis), การวิเคราะห์ความสัมพันธ์ภายในกลุ่มข้อมูล (association analysis), การจัดหมวดหมู่ข้อมูล (data segmentation), การสรุปข้อมูล (data summarization), การวิเคราะห์ข้อมูลเบี่ยงเบน (data deviation analysis) และการวิเคราะห์ใบแบบอื่นๆ อีกหลายประเภท ดังนั้นงานวิจัยนี้ที่เน้นเฉพาะการจำแนกข้อมูลจึงเป็นงานวิเคราะห์ประเภทหนึ่งของการทำเหมืองข้อมูล

การจำแนกข้อมูลจัดเป็นงานเรียนรู้ประเภทมีการชี้แนะ (supervised learning) ผู้ใช้งานโปรแกรมหรือนักวิเคราะห์ข้อมูล จะเป็นผู้ระบุเป้าหมายว่าต้องการจำแนกข้อมูลในแอททริบิวต์ใด จากนั้นส่งอินพุทเป็นข้อมูลที่มีค่าปรากฏในแอททริบิวต์เป้าหมาย รวมถึงมีค่าในแอททริบิวต์อื่นๆ ที่เป็นแอททริบิวต์ประกอบ ข้อมูลนี้ถูกเรียกว่าข้อมูลฝึก (training data) เนื่องจากโปรแกรมจะใช้

ข้อมูลนี้ฝึกกระบวนการสร้างโมเดล เพื่อใช้โมเดลนี้ทำนายค่าของแอททริบิวต์เป้าหมายได้อย่างแม่นยำ การทำเหมืองข้อมูลแบบจำแนกจึงเป็นกระบวนการค้นหาแอททริบิวต์ที่สามารถนำมาใช้ช่วยทำนายค่าเป้าหมาย (หรือจำแนกค่าของแอททริบิวต์เป้าหมาย) ได้อย่างแม่นยำที่สุด เทคนิคในการค้นหา แอททริบิวต์ดังกล่าวมีได้หลายเทคนิค เช่น ใช้การสร้าง decision tree เพื่อให้โครงสร้างข้อมูลที่ได้เป็นเครื่องมือในการจำแนกค่าแอททริบิวต์เป้าหมาย หรือใช้การคำนวณหาข้อมูลที่อยู่ใกล้ที่สุดแล้วตัดสินค่าแอททริบิวต์เป้าหมายตามข้อมูลนั้น และเทคนิคอื่นๆ อีกมาก การเลือกใช้เทคนิคที่แตกต่างกันจะให้ประสิทธิภาพของโมเดลแตกต่างกัน และให้รูปแบบโมเดลที่แตกต่างกันด้วย รูปแบบที่ยอมรับกันโดยทั่วไปว่าแปลความได้ง่ายและมีความแม่นยำค่อนข้างสูงคือการใช้ decision tree ที่สามารถแปลงเป็น decision rule ได้โดยความหมายของโมเดลไม่เปลี่ยนไป

งานวิจัยนี้เลือกใช้เทคนิคการสร้าง decision tree หรือต้นไม้ตัดสินใจ เนื่องจากต้องการให้ผู้ใช้ทั่วไปสามารถใช้และทำความเข้าใจโมเดลได้ง่าย วิธีการสร้างโมเดลใช้เทคนิคพื้นฐานเช่นเดียวกับอัลกอริทึม ID3 (Quinlan 1986; 1993) แต่ได้เพิ่มเติมความสามารถของโมเดลให้สามารถแสดงผลในลักษณะของกฎการตัดสินใจ (decision rules) เพื่อให้สามารถนำโมเดลไปเป็นข้อมูลในฐานความรู้ของระบบผู้เชี่ยวชาญ หรือระบบที่ช่วยในการตัดสินใจแบบอื่นๆ นอกจากนี้ในงานวิจัยนี้ยังได้เพิ่มเติมฟังก์ชันการคำนวณความน่าจะเป็นประกอบในโครงสร้าง decision tree ค่าความน่าจะเป็นนี้เชื่อมโยงถึงสัดส่วนการครอบคลุมข้อมูลของโมเดล ค่าความน่าจะเป็นที่มีค่าสูงหมายถึงกฎการตัดสินใจนั้นมีความเป็นไปได้สูงที่จะถูกนำไปใช้จำแนกค่าแอททริบิวต์เป้าหมาย และค่าความน่าจะเป็นที่มีค่าต่ำก็จะหมายถึงโอกาสที่กฎนั้นจะถูกนำไปใช้งานมีน้อย ดังนั้นค่าความน่าจะเป็นนี้จึงถูกนำไปประยุกต์ใช้ในการลดขนาดของโมเดลลงได้ ทั้งนี้เพราะในการใช้งานจริงเมื่อโปรแกรมเรียนรู้จากข้อมูลที่มีขนาดใหญ่และมีจำนวนแอททริบิวต์มาก มักจะให้ผลลัพธ์เป็นโมเดลที่มีขนาดใหญ่และมีความซับซ้อนมาก โมเดลเช่นนี้จะไม่สะดวกในการนำไปใช้งานและมักจะมีโอกาสให้ผลการจำแนกข้อมูลที่มีความแม่นยำต่ำ

ผลลัพธ์ที่ได้จากการทำงานของโปรแกรมในลักษณะของกฎการตัดสินใจ นอกจากจะแปลผลได้ง่ายแล้ว ยังเอื้อต่อการปรับปรุงผลลัพธ์ให้มีขนาดที่เล็กลงได้ โมเดลที่มีขนาดเล็กจะใช้งานสะดวกกว่าโมเดลที่มีขนาดใหญ่และซับซ้อน การพิจารณาลดขนาดของโมเดลจะใช้ค่าความน่าจะเป็นสูงสุดเป็นเกณฑ์ (เรียกว่าค่า maxProb) และใช้เกณฑ์นี้กำหนดค่าความน่าจะเป็นต่ำสุด (เรียกว่าค่า minProb) ที่จะใช้คัดเลือกผลลัพธ์ กฎการตัดสินใจที่มีค่าความน่าจะเป็นอยู่ระหว่างช่วง [minProb-maxProb] นี้เท่านั้นที่จะถูกคัดเลือกเป็นผลลัพธ์สุดท้าย งานวิจัยนี้ได้ทดลองพิจารณาค่า minProb ที่แปรผันตามค่า maxProb ที่ขนาดต่างๆกันสิบขนาด ได้แก่ 1/2(maxProb), 1/3(maxProb), 1/4(maxProb), 1/5(maxProb), 1/6(maxProb), 1/7(maxProb), 1/8(maxProb), 1/9(maxProb), 1/10(maxProb)

จากผลการทดลองพบว่าที่ขนาด 1/3(maxProb) ถึง 1/6(maxProb) ให้โมเดลที่มีค่าความแม่นตรง (accuracy) ของการจำแนกข้อมูลดีที่สุด และสามารถลดขนาดของโมเดลได้โดยเฉลี่ยสูงถึง 72.37% นั่นคือถ้าโมเดลเริ่มต้นมีจำนวนกฎการตัดสินใจ 100 กฎ เมื่อคัดเลือกด้วยเกณฑ์ค่าความน่าจะเป็นจะลดจำนวนกฎเหลือเพียง 28 กฎ

นอกจากความสามารถในการลดขนาดของโมเดล ทำให้ได้ผลลัพธ์ที่ใช้งานได้สะดวกแล้ว จากผลการทดลองยังพบว่าโมเดลที่มีขนาดเล็กสามารถทำนายแอททริบิวต์เป้าหมายในชุดข้อมูลทดสอบได้ดีกว่าโมเดลที่ยังไม่ได้ลดขนาดลง สาเหตุหลักของการเพิ่มค่าความแม่นตรงนี้เนื่องจากโมเดลที่มีขนาดเล็กลงจะช่วยลดผลกระทบในเรื่อง overfitting ของโมเดล

ข้อเสนอแนะ

จากแนวทางการลดขนาดของโมเดล ด้วยการใช้เฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง จะทำให้ได้โมเดลที่มีจำนวนกฎน้อย แต่ผลที่อาจจะตามมาคือกฎที่ถูกคัดเลือกไว้อาจจะมีเงื่อนไขการพิจารณาไม่ครบถ้วนทำให้โมเดลไม่สมบูรณ์ หรือกฎบางส่วนอาจขัดแย้งกัน แนวทางแก้ไขปัญหากรณีเงื่อนไขไม่ครบถ้วนคือจะต้องมี default rule เตรียมไว้ ส่วนในกรณีที่กฎขัดแย้งอาจใช้วิธีการโหวตจากทุกกฎที่เกี่ยวข้อง จากนั้นใช้ค่าทำนายเป็นค่าส่วนใหญ่จากผลโหวต แนวทางการใช้งานโมเดลในกรณีดังกล่าวแสดงเป็นขั้นตอนได้ดังอัลกอริทึมต่อไปนี้

Algorithm 4.3 Probabilistic-decision-rule inferring

Input: a knowledge base (KB) containing probabilistic decision rules, and
a new case with unknown class information

Output: a decision on most likely class of the new case

---

(1) Read all attribute-value pairs appeared in the given case

(2) Compare the attribute-value pairs with each relevant rule in the KB to get the decision class value

(3) Compute the decision confidence as
(number of matched attribute-value pair) × (probability of the decision rule)

(4) Output a final decision based on the majority voting scheme

(5) If there is no rule matched the new case,
Output a final decision based on the majority of the top five decision rules

---

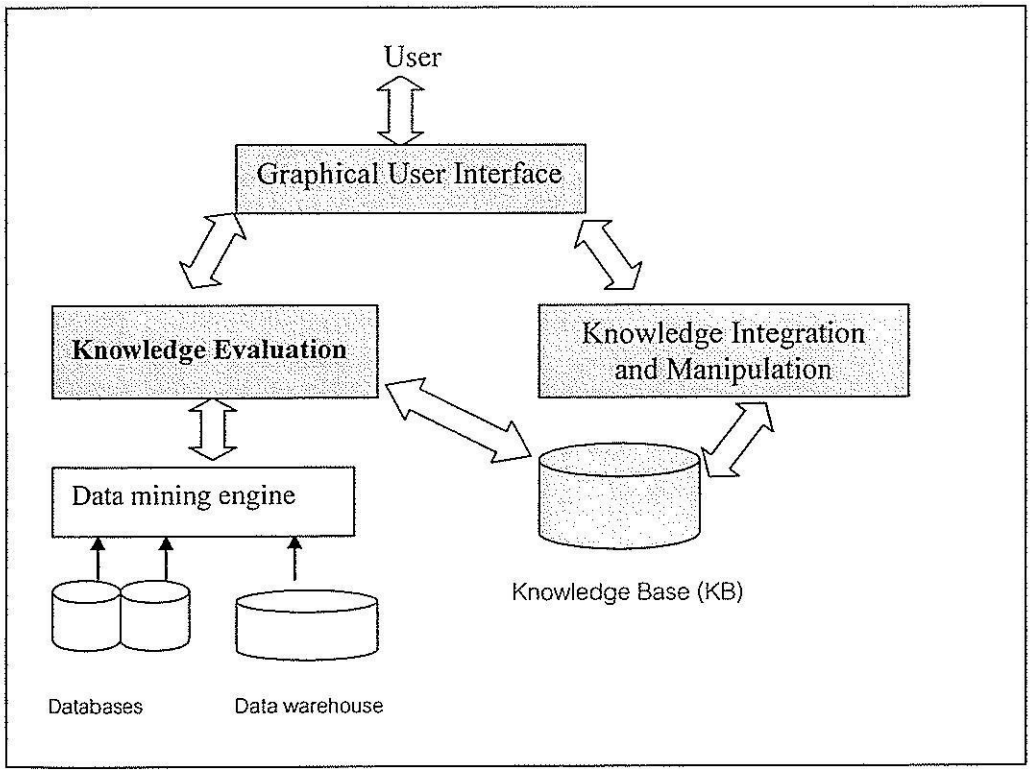
บทที่ 5

การประมวลผลหลังกระบวนการทำเหมืองข้อมูล (โครงการวิจัยที่ 4)

5.1 วิธีดำเนินการวิจัยโครงการวิจัยที่ 4

5.1.1 กรอบแนวคิดของงานวิจัย

โครงการวิจัยนี้มีวัตถุประสงค์หลักเพื่อการพัฒนาเทคนิคในการจัดการกับความรู้ที่เป็นผลลัพธ์จากกระบวนการทำเหมืองข้อมูล ทั้งนี้เนื่องจากผลลัพธ์จากการทำเหมืองข้อมูลมักจะได้ความรู้ที่เป็นโมเดลหรือแพทเทิร์นของข้อมูลในปริมาณมากเกินไป การจัดการกับความรู้ที่ได้จึงหมายถึงการประเมินคุณภาพของความรู้เพื่อคัดเลือกเฉพาะความรู้ที่น่าจะใช้ประโยชน์ได้มากที่สุด โดยระบบต้นแบบที่พัฒนาขึ้นจะประกอบด้วยส่วนประกอบหลักสองส่วนคือ ส่วนประกอบแรกคือส่วนวิเคราะห์ความรู้ (knowledge evaluation) ที่ได้จากการทำเหมืองข้อมูล และส่วนประกอบที่สองคือส่วนรวบรวมและจัดการกับความรู้ (knowledge integration and manipulation) ซึ่งจะรวมถึงการสอบถามความรู้ที่รวบรวมไว้ในฐานความรู้ สถาปัตยกรรมหลักของระบบจัดการความรู้นี้แสดงได้ดังรูปที่ 5.1



รูปที่ 5.1 สถาปัตยกรรมของระบบจัดการความรู้ที่ได้จากการทำเหมืองข้อมูล

ส่วนวิเคราะห์ความรู้ ทำหน้าที่รับโมเดลข้อมูลจากการทำเหมืองข้อมูลแบบจำแนก โมเดลข้อมูลนี้จะอยู่ในรูปแบบของต้นไม้ตัดสินใจที่มีโครงสร้างหลักคือโหนดต่างๆ (node) ของต้นไม้ และเส้นเชื่อมโหนดทุกเส้น (edge) เมื่อได้รับข้อมูล node และ edge ทั้งหมดจากโปรแกรม data mining engine ส่วนวิเคราะห์ความรู้จะแปลงข้อมูล node และ edge ให้อยู่ในรูปแบบของกฎการตัดสินใจ (decision rule) ซึ่งจะเป็นข้อความในลักษณะ IF...THEN... ในขั้นตอนนี้จะมีการคำนวณค่า coverage ของแต่ละกฎ แล้วแปลงเป็นค่าสัดส่วนของจำนวนข้อมูลที่กฎครอบคลุมเทียบกับจำนวนข้อมูลทั้งหมด ค่าสัดส่วนที่ได้จะมีค่าอยู่ระหว่าง 0.0-1.0 ดังนั้นค่าสัดส่วนนี้สามารถแปลความหมายได้ว่า เป็นความน่าจะเป็นที่กฎจะประยุกต์ใช้ได้กับเหตุการณ์ที่จะเกิดขึ้นในอนาคต กฎที่มีความน่าจะเป็นสูงจะถูกคัดเลือกและบันทึกไว้ในฐานความรู้

ส่วนรวบรวมและจัดการกับความรู้ ทำหน้าที่จัดเก็บความรู้ในรูปแบบของกฎการตัดสินใจและทำหน้าที่เป็น expert system shell เพื่ออำนวยความสะดวกให้ผู้ใช้สามารถสอบถามความรู้จากฐานความรู้ได้ ผลลัพธ์ที่แสดงให้ผู้ใช้เห็นจะเป็นข้อแนะนำการตัดสินใจพร้อมกับแสดงค่าความน่าจะเป็นของข้อแนะนำนั้น ค่าความน่าจะเป็นนี้จะหมายถึงระดับความเชื่อมั่นของคำแนะนำ

5.1.2 การออกแบบอัลกอริทึมเพื่อการประเมินและคัดเลือกกฎ

ในการออกแบบอัลกอริทึมที่เป็นโมดูลหลักของโปรแกรมการประมวลผลหลังกระบวนการทำเหมืองข้อมูล จะออกแบบสอดคล้องกับสถาปัตยกรรมของระบบที่แสดงดังรูปที่ 5.1 ดังนั้นระบบนี้จะประกอบด้วยสองโมดูล คือ Knowledge evaluation และ Knowledge integration and manipulation รายละเอียดของแต่ละอัลกอริทึมแสดงได้ดังต่อไปนี้

โมดูล Knowledge evaluation

อัลกอริทึมในส่วนนี้ทำหน้าที่รับข้อมูล node และ edge จากโปรแกรมทำเหมืองข้อมูลที่ทำหน้าที่สร้างโมเดลข้อมูลในลักษณะของต้นไม้ตัดสินใจ อัลกอริทึมนี้เริ่มต้นทำงานด้วยการแสดงจอภาพโต้ตอบกับผู้ใช้ให้ระบุชื่อฐานข้อมูลและค่า coverage จากผู้ใช้ ค่า coverage นี้จะถูกใช้เป็นเกณฑ์ขั้นต่ำในการคัดเลือกกฎการตัดสินใจเพื่อบันทึกลงฐานความรู้ โมเดลข้อมูลที่แสดงด้วยโครงสร้าง node และ edge จะถูกแปลงเป็นกฎการตัดสินใจด้วยวิธีการ traverse โครงสร้างต้นไม้ตั้งแต่โหนดราก (โหนดหมายเลข 0) ไล่ไปตามเส้นเชื่อมต่างๆจนกระทั่งถึงโหนดใบ โครงสร้างข้อมูลของ node และ edge แสดงได้ดังนี้

```
node(nodeID, [Positive_Instances]-[Negative_Instancess])
edge(ParentNode, EdgeLabel, ChildNode)
```

โหนดแต่ละโหนดจะประกอบด้วยหมายเลขโหนดและข้อมูลในโหนด การระบุข้อมูลที่อยู่ในโหนดแต่ละโหนดจะใช้วิธีระบุหมายเลขข้อมูล (ข้อมูลแต่ละรายการจะมีหมายเลขกำกับ) โดยข้อมูลจะถูกแยกเป็น positive instances และ negative instances ถ้าโหนดนั้นเป็นโหนดใบและข้อมูลเป็น positive instances ทั้งหมด ลิสต์ของ negative instances จะเป็นลิสต์ว่าง และในทำนองเดียวกันถ้าโหนดนั้นเป็นโหนดใบและข้อมูลในโหนดเป็น negative instances ทั้งหมด ลิสต์ของ positive instances จะเป็นลิสต์ว่าง ดังนั้นการปรากฏลิสต์ใดลิสต์หนึ่งในโหนดเป็นลิสต์ว่างจะหมายถึงการเป็นโหนดใบ

ในส่วนของโครงสร้างกิ่งหรือ edge จะทำหน้าที่เชื่อมโยงโหนดที่เป็นโหนดแม่ไปยังโหนดลูก ดังนั้นโครงสร้าง edge จะระบุข้อมูลสามส่วนคือ หมายเลขโหนดที่เป็นโหนดแม่ของ edge, ชื่อของ edge (คือชื่อและค่าของแอททริบิวต์ เช่น outlook=sunny) และหมายเลขของโหนดที่เป็นโหนดลูก

ผลลัพธ์สุดท้ายที่ได้จากอัลกอริทึมคือกฎการตัดสินใจ หรือ decision rules โดยกฎนี้จะเรียงลำดับตามค่าความน่าจะเป็นสูงสุดลดหลั่งลงมาจนกระทั่งถึงกฎการตัดสินใจที่มีค่าความน่าจะเป็นต่ำสุดตามที่ผู้ใช้กำหนด รายละเอียดขั้นตอนต่างๆในอัลกอริทึมแสดงได้ดังต่อไปนี้

Algorithm 5.1 Knowledge evaluation

Input: a data model as decision tree with node and edge structures

Output: a set of probabilistic decision rules ranking in descending order

---

(1) Display GUI to get a dataset name and a minimum probability value

(2) Traverse tree from a root node to each leaf node

  (2.1) Collect edge information and count number of data instances

  (2.2) Compute probability as a proportion
  (number of instances at leaf node) / (total data instances in a data set)

  (2.3) Assert a rule containing a triplet

    <attribute-value pair, class, probability value> into temporary KB

(3) Sort rules in the KB in descending order according to the rules' probability

(4) Remove rules that have probability less than the specified threshold

(5) Assert selected rules into the KB and return KB as an output

---

ขั้นตอนแรกของอัลกอริทึมเป็นการอ่านข้อมูลจาก node และ edge เรียงลำดับตามค่าหมายเลขโหนด โดยเริ่มจากโหนดรากที่เป็นโหนดหมายเลขศูนย์ไล่ตามแต่ละกิ่งหรือ edge ลงมาจนกระทั่งถึงโหนดใบ ในขณะที่อ่านข้อมูลในแต่ละกิ่งจะนับจำนวนข้อมูลที่ถูกแยกย่อยลงมาในแต่ละกิ่งเพื่อคำนวณค่าความน่าจะเป็นในการครอบคลุมข้อมูลของกฎการตัดสินใจ ค่าความน่าจะเป็น

นี้เป็นสัดส่วนระหว่างจำนวนข้อมูลที่โหนดใบหารด้วยจำนวนข้อมูลทั้งหมด (นั่นคือหารด้วยข้อมูลที่โหนดราก) จากนั้นบันทึกกฎการตัดสินใจที่แต่ละกฎมีส่วนประกอบ 3 ส่วน คือ ค่าของแอททริบิวต์ที่เป็นปัจจัยประกอบการตัดสินใจ, คลาสของข้อมูลที่เป็นผลของการตัดสินใจ และ ค่าความน่าจะเป็นของกฎการตัดสินใจ

เมื่อแปลงโครงสร้างต้นไม้ตัดสินใจเป็นกฎการตัดสินใจได้ครบในทุกกิ่ง และทุกโหนดใบแล้ว จะมีการเรียงลำดับกฎตามค่าความน่าจะเป็น (ขั้นตอนที่ 3 ตามอัลกอริทึม) โดยเรียงจากค่ามากสุดลงมาหาค่าน้อยที่สุด ต่อจากนั้นจะตัดทิ้งกฎที่มีค่าความน่าจะเป็นต่ำกว่าเกณฑ์ที่ผู้ใช้กำหนด (ขั้นตอนที่ 4 ตามอัลกอริทึม) และในขั้นตอนสุดท้ายเป็นการบันทึกกฎการตัดสินใจที่ได้รับการคัดเลือกแล้วลงในฐานความรู้

โมดูล Knowledge integration and maipulation

อัลกอริทึมในส่วนนี้ทำหน้าที่แปลงกฎการตัดสินใจที่คัดเลือกไว้เฉพาะกฎที่มีค่าความน่าจะเป็นสูงถึงเกณฑ์ที่กำหนด เป็นกฎที่จะใช้ในระบบผู้เชี่ยวชาญ (expert system) เพื่ออำนวยความสะดวกให้ผู้ใช้สามารถสอบถามความรู้ต่างๆ จากฐานความรู้ กฎที่จำเป็นต้องใช้ในระบบผู้เชี่ยวชาญจะประกอบด้วยกฎที่เรียกว่า expert rules และ consulting rules

กฎในกลุ่มของ expert rules จะถูกใช้ในระดับ top-goal ของกระบวนการอนุมาน (inference process) วิธีการแปลงจากกฎการตัดสินใจให้เป็นกฎที่เรียกว่า expert rule จะต้องมีการสร้างส่วน head และส่วน body ของกฎ expert ส่วน head จะเป็นผลการตัดสินใจ (คือ edge label ในโหนดใบของโครงสร้างต้นไม้ หรือส่วน Then ในกฎการตัดสินใจ) ส่วน body จะเป็นรายละเอียดแอททริบิวต์ที่ปรากฏในส่วน If ของกฎการตัดสินใจ

กฎในกลุ่มของ consulting rules จะถูกใช้เพื่อการโต้ตอบและสอบถามรายละเอียดข้อมูล (หมายถึง ค่าของแอททริบิวต์ต่างๆ) จากผู้ใช้ในระหว่างที่ระบบผู้เชี่ยวชาญทำการวิเคราะห์เพื่อหาคำแนะนำที่เหมาะสมแสดงแก่ผู้ใช้ จำนวนกฎที่สร้างจะเท่ากับจำนวนแอททริบิวต์ทั้งหมดของข้อมูล ส่วน head ของกฎจะเป็นชื่อแอททริบิวต์ ส่วน body ของกฎจะเป็นชื่อเพรดิเคตที่มีอาร์กิวเม้นท์เป็นชื่อแอททริบิวต์และค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์นั้นๆ

รายละเอียดขั้นตอนต่างๆ ของการแปลงกฎการตัดสินใจ ให้เป็น expert rules และ consulting rules เพื่อใช้ในระบบผู้เชี่ยวชาญแสดงได้ดังอัลกอริทึมต่อไปนี้

Algorithm 5.2 Knowledge integration and manipulation

Input: a set of probabilistic decision rules stored in KB

Output: a set of rules to be used by an expert system shell

(1) For each probabilistic decision rule

  (1.1) Scan information in the If-part and the Then-part

  (1.2) Generate head of expert rule from the Then-part
    type (Then-part, probability value) :-

  (1.3) Generate body of expert rule from the If-part
    :- atrribute_name1(value), ..., attribute_nameN(value).

  (1.4) Write expert rule in a file '1.knb'

(2) For each data attribute

  (2.1) Generate head of consulting rule
    attribute_name(X) :-

  (2.2) Generate body of consulting rule
    :- menuask(attribute_name, X, [list of attribute values]).

(3) Assert consulting rules into the file '1.knb' and return KB as an output

5.1.3 การพัฒนาโปรแกรมเพื่อการประมวลผลหลังการทำเหมืองข้อมูล

โปรแกรมเพื่อการทำเหมืองข้อมูลแบบจำแนกนี้พัฒนาขึ้นโดยใช้ภาษาโปรล็อก การอธิบายขั้นตอนพัฒนาโปรแกรมจะใช้ไวยากรณ์ของภาษาโปรล็อกตามมาตรฐานของ SWI Prolog เวอร์ชัน 5.6.55 (ดาวน์โหลดได้จากเว็บไซต์ www.swi-prolog.org) ลักษณะเด่นประการหนึ่งของภาษาโปรล็อกคือการใช้รูปแบบเดียวกันของทั้งข้อมูลและคำสั่งที่ทำงานกับข้อมูล

รูปแบบของข้อมูล

ข้อมูลที่จะเป็นอินพุทของโปรแกรมทำเหมืองข้อมูล จะมีลักษณะเป็นข้อความที่อยู่ในรูปแบบของข้อความที่เป็นจริง หรือ fact ตัวอย่างของข้อมูลแสดงได้ดังรูปที่ 5.2 ข้อมูลดังรูปเป็นข้อมูลการวินิจฉัยของจักษุแพทย์ว่าคนไข้ที่มารับการวินิจฉัยแต่ละราย สามารถใส่คอนแท็กเลนส์ได้หรือไม่ ถ้าคนไข้สามารถใส่คอนแท็กเลนส์ได้จะมีค่า class=yes ส่วนรายที่ไม่สามารถใส่คอนแท็กเลนส์ (เนื่องจากพยาธิสภาพไม่เหมาะสม เช่นมีนัยน์ตาแห้งเกินไป) จะมีค่า class=no

```
%% Data lens

% attributes: names and their possible values
%
attribute(age, [young, pre_presbyopic, presbyopic]).
attribute(spectacle, [myope, hypermetrope]).
attribute(astigmatism, [no, yes]).
attribute(tear, [reduced, normal]).
attribute(class, [ yes, no]).

% data
instance(1, class=no, [age=young, spectacle=myope, astigmatism=no, tear=reduced]).
instance(2, class=yes, [age=young, spectacle=myope, astigmatism=no, tear=normal]).
instance(3, class=no, [age=young, spectacle=myope, astigmatism=yes, tear=reduced]).
instance(4, class=yes, [age=young, spectacle=myope, astigmatism=no, tear=normal]).
instance(5, class=no, [age=young, spectacle=hypermetrope, astigmatism=no, tear=reduced]).
instance(6, class=yes, [age=young, spectacle=hypermetrope, astigmatism=no, tear=normal]).
instance(7, class=no, [age=young, spectacle=hypermetrope, astigmatism=yes, tear=reduced]).
instance(8, class=yes, [age=young, spectacle=hypermetrope, astigmatism=yes, tear=normal]).
instance(9, class=no, [age=pre_presbyopic, spectacle=myope, astigmatism=no, tear=reduced]).
instance(10,class=yes, [age=pre_presbyopic, spectacle=myope, astigmatism=no, tear=normal]).
instance(11,class=no,[age=pre_presbyopic, spectacle=myope, astigmatism=yes, tear=reduced]).
instance(12,class=yes,[age=pre_presbyopic, spectacle=myope, astigmatism=yes, tear=normal]).
instance(13,class=no,[age=pre_presbyopic,spectacle=hypermetrope,astigmatism=no,tear=reduced]).
instance(14,class=yes,[age=pre_presbyopic,spectacle=hypermetrope,astigmatism=no,tear=normal]).
instance(15,class=no,[age=pre_presbyopic,spectacle=hypermetrope,astigmatism=yes,tear=reduced]).
instance(16,class=no,[age=pre_presbyopic,spectacle=hypermetrope,astigmatism=yes, tear=normal]).
instance(17,class=no, [age=presbyopic, spectacle=myope, astigmatism=no, tear=reduced]).
instance(18,class=no, [age=presbyopic, spectacle=myope, astigmatism=no, tear=normal]).
instance(19,class=no, [age=presbyopic, spectacle=myope, astigmatism=yes, tear=reduced]).
instance(20,class=yes, [age=presbyopic, spectacle=myope, astigmatism=yes, tear=normal]).
instance(21,class=no,[age=presbyopic, spectacle=hypermetrope, astigmatism=no, tear=reduced]).
instance(22,class=yes,[age=presbyopic, spectacle=hypermetrope, astigmatism=no, tear=normal]).
instance(23,class=no,[age=presbyopic, spectacle=hypermetrope, astigmatism=yes, tear=reduced]).
instance(24,class=no,[age=presbyopic, spectacle=hypermetrope, astigmatism=yes, tear=normal]).
%
```

รูปที่ 5.2 ข้อมูลที่ใช้เพื่อสร้างกฎการตัดสินใจ

บรรทัดแรกของข้อมูลเริ่มต้นด้วยเครื่องหมาย % หมายถึง comment โครงสร้างของไฟล์ข้อมูลจะแยกส่วนประกอบออกเป็นสองส่วนคือ ส่วนคำอธิบายแอททริบิวต์ (ได้แก่ส่วนข้อความ attribute ...) และส่วนแสดงรายละเอียดของข้อมูลแต่ละเรคคอร์ด (ได้แก่ส่วนข้อความ instance ...) ในส่วนที่อธิบายแอททริบิวต์ ภายในจะประกอบด้วยสองอาร์กิวเม้นต์ อาร์กิวเม้นต์แรกจะบอกชื่อแอททริบิวต์ อาร์กิวเม้นต์ที่สองเป็นลิสต์ของค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์นั้น ในส่วนของข้อมูลหรือ instance จะประกอบด้วยสามอาร์กิวเม้นต์ คือ หมายเลขของข้อมูล, ค่าคลาสของข้อมูล, ลิสต์ที่ระบุค่าของแต่ละแอททริบิวต์ โดยวิธีการระบุค่าจะใช้รูปแบบ attribute-value pair หรือ ชื่อแอททริบิวต์=ค่าของแอททริบิวต์ เมื่อสร้างข้อมูลในรูปแบบที่กำหนดเสร็จแล้วจะต้องบันทึกไฟล์ให้อยู่ในรูปแบบของโปรแกรม Prolog ที่มีส่วนขยาย (file extension) เป็น .pl เช่น

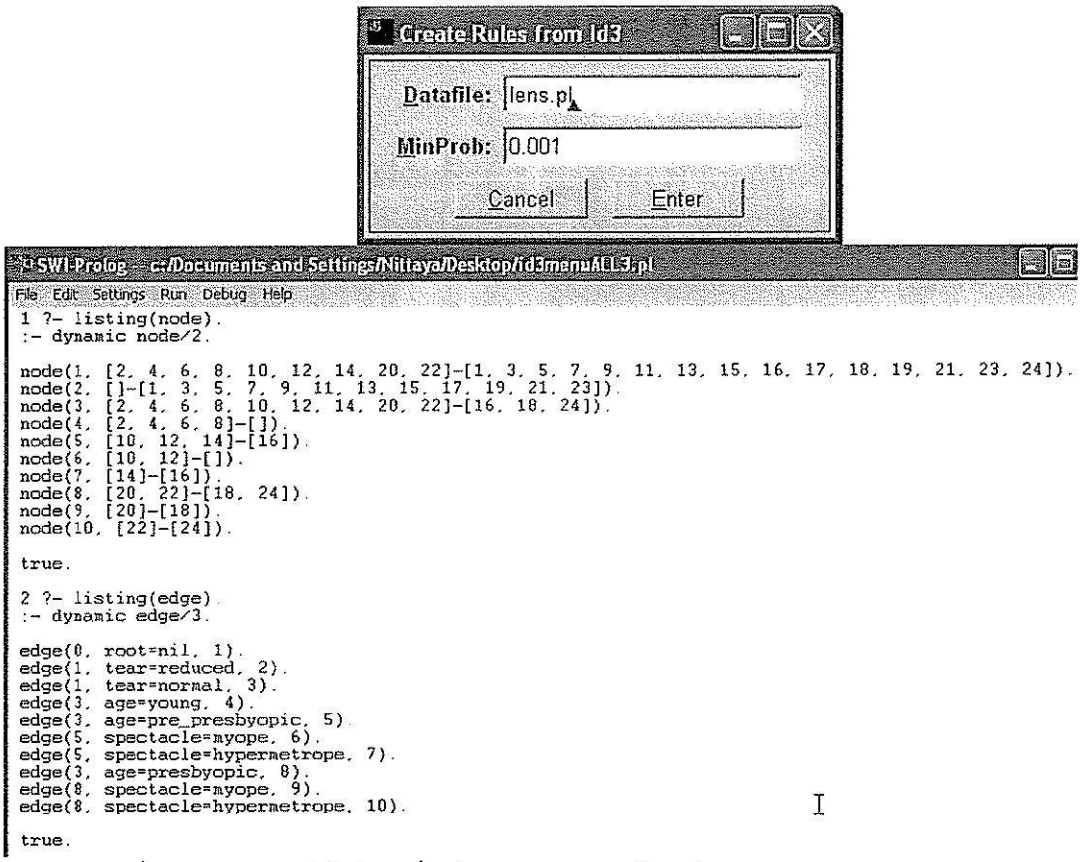
ตัวอย่างไฟล์ข้อมูลในรูปที่ 5.2 บันทึกอยู่ในชื่อ lens.pl ข้อมูลนี้จะใช้เป็นตัวอย่างประกอบคำอธิบายการทำงานของโปรแกรมวิเคราะห์ความรู้ และโปรแกรมรวบรวมและจัดการกับความรู้

โปรแกรมวิเคราะห์ความรู้

การทำงานของโปรแกรมในส่วนนี้ จะเริ่มหลังจากมีการสร้างโมเดลข้อมูลด้วยหลักการของการสร้างต้นไม้ตัดสินใจ (อัลกอริทึม ID3) และบันทึกโมเดลไว้ในเพรดิเคตชื่อ node และ edge คำสั่งต่างๆของโปรแกรมหลักแสดงได้ดังนี้

```
mainId3(Min) :-  init(AllAttr, EdgeList),  % initialize node and edge information
                 getnode(N),               % get node ID
                 create_edge_onelevel(N, AllAttr, EdgeList),  % create tree
                 addAllKnowledge,          % generate decision rules
                 selectRule(Min, Res),     % select top rules
                 writeln(Res),
                 tell('1.knb'),            % write selected decision rules to file
                 writeHeadF,               % transform to expert rules (head)
                 maplist(createRule1, Res),
                 nl, writeTailF,           % generate body of expert rules
                 told,                     % write expert rules to file and close it
                 writeln(endProcess).
```

โปรแกรมหลักชื่อ mainId3 จะรับอาร์กิวเม้นต์ Min ที่ระบุค่าความน่าจะเป็นขั้นต่ำที่ผู้ใช้ต้องการ จากนั้นเริ่มสร้างต้นไม้ตัดสินใจด้วยการเรียกใช้คำสั่งย่อยสามคำสั่งคือ init, getnode และ create_edge_onelevel คำสั่ง create_edge_onelevel จะมีการทำงานซ้ำแบบ recursive เพื่อสร้างต้นไม้ตัดสินใจคราวละหนึ่งระดับ การสร้างต้นไม้จะยุติเมื่อสามารถแยกข้อมูลที่มีสองคลาสปนกันให้เป็นข้อมูลคลาสเดียวกันได้ทั้งหมด หรือเมื่อไม่มีแอททริบิวต์ให้ใช้แยกข้อมูลได้อีกต่อไป จากข้อมูล lens.pl ในรูปที่ 5.2 เมื่อสร้างต้นไม้ตัดสินใจเสร็จ จะได้โครงสร้าง node และ edge ดังรูปที่ 5.3



รูปที่ 5.3 จอภาพให้ผู้ใช้ระบุชื่อข้อมูลและโมเดลข้อมูลในลักษณะ node และ edge

ตัวอย่างในรูปที่ 5.3 ระบุค่าความน่าจะเป็นขั้นต่ำ 0.001 จะทำให้ได้กฎการตัดสินใจจำนวนสามกฎ ดังต่อไปนี้

0.5 >> [tear=reduced] >> no,

0.166667 >> [tear=normal, age=young] >> yes,

0.0833333 >> [tear=normal, age=pre_presbyopic, spectacle=myope] >> yes

โครงสร้างของกฎการตัดสินใจจะประกอบด้วยรายละเอียดสามส่วน แต่ละส่วนแยกจากกันด้วยสัญลักษณ์ >> รายละเอียดส่วนแรกระบุความน่าจะเป็นที่กฎนี้สามารถครอบคลุมข้อมูล รายละเอียดส่วนที่สองระบุลักษณะหรือแอททริบิวต์ที่ใช้ประกอบการตัดสินใจ และส่วนสุดท้ายเป็นผลการวินิจฉัยของแพทย์ ค่า yes หมายถึงแนะนำให้คนไข้ที่มีปัญหาด้านสายตาใส่คอนแท็กเลนส์ได้ ค่า no หมายถึงไม่แนะนำให้ใส่คอนแท็กเลนส์ ตัวอย่างเช่นกฎข้อแรกระบุว่า ถ้าคนไข้มีอัตราการสร้างน้ำตาต่ำ (tear=reduced) แพทย์จะไม่แนะนำให้ใส่คอนแท็กเลนส์ ค่าความน่าจะเป็น 0.5 หมายถึงกฎนี้ครอบคลุมข้อมูล 0.5 หรือ 50% (นั่นคือกฎนี้สังเคราะห์ได้จากข้อมูลคนไข้ 12 ราย จากข้อมูลคนไข้ทั้งหมดจำนวน 24 ราย)

โปรแกรมรวบรวมและจัดการกับความรู้

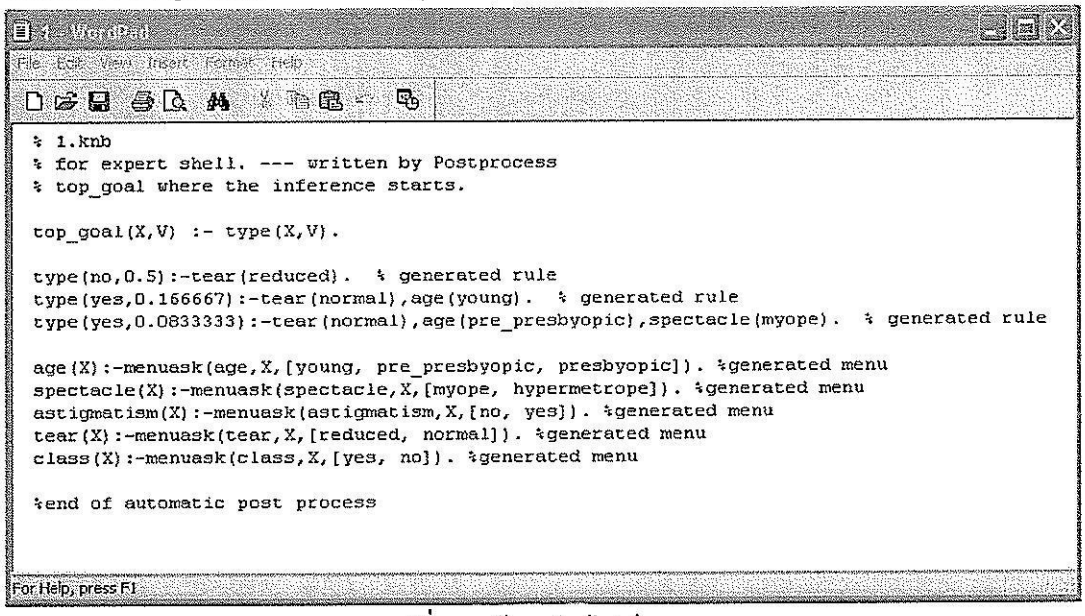
กฎการตัดสินใจที่ได้จากโปรแกรมวิเคราะห์ความรู้ จะถูกนำมาแปลงให้เป็นกฎเพื่อใช้ในระบบผู้เชี่ยวชาญ โปรแกรมแปลงกฎการตัดสินใจให้เป็นกฎประเภท expert rules แสดงได้ดังนี้

```
writeHeadF :-
        format('% 1.knb ~n% for expert shell. --- written by Postprocess'),
        format('~n% top_goal where the inference starts.~n'),
        format('~ntop_goal(X,V) :- type(X,V).~n').

writeTailF:-
        findall(_,(attribute(S,L),
                format('~n~w(X):-menuask(~w,X,~w). %generated menu',[S,S,L]))
               ,_),
        format('~n~n%end of automatic post process').
```

โปรแกรมแปลงกฎการตัดสินใจให้เป็นกฎประเภท consult rules แสดงได้ดังนี้

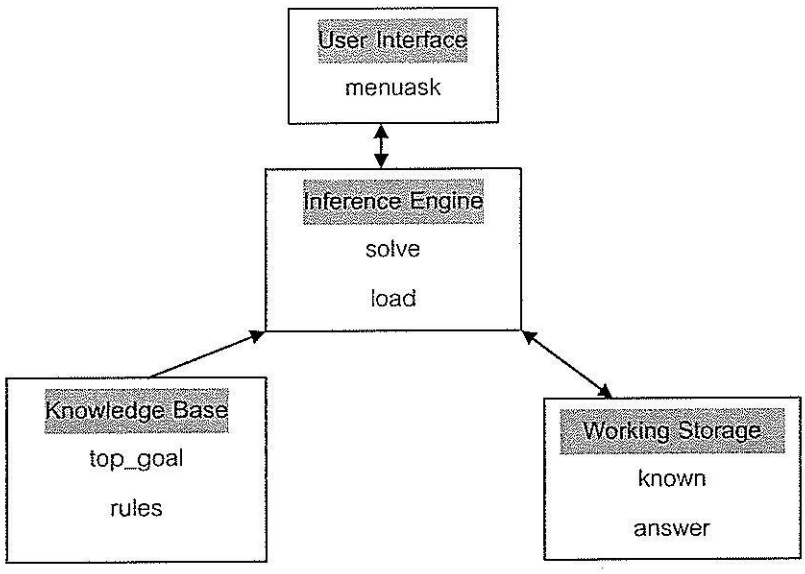
```
transform1([X=V], [Res]) :-
        atomic_list_concat([X,'(',V,')'], Res1),
        term_to_atom(Res, Res1),!.

transform1([X=V|T], [Res|T1]) :-
        atomic_list_concat([X,'(',V,')'], Res1),
        term_to_atom(Res, Res1),
        transform1(T, T1).

createRule1(I) :- I=Z>>X>>Y,
        transform1(X,BodyL),
        format('~ntype(~w,~w):-',[Y,Z]),
        myformat(BodyL) , write('  % generated rule'),!.

myformat([X]) :- write(X), write('.'),!.
myformat([H|T]) :- write(H), write(','), myformat(T).
```

เมื่อโปรแกรมที่ทำหน้าที่แปลงจากกฎการตัดสินใจ ให้เป็นกฎที่สามารถใช้งานได้กับระบบผู้เชี่ยวชาญ ทำการแปลงกฎทั้งหมดเสร็จสิ้น จะบันทึกผลลัพธ์ที่ได้ลงในไฟล์ชื่อ '1.knb' ซึ่งจากตัวอย่างข้อมูล lens.pl เมื่อแปลงกฎเสร็จจะได้ข้อมูลดังรูปที่ 5.4



รูปที่ 5.4 ข้อมูลในไฟล์ 1.knb

โปรแกรม expert system shell จะทำหน้าที่อนุมานจาก expert rules และ consult rules เพื่อแสดงเป็นคำแนะนำแก่ผู้ใช้ โครงสร้างของ expert system shell รวมถึงคำสั่งที่ใช้ในแต่ละส่วนแสดงดังรูปที่ 5.5



รูปที่ 5.5 โครงสร้างของ expert system shell

โครงสร้างของ expert system shell ประกอบด้วยส่วน User interface ทำหน้าที่โต้ตอบในลักษณะ interactive กับผู้ใช้ คำสั่งที่เขียนขึ้นเพื่อใช้กับการทำงานในส่วนนี้คือคำสั่ง menuask ส่วน Inference engine ทำหน้าที่รับข้อมูลบางส่วนจากผู้ใช้เพื่ออนุมานจาก expert rules ในฐานความรู้และหาคำแนะนำที่เหมาะสมที่สุดแสดงแก่ผู้ใช้ ส่วน Inference engine จะติดต่อกับ Working storage ในระหว่างที่ค้นหากฎที่เกี่ยวข้องกับสิ่งที่ผู้ใช้ถาม โดยในส่วน Working storage จะมีคำสั่ง known และ answer ในส่วนของ Inference engine จะประกอบด้วยคำสั่ง load ทำหน้าที่เรียกใช้ไฟล์ที่เกี่ยวข้องในฐานความรู้ และคำสั่ง solve ที่ผู้ใช้สั่งเพื่อให้ระบบผู้เชี่ยวชาญแสดงคำแนะนำ ในส่วนของฐานความรู้จะประกอบด้วยคำสั่งระดับบนสุดเรียกว่า top_goal และกฎต่างๆ เรียกว่า rules ในกรณีที่ผู้ใช้ต้องการทราบเหตุผลหรือคำอธิบาย ระบบผู้เชี่ยวชาญจะมีคำสั่ง why ให้เรียกใช้เพื่อแสดงคำอธิบายประกอบการให้คำแนะนำ

คำสั่งต่างๆใน expert system shell แสดงได้ดังต่อไปนี้

```
expertshell :-    greeting,
                  repeat,
                  write('expert-shell> '),
                  read(X),
                  do(X),
                  (X == quit ; X == 99),
                  writeln('>>>>Goodbye, see you later<<<<'), !.
greeting :-  write('This is the Easy Expert System shell.'), nl,
             native_help.
do(help) :- native_help, !.
do(load) :- load_kb, !.
do(solve) :- solve, !.
do(why) :- why,!.
do(quit).
do(99).
do(X) :-  write(X),
          write(' is not a legal command.'), nl,
          fail.
native_help :-    write('Type help. load. solve. why.  quit. or 99.'),nl,
                  write('at the prompt.'), nl.
```

```
load_kb :- write('Enter file name in single quotes (ex. "1.knb".): '),
           read(F),
           reconsult(F).

solve :-   retractall(known( _) ),
           retractall(answer(_,_)),
           top_goal(X,V),
           format('The answer is __~w__ with probability ~w',[X,V]),
           assert(answer(X,V)), nl.
solve :-   write('No answer found.'),nl.
menuask(Pred, Value, Menu) :-
           menuask(Pred,Menu),
           atomic_list_concat([Pred,'(',Value,')'],X),
           term_to_atom(T,X),known(T),!.
menuask(Pred,_) :-
           atomic_list_concat([Pred,'(','_',')'],X),  % check for recorded predicate
           term_to_atom(T,X),known(T),!.  % not ask again
menuask(Attribute,Menu):-
           nl, write('What is the value for '), write(Attribute), write('?'), nl,
           addchoice(Menu,MenuRes),
           writeln(MenuRes), write('Enter the  choice> '), read(C),
           member(C-V,MenuRes),
           (C=99 -> abort ; true ),
           atomic_list_concat([Attribute,'(',V,')'],X),
           term_to_atom(T,X),
           asserta(known(T)) .
why :-     answer(A,V),
           format('~nThe answer is ...~w... with probability = ~w.~n',[A,V]),
           findall( X , known(X),Result),
           writeln('The known storage are'),
           writeln(Result).
```