รหัสโครงการ SUT7-705-48-24-12

รายงานการวิจัย

การพัฒนาการทำเหมืองข้อมูลแบบจำแนก

(The development of classification data mining)

ผู้วิจัย

หัวหน้าโครงการ

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พ.ศ. 2548 และ 2549

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

พฤศจิกายน 2552

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณมหาวิทยาลัยเทคโนโลยีสุรนารีและสำนักงานคณะกรรมการวิจัยแห่งชาติ ที่ได้จัดสรรงบประมาณในการทำวิจัยให้ในปีงบประมาณ 2548 และ 2549 โครงการนี้ยังได้รับงบประมาณบางส่วน รวมถึงความร่วมมือในการดำเนินงานจากหน่วยปฏิบัติการวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ (Data Engineering and Knowledge Discovery -- DEKD -- Research Unit) ขอขอบคุณผู้ทรงคุณวุฒิที่ได้เสียสละเวลาทำหน้าที่ตรวจข้อเสนอโครงการ และตรวจร่างรายงานการวิจัยฉบับสมบูรณ์

บทคัดย่อภาษาไทย

องค์กรสมัยใหม่มักจะมีการสร้างข้อมูลอิเล็กทรอนิกส์ปริมาณมหาศาล เก็บบันทึกไว้ในฐานข้อมูลของหน่วยงาน ข้อมูลเหล่านี้เป็นวัตถุดิบที่มีค่าสำหรับกระบวนการอัตโนมัติที่เรียกว่าการค้นหาความรู้จากฐานข้อมูลหรือการทำเหมืองข้อมูล มีวัตถุประสงค์เพื่อสำรวจหาความรู้ที่มีประโยชน์ในการสนับสนุนการตัดสินใจระดับสูง ความรู้ที่สำรวจได้อาจจะเป็นตัวแบบของข้อมูลที่แสดงในรูปแบบสรุป ความสัมพันธ์ระหว่างข้อมูลที่แสดงในลักษณะกฎ หรือตัวแทนของกลุ่มข้อมูลที่แสดงด้วยค่ากลางของกลุ่ม ในหลายทศวรรษที่ผ่านมาได้มีความพยายามประยุกต์ใช้เทคโนโลยีเหมืองข้อมูลและการเรียนรู้ของเครื่องจักร เพื่อสังเคราะห์ความรู้จากข้อมูลที่มีอยู่ด้วยวิธีการโปรแกรมเชิงคำสั่งและเชิงวัตถุ ในงานวิจัยนี้ได้เสนอกรอบแนวคิดเชิงตรรกะเพื่อการพัฒนาระบบเหมืองข้อมูลที่ทำหน้าที่จำแนกประเภทของข้อมูล การพัฒนาระบบใช้วิธีการโปรแกรมเชิงประกาศที่มีพื้นฐานจากตรรกศาสตร์ เนื่องจากเป็นการโปรแกรมระดับสูงที่เขียนคำสั่งสั้น ทำให้สามารถลดเวลาการพัฒนาโปรแกรมของโปรแกรมเมอร์ และการโปรแกรมในเชิงตรรกะมีความเหมาะสมสำหรับงานที่ต้องเกี่ยวข้องอย่างมากกับเรื่องของความรู้ ผู้วิจัยได้พัฒนาอัลกอริทึมที่สามารถสร้างความรู้ที่มีค่าความน่าจะเป็นกำกับ เทคนิคการเรียนรู้ใช้วิธีสร้างต้นไม้ตัดสินใจที่สังเคราะห์ขึ้นจากข้อมูล ความรู้ที่ได้จะถูกแปลงให้อยู่ในรูปแบบกฎการตัดสินใจที่มีค่าความน่าจะเป็นกำกับ รายงานการวิจัยนี้แสดงวิธีการพัฒนาระบบเหมืองข้อมูลด้วยภาษาโปรล็อก และจากผลการทดสอบโปรแกรมด้วยข้อมูลมาตรฐานแสดงให้เห็นถึงความเป็นไปได้ที่จะเชื่อมต่อผลลัพธ์ที่อยู่ในรูปแบบกฎการตัดสินใจที่มีค่าความน่าจะเป็นกำกับ ประกอบเข้ากับระบบฐานความรู้

บทคัดย่อภาษาอังกฤษ

Modern organizations normally generate huge amount of data in electronic form stored in databases. These data are a valuable resource for automatic discovering of useful knowledge, known as knowledge discovery in databases or data mining, to support high-level decisions. Discovered knowledge may be patterns of data represented in summarized form, relationships among data represented as rules, or representatives of data subgroups represented by mean values. During the past decades there has been an increasing interest in devising database and machine learning technologies to automatically induce knowledge from stored data using imperative and object-oriented programming styles. In this research, we propose a data mining system based on a logical framework. The proposed logic-based system performs a data classification task. Declarative programming based on logic can greatly reduce the burden of programmers as it is a very high-level programming scheme suitable for the development of knowledge intensive tasks. We devise algorithms to generate probabilistic knowledge from the induced decision tree and infer decision from the induced probabilistic rules. The implementation of the proposed algorithms is demonstrated via a Prolog programming language. Experimental results on several data domains emphasize the simple form of knowledge representation and the potential of incorporating learning results as probabilistic knowledge in the knowledge-base system.

สารบัญ

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

ความสำคัญและที่มาของปัญหาการวิจัย

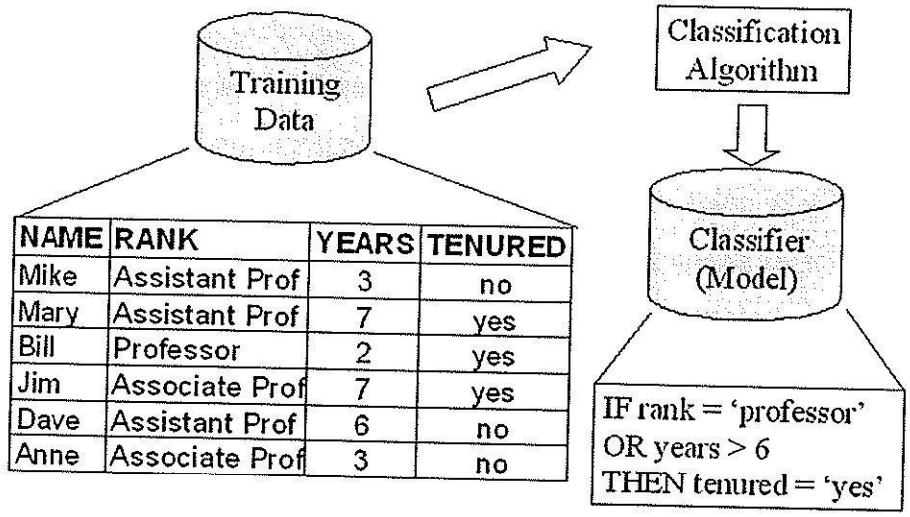
ในปัจจุบันเครื่องคอมพิวเตอร์และอุปกรณ์นำเข้าข้อมูล เช่น สแกนเนอร์ (scanner), เครื่องอ่านรหัสบาร์ (bar-code reader) มีใช้อย่างแพร่หลาย ประกอบกับอุปกรณ์เก็บข้อมูล เช่น ฮาร์ดดิสก์มีราคาถูกลง ทำให้มีข้อมูลที่ถูกบันทึกอยู่ในรูปแบบอิเล็กทรอนิกส์ปริมาณมหาศาล การใช้แรงงานผู้เชี่ยวชาญวิเคราะห์ข้อมูล เพื่อจะนำความรู้จากข้อมูลมาใช้ให้เกิดประโยชน์เป็นสิ่งที่แทบจะเป็นไปไม่ได้ ปัญหานี้จะเห็นได้ชัดเจนในกรณีของข้อมูลที่ได้รับจากดาวเทียมสำรวจสภาพอากาศและพื้นผิวโลก ที่มีการส่งข้อมูลขนาดเป็นเทอราไบต์ส่งมายังสถานีวิจัยทุกวัน

การใช้แรงงานผู้เชี่ยวชาญวิเคราะห์ข้อมูลด้วยโปรแกรมทางสถิติ เช่น SPSS เป็นงานที่ใช้เวลามากจนไม่สามารถนำผลการวิเคราะห์มาใช้ประโยชน์ทันเวลาได้ แนวทางที่จะช่วยแก้ไขปัญหาความล่าช้านี้ ก็คือ ทำให้กระบวนการวิเคราะห์ข้อมูลเป็นอัตโนมัติมากขึ้น ลดขั้นตอนการควบคุมและสั่งงานจากผู้เชี่ยวชาญให้น้อยลง โดยให้ระบบคอมพิวเตอร์ทำหน้าที่ค้นหาแพทเทิร์นโมเดลข้อมูล หรือแนวโน้มต่างๆที่น่าสนใจจากข้อมูลด้วยความสามารถของระบบเอง ทำให้เกิดเป็น *ระบบเหมืองข้อมูล* (data mining system)

การทำเหมืองข้อมูลจำแนกย่อยออกได้เป็นหลายประเภท เช่น การทำเหมืองข้อมูลแบบจำแนก (classification) แบบหาความสัมพันธ์ (association) แบบสรุปข้อมูล (summarization) แบบจัดกลุ่มข้อมูล (clustering) เป็นต้น การทำเหมืองข้อมูลแบบที่นิยมใช้แพร่หลายมากที่สุด (Tan et al. 2006) คือ แบบจำแนก เพราะสามารถนำโมเดลที่เป็นผลลัพธ์ของการทำเหมืองข้อมูลไปใช้ทำนายข้อมูลในอนาคต หรือข้อมูลที่ยังไม่ทราบประเภท ดังตัวอย่างในรูปที่ 1.1 แสดงการทำเหมืองข้อมูลกับข้อมูลอาจารย์มหาวิทยาลัย (Han and Kamber 2006) ข้อมูลที่ใช้ในการสร้างโมเดลเรียกว่าข้อมูลฝึก (training data) จากข้อมูลตัวอย่างในรูปที่ 1.1 ข้อมูลนี้มีจำนวนแอททริบิวต์ (attribute) สี่แอททริบิวต์ได้แก่ name, rank, years และ tenured วัตถุประสงค์ของการทำเหมืองข้อมูลในตัวอย่างนี้คือต้องการค้นหารูปแบบหรือโมเดลของอาจารย์มหาวิทยาลัยที่มีสถานภาพ tenured = yes ผลลัพธ์ที่ได้จะเป็นโมเดลข้อมูลที่เรียกว่า ตัวจำแนก (classifier) ซึ่งจะปีนโมเดลในลักษณะของกฎถ้า-แล้ว หรือ IF-THEN rules

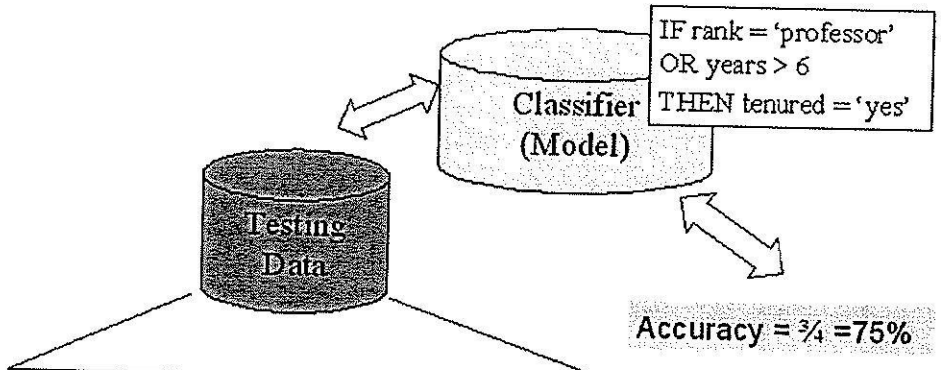
เนื่องจากกฎที่ได้นี้เกิดจากการเรียนรู้โดยอาศัยชุดข้อมูลฝึก ซึ่งอาจจะมีข้อผิดพลาดได้ ในกระบวนการทำเหมืองข้อมูลจึงต้องมีการทดสอบความถูกต้องของกฎหรือตัวจำแนกที่ได้ (รูปที่

1.2) ข้อมูลที่ใช้ในการทดสอบเรียกว่า ข้อมูลทดสอบ (testing data) ข้อมูลนี้จะเป็นข้อมูลที่แตกต่างจากข้อมูลฝึก แต่จะมีโครงสร้างของข้อมูล (จำนวนแอททริบิวต์ ชื่อแอททริบิวต์ และชนิดข้อมูลในแอททริบิวต์) เหมือนข้อมูลฝึก เมื่อโมเดลที่เป็นตัวจำแนกผ่านการทดสอบว่ามีความถูกต้องสูง ก็จะถูกนำไปใช้ทำนายกับข้อมูลใหม่ (ดังแสดงในรูปที่ 1.3) ตัวอย่างการทำนายข้อมูลใหม่ที่ไม่ทราบสถานภาพ tenured แสดงได้ว่า Professor Jeff ผู้ซึ่งมีอายุงาน 4 ปี คาดหมายได้ว่าจะมีสถานภาพ Tenured = yes



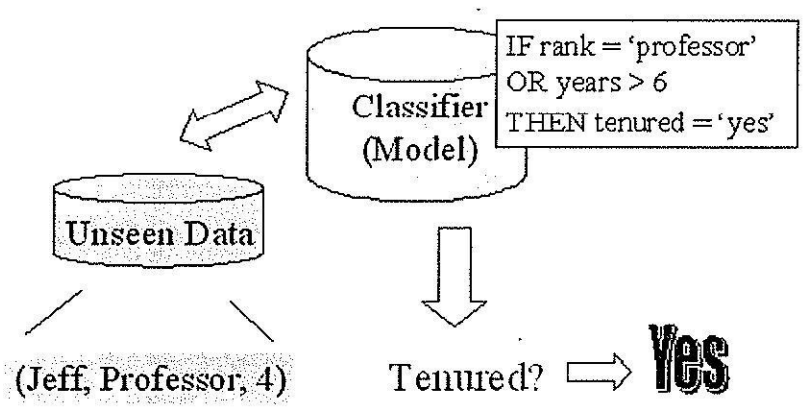
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

รูปที่ 1.1 ขั้นตอนการสร้างโมเดลที่เป็นตัวจำแนกข้อมูล



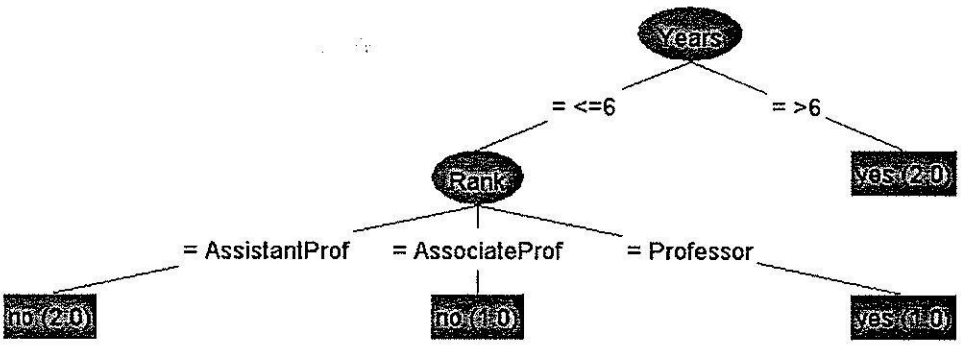
NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

รูปที่ 1.2 ขั้นตอนการทดสอบโมเดล



รูปที่ 1.3 การใช้โมเดลจำแนกข้อมูลที่เกิดขึ้นใหม่

การทำเหมืองข้อมูลแบบจำแนก หรือ classification data mining มีได้หลากหลายเทคนิคเช่น การใช้โครงข่ายประสาทเทียม (neural network) การวิเคราะห์จากข้อมูลใกล้เคียง (nearest neighbor) หรือการใช้ทฤษฎีความน่าจะเป็น (Bayes theorem) แต่เมื่อต้องการผลลัพธ์ที่เข้าใจได้ง่ายส่วนใหญ่จะใช้วิธีการสร้างต้นไม้ตัดสินใจเชิงอุปนัย (decision-tree induction) เพื่อจำแนกและอธิบายลักษณะเด่นของข้อมูลที่ถูกจำแนก (Quinlan 1993) จากข้อมูลฝึกในรูปที่ 1.1 เมื่อทำการค้นหาโมเดลด้วยวิธีการสร้างต้นไม้ตัดสินใจเชิงอุปนัยจะได้โมเดลดังรูปที่ 1.4



รูปที่ 1.4 โมเดลในลักษณะต้นไม้ตัดสินใจ

โมเดลในลักษณะของต้นไม้ตัดสินใจข้างต้นจะมีความหมายเทียบเท่ากับกฎการตัดสินใจ IF-THEN ดังต่อไปนี้

IF Years >6	THEN Tenured = yes
IF Years <=6 and Rank = Professor	THEN Tenured = yes
IF Years <=6 and Rank = AssociateProf	THEN Tenured = no
IF Years <=6 and Rank = AssociateProf	THEN Tenured = no

วิธีการสร้างต้นไม้ตัดสินใจเชิงอุปนัย มีขั้นตอนโดยทั่วไปดังนี้

(1) เลือกแอททริบิวต์ ที่ทำหน้าที่เป็นโหนดรากของต้นไม้ (root node)

(2) สร้างเส้นทางเชื่อมจากโหนดรากไปยังโหนดลูก จำนวนเส้นทางเชื่อมจะเท่ากับจำนวนค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์ที่เป็นโหนดราก

(3) ถ้าโหนดลูก เป็นกลุ่มของข้อมูลที่อยู่ในคลาสเดียวกันทั้งหมด ให้หยุดการสร้างต้นไม้ แต่ถ้าโหนดลูกมีข้อมูลของหลายคลาสปะปนกันอยู่ ต้องสร้าง subtree เพื่อจำแนกข้อมูลต่อไป โดยเลือกแอททริบิวต์มาทำหน้าที่เป็นโหนดรากของ subtree และทำซ้ำในขั้นตอนที่ 2 และ 3

โครงสร้างข้อมูลประเภทต้นไม้ตัดสินใจเริ่มถูกนำมาใช้อย่างแพร่หลายในงานด้านการเรียนรู้ของเครื่องจักร หรือ machine learning ดังปรากฏในเอกสารของ Breiman และคณะ (Breiman et al. 1984) และงานที่เป็นจุดเริ่มต้นสร้างความนิยมให้กับการใช้โครงสร้างต้นไม้ตัดสินใจ ได้แก่ การพัฒนาอัลกอริทึม ID3 ของ John Ross Quinlan (Quinlan 1986) และพัฒนาต่อมาเป็นโปรแกรม C4.5 (Quinlan 1993)

การสร้างต้นไม้ตัดสินใจเชิงอุปนัยเพื่อจำแนกข้อมูล ผู้ออกแบบอัลกอริทึมจะต้องคำนึงถึงกรณีข้อมูลรบกวนหรือข้อมูลที่มีความผิดพลาด เนื่องจากข้อมูลรบกวนจะทำให้โครงสร้างต้นไม้ผิดเพี้ยนไปจากที่ควรจะเป็น ซึ่งโดยทั่วไปจะส่งผลให้โครงสร้างต้นไม้มีขนาดใหญ่เกินไป เพราะพยายามจะขยายกิ่งออกไปให้สามารถอธิบายข้อมูลรบกวน การกระทำเช่นนี้จะนำไปสู่ลักษณะ overfitting (การพยายามปรับโครงสร้างให้เจาะจงอธิบายได้กับข้อมูลทุกรายการมากเกินไป)

ปัญหาเรื่อง overfitting ได้มีการศึกษาและนำเสนอไว้ในเอกสารจำนวนมาก (Schaffer 1993; Talmon and McNair 1992; Wolpert 1992) แนวทางการแก้ไขปัญหาโดยทั่วไปคือ

(1) ใช้วิธีการควบคุมขนาดของต้นไม้ในระหว่างขั้นตอนของการสร้างกิ่งที่แยกออกจากโหนดแม่ ไม่ให้มีจำนวนโหนดมากเกินไป (pre-pruning)

(2) ใช้วิธีการสร้างต้นไม้ให้เสร็จแล้วจึงตัดกิ่งย่อยของต้นไม้ (post-pruning) ที่คาดว่ากำลังพยายามสร้างขึ้นเพื่อให้ครอบคลุมข้อมูลรบกวน

แนวทางแก้ปัญหาแบบแรก นิยมใช้ในโปรแกรมสร้างต้นไม้ตัดสินใจเชิงอุปนัยในยุคแรกๆ (Freidman 1977) แต่ภายหลังมีการพบว่าแนวทางนี้ไม่เสถียรเท่าที่ควร จึงเปลี่ยนมาใช้วิธี post-pruning เทคนิค post-pruning ถูกเสนอขึ้นโดย Breiman และคณะ (1984) ภายหลัง Kim and Koehler (1994) ได้ศึกษาวิเคราะห์เงื่อนไขที่จะพิจารณาว่าการ pruning ในระดับใดจึงจะให้ต้นไม้ตัดสินใจที่มีความแม่นตรงสูงที่สุด เทคนิค post-pruning ที่ Quinlan ใช้ (Quinlan 1987) จะกัน

ข้อมูลส่วนหนึ่งไว้เพื่อประโยชน์ในการตรวจสอบประสิทธิภาพการทำนายของโมเดล และตัดบางส่วนของโครงสร้างต้นไม้ที่ไม่ก่อประโยชน์ออกไป โดยการตรวจสอบจะใช้วิธีหาความสัมพันธ์ในเชิงสถิติ

Gelford และคณะ (1993) ได้เสนออัลกอริทึมในการสร้างต้นไม้และมีการตัดกิ่งบางส่วนทิ้งด้วยการแบ่งข้อมูลออกเป็นสองส่วนเช่นกัน ส่วนแรกใช้ในการสร้างต้นไม้ ส่วนที่เหลือใช้ในการตรวจสอบเพื่อตัดส่วนที่ไม่ก่อประโยชน์ทิ้ง จากนั้นสลับบทบาทของข้อมูล (cross-validation) ให้ส่วนตรวจสอบถูกนำมาใช้ในการสร้างต้นไม้และส่วนที่ใช้สร้างไปทำหน้าที่ตรวจสอบ ทำซ้ำการสร้างและ prune โครงสร้างต้นไม้จนกว่าต้นไม้ที่ได้ไม่มีความแม่นตรงเพิ่มขึ้น ผู้พัฒนาพิสูจน์เพื่อรับรองว่าอัลกอริทึมจะลู่เข้าสู่ผลลัพธ์สุดท้าย คือได้ต้นไม้ตัดสินใจที่มีความแม่นตรงและขนาดไม่ใหญ่ซับซ้อนเกินไป

เทคนิคการ prune อีกแนวทางหนึ่งที่ Quinlan and Rivest (1989) ใช้คือ ใช้หลักการของ minimum descriptive length (MDL) ทั้งในการสร้างต้นไม้และตัดกิ่งของต้นไม้ แต่เทคนิคนี้ยังมีข้อบกพร่องที่ถูกค้นพบในภายหลังโดย Wallace and Pattrick (1993) อีกแนวทางหนึ่งที่ใช้ได้ผลในการ prune ต้นไม้ คือ ใช้เทคนิค dynamic programming (Bohanec and Bratko 1994)

จากการที่เทคนิค pruning มีได้หลายวิธี ทำให้มีผู้สนใจศึกษาวิเคราะห์เปรียบเทียบประสิทธิภาพของเทคนิค pruning ในบางประเภท ดังปรากฏในงานวิจัยของ Esposito et al. (1995); Cohen (1993); Mingers (1989) ในภายหลังได้มีนักวิจัยพยายามคิดค้นเทคนิค pruning ที่มีประสิทธิภาพมากขึ้น เช่นในงานวิจัยของ Cohen and Jensen (1997) ที่ประยุกต์ใช้ทฤษฎี multiple testing (MT) โดยพยายามให้หลีกเลี่ยงลักษณะ overfitting มากที่สุด

รายงานการวิจัยฉบับนี้เป็นการสร้าง classification mining engine ที่ใช้เทคนิคต้นไม้ตัดสินใจ เนื่องจากมีรูปแบบที่ผู้ใช้เข้าใจได้ง่าย และมีขั้นตอนที่ไม่ซับซ้อน เหมาะสมสำหรับผู้ใช้ทั่วไป โดยจะพิจารณาการจัดการกับข้อมูลรบกวนที่มีประสิทธิภาพด้วยหลักการพื้นฐานของค่าความน่าจะเป็น ทำให้ได้โครงสร้างต้นไม้ตัดสินใจในลักษณะ probabilistic tree รวมถึงมีการทำ pruning เพื่อให้โครงสร้างต้นไม้ตัดสินใจมีความซับซ้อนลดลง classification mining engine นี้ จะถูกรวมเข้าเป็นโมดูลหนึ่งของระบบเหมืองข้อมูล SUT Miner

วัตถุประสงค์ของการวิจัย

เพื่อออกแบบและพัฒนาโปรแกรมคอมพิวเตอร์เพื่อการจำแนกข้อมูล (classification data mining) ในงานทำเหมืองข้อมูล เทคนิคที่ใช้จะเป็นการสร้างต้นไม้ตัดสินใจเชิงอุปนัยที่ทนต่อข้อมูลรบกวน โดยพัฒนาเทคนิคที่จะช่วยให้การสร้างต้นไม้ตัดสินใจเชิงอุปนัย (decision-tree

induction) ทำงานกับข้อมูลที่มีข้อมูลรบกวนปะปนอยู่มากได้โดยไม่ทำให้โปรแกรมหยุดทำงานกลางคัน ผลลัพธ์หรือ output ที่ได้จากซอฟต์แวร์นี้จะเป็นโมเดลที่สามารถนำไปใช้ในการจำแนกข้อมูล เพื่อประโยชน์ในการทำนายข้อมูลหรือเหตุการณ์ในอนาคต นอกจากนี้โมเดลในลักษณะของ decision rules สามารถนำไปใช้เป็นความรู้ในระบบฐานความรู้ (knowledge-base system) หรือระบบผู้เชี่ยวชาญ (expert system) ได้

ขอบเขตของการวิจัย

โครงการวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนาซอฟต์แวร์เพื่อการทำเหมืองข้อมูล ประเภทการจำแนกข้อมูล (data classification) โดยใช้โครงสร้างต้นไม้ตัดสินใจเชิงอุปนัยที่ทนต่อข้อมูลรบกวน ในกระบวนการสร้างต้นไม้ตัดสินใจโหนดต่างๆ ที่ใช้ในการสร้างต้นไม้ตัดสินใจจะเป็นแอท ทริบิวต์เดี่ยว เพื่อให้การพิจารณาความซับซ้อนของโครงสร้างต้นไม้สามารถวัดได้จากจำนวนโหนดในต้นไม้ ค่าในแต่ละแอททริบิวต์ของชุดข้อมูลอาจจะปรากฏค่าเป็นจำนวนเลข (numeric) หรือเป็นค่าสัญลักษณ์ (nominal or categorical) ก็ได้ ในกรณีที่แอททริบิวต์เป็นค่าต่อเนื่อง (continuous) หรือตัวเลขที่มีการกระจายของค่าจำนวนมาก ก่อนจะมีการนำเข้าข้อมูลจะต้องมีการจัดช่วงของค่าด้วยวิธีการทำ discretization ซึ่งเป็นขั้นตอนที่อยู่นอกเหนือขอบเขตของงานวิจัยนี้

การพัฒนาโปรแกรมใช้ภาษา Prolog ซึ่งเป็นภาษาเชิงตรรกะ เนื่องจากมีแนวคิดและรูปแบบที่เหมาะสมสำหรับงานที่จะพัฒนาเป็นฐานความรู้ต่อไปในอนาคต ภาษา Prolog ที่ใช้ในงานวิจัยนี้ใช้มาตรฐานของ SWI Prolog (www.swi-prolog.org) ซึ่งเป็นซอฟต์แวร์ประเภทโอเพนซอร์ส (open-source software) ทำให้ผู้ใช้สามารถนำไปใช้งานหรือนำไปพัฒนาต่อได้โดยไม่มีปัญหาเรื่องลิขสิทธิ์ซอฟต์แวร์

การทดสอบโปรแกรมที่พัฒนาขึ้น จำแนกเป็นการทดสอบความถูกต้องของโมเดลและการทดสอบประสิทธิภาพของโปรแกรม โมเดลที่สร้างจากโปรแกรมจะเป็นต้นไม้ตัดสินใจที่มีลักษณะเช่นเดียวกับผลลัพธ์ที่ได้จากโปรแกรม ID3 (Quinlan 1986) แต่เนื่องจากโมเดลที่เรียนรู้จากข้อมูลฝึกขนาดใหญ่จะมีความซับซ้อนมาก จึงไม่สะดวกที่จะทดสอบเปรียบเทียบตัวโครงสร้างของตัวโมเดลโดยตรง แต่จะใช้วิธีทดสอบความแม่นตรงของโมเดลเมื่อใช้ทำนายคลาสของข้อมูลชุดทดสอบ จากนั้นเปรียบเทียบความแม่นตรงในการทำนายคลาสโดยโมเดลที่สร้างจากโปรแกรมที่พัฒนาขึ้นเทียบกับโมเดลจากโปรแกรม ID3 ในส่วนของการทดสอบประสิทธิภาพของโปรแกรมเป็นการทดสอบการลดขนาดของโมเดล โดยใช้โมเดลทั้งหมดที่มีค่าความน่าจะเป็นตั้งแต่ 0.0 ถึง 1.0 ทดสอบความแม่นตรงในการทำนาย (predicting accuracy) เปรียบเทียบกับความแม่นตรงที่ได้เมื่อโมเดลมีขนาดเล็กลง การลดขนาดจะลดเป็นสัดส่วนจากค่าความน่าจะเป็นสูงสุด (หรือ

max.Prob) โดยเลือกใช้สัดส่วนตั้งแต่ 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), จนถึงขนาด 1/10(max.Prob)

ประโยชน์ที่ได้รับจากการวิจัย

ซอฟต์แวร์ที่พัฒนาขึ้นนี้มีจุดมุ่งหมายเพื่อใช้ประโยชน์ในงานวิเคราะห์และจำแนกข้อมูลอัตโนมัติ สามารถใช้งานได้จริงกับข้อมูลที่รวบรวมจากงานประเภทต่างๆ ถึงแม้ข้อมูลที่รวบรวมมาจะมีข้อมูลที่บันทึกผิดพลาดซึ่งจัดเป็นข้อมูลรบกวนปะปนอยู่ ซอฟต์แวร์นี้ก็ยังสามารถสังเคราะห์โครงสร้างต้นไม้ที่นำไปใช้จำแนกประเภทข้อมูล (data classification) และโครงสร้างต้นไม้นี้อธิบายลักษณะเด่นของข้อมูลในแต่ละประเภทได้ ผลที่ได้ (output) จะมุ่งประโยชน์ไปที่การทำนายและคาดหมาย (prediction) ประเภทของข้อมูลที่จะเกิดขึ้นในอนาคต

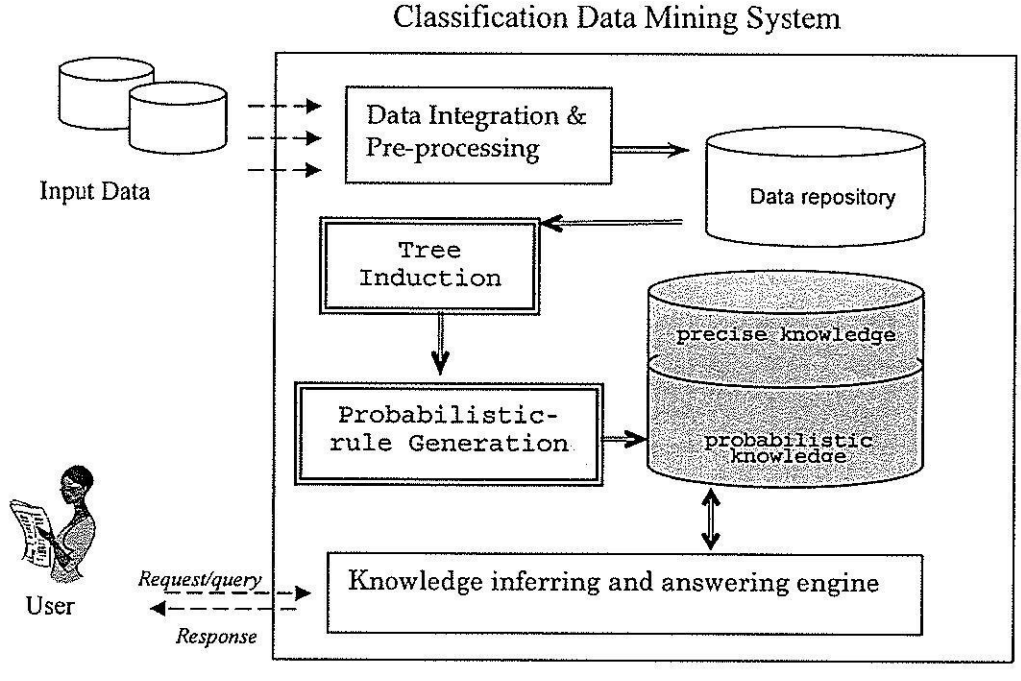
ดังนั้นซอฟต์แวร์เหมืองข้อมูลที่พัฒนาขึ้นนี้ จึงใช้ประโยชน์ได้กับทุกวงการที่เกี่ยวข้องกับงานวิเคราะห์ข้อมูล โดยจะช่วยให้งานวิเคราะห์ข้อมูลทำได้รวดเร็วขึ้น และผู้ใช้ไม่จำเป็นต้องเป็นผู้มีความเชี่ยวชาญเฉพาะในด้านการวิเคราะห์ข้อมูล ประโยชน์จึงเกิดกับทุกหน่วยงานที่มีการเก็บรวบรวมข้อมูลและวิเคราะห์ข้อมูล โดยเฉพาะในกรณีที่การรวบรวมข้อมูลใช้วิธีการส่งเจ้าหน้าที่ไปสอบถามและจดบันทึก เช่น การสำรวจสำมะโนประชากร การจดบันทึกอาจเกิดข้อผิดพลาดทำให้มีข้อมูลรบกวนเกิดขึ้น ซอฟต์แวร์ที่มีความสามารถจัดการกับข้อมูลรบกวนที่พัฒนาขึ้นนี้จะช่วยให้รับมือกับข้อมูลเช่นนี้ได้

ซอฟต์แวร์นี้ผู้วิจัยจะเผยแพร่เป็นสาธารณะผ่านเว็บไซต์ของหน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ (Data Engineering and Knowledge Discovery, DEKD) มหาวิทยาลัยเทคโนโลยีสุรนารี ผู้สนใจในทุกหน่วยงานสามารถนำไปใช้ในงานวิเคราะห์ข้อมูลได้ ซึ่งคาดว่าจะช่วยให้ประเทศชาติประหยัดเงินตราซื้อซอฟต์แวร์สำเร็จรูปเพื่อการวิเคราะห์ข้อมูลลงได้จำนวนมาก

บทที่ 2
วิธีดำเนินการวิจัย

กรอบแนวคิดของงานวิจัย

วัตถุประสงค์หลักของงานวิจัยนี้คือ การพัฒนาซอฟต์แวร์เพื่อสร้าง classifier หรือตัวจำแนก ซึ่งจะเป็นโมเดลที่สามารถนำไปใช้จำแนกประเภทข้อมูลหรือจำแนกค่าที่สนใจในข้อมูลที่จะเกิดขึ้นใหม่ในอนาคต การสร้างโมเดลใช้เทคนิคของต้นไม้ตัดสินใจเชิงอุปนัย และเพื่อป้องกันความบกพร่องของโมเดลในกรณีที่มีข้อมูลรบกวนปรากฏอยู่ในชุดข้อมูลฝึก งานวิจัยนี้จึงใช้การคำนวณค่าความน่าจะเป็นของการปรากฏข้อมูลในแต่ละกิ่งของโครงสร้างต้นไม้ตัดสินใจ เพื่อเรียงลำดับกฎการตัดสินใจ หรือ decision rules ที่แปลงมาจากต้นไม้ตัดสินใจตามสัดส่วนที่แต่ละกฎสามารถอธิบายข้อมูลได้ (หรือเรียกว่าค่า coverage ของกฎการตัดสินใจ) ผลลัพธ์สุดท้ายที่ได้จะเป็นเซตของ probabilistic decision rules ที่สามารถคัดเลือกไปเก็บไว้ในฐานความรู้เพื่อการใช้งานต่อไป แนวคิดโดยรวมของงานวิจัยนี้แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 กรอบแนวคิดของการออกแบบและพัฒนาซอฟต์แวร์เพื่อการทำเหมืองข้อมูลแบบจำแนก

ในกรอบเชิงแนวคิดของระบบเหมืองข้อมูลเพื่อการจำแนก (Classification data mining system) การทำงานของระบบจะต้องใช้ส่วนประกอบแรกคือ Data integration and Pre-processing

ทำหน้าที่รวมข้อมูลที่อาจจะถูกส่งมาจากหลายแหล่ง และจัดรูปแบบข้อมูลที่อาจแตกต่างกันให้อยู่ในมาตรฐานที่กำหนดไว้ (ในงานวิจัยนี้ใช้มาตรฐาน Horn clause ของภาษา Prolog นั่นคือข้อมูลแต่ละรายการหรือแต่ละเรคคอร์ดจะเป็นหนึ่งคลอสหรือหนึ่งข้อความตามรูปแบบ fact ของ Prolog) และถ้าข้อมูลมีขนาดใหญ่มากผู้ใช้งานระบบสามารถเรียกใช้เครื่องมือ Sampling ในส่วน Pre-processing เพื่อคัดเลือกข้อมูลตัวแทนมาใช้ในขั้นตอนการเรียนรู้เพื่อสร้างโมเดลของข้อมูล ในกรณีที่ข้อมูลมีแอททริบิวต์หรือ feature มากเกินจำเป็น ผู้ใช้งานระบบสามารถสั่งผ่านฟังก์ชัน Feature selection ให้คัดเลือกเฉพาะ feature ที่คาดว่าจะเป็นประโยชน์ในการเรียนรู้เพื่อสร้างโมเดล ข้อมูลที่ผ่านการทำงานของส่วน Data integration and Pre-processing จะถูกเก็บไว้ในฐานข้อมูลในลักษณะของ Prolog data file (หมายถึงไฟล์ข้อมูลที่มีส่วนขยายเป็น .pl)

ส่วนประกอบส่วนที่สองคือ Tree induction ซึ่งเป็นส่วน mining engine ที่ทำหน้าที่อ่านข้อมูลและสร้างโมเดลข้อมูลในลักษณะของต้นไม้ตัดสินใจ เมื่ออ่านข้อมูลจนจบไฟล์และการสร้างต้นไม้ตัดสินใจเสร็จสิ้น โครงสร้างต้นไม้ทั้งต้นที่แทนด้วยโหนดและกิ่งต่างๆของโหนดตั้งแต่โหนดรากจนกระทั่งถึงโหนดใบ จะถูกส่งต่อไปยังส่วนที่สามที่ทำหน้าที่แปลงจากต้นไม้ตัดสินใจให้เป็นกฎการตัดสินใจ

ส่วนของ Probabilistic-rule generation จะทำหน้าที่ traverse ไปตามกิ่งต่างๆของต้นไม้พร้อมทั้งคำนวณสัดส่วนของข้อมูลที่ปรากฏในแต่ละกิ่ง เพื่อกำหนดเป็นค่าความน่าจะเป็นในการครอบคลุมข้อมูลของแต่ละกฎการตัดสินใจ ในขั้นตอนนี้ผู้ใช้สามารถกำหนดค่าเกณฑ์ขั้นต่ำของค่าความน่าจะเป็นเพื่อคัดเลือกเฉพาะกฎที่อธิบาย (หรือ cover) ข้อมูลได้มาก กฎการตัดสินใจที่ผ่านเกณฑ์คัดเลือกจะถูกบันทึกไว้ในฐานความรู้ เพื่อการใช้งานต่อไปในกระบวนการสอบถามความรู้ของระบบผู้เชี่ยวชาญหรือระบบฐานความรู้ แต่เนื่องจากความรู้ที่ได้จากการทำเหมืองข้อมูลเป็นความรู้ที่สังเคราะห์จากข้อมูลที่เกิดขึ้นจริงในอดีต ข้อมูลอาจจะมีข้อผิดพลาดและข้อมูลเหล่านี้อาจไม่แสดงรูปแบบหรือโมเดลที่ถูกต้องครบถ้วนได้ทั้งหมด จึงต้องแยกความรู้ที่ได้จากการทำเหมืองข้อมูลไว้ในฐานความรู้ส่วนที่เรียกว่า probabilistic knowledge ในส่วนของความรู้ที่ได้จากการสัมภาษณ์ผู้เชี่ยวชาญซึ่งมักจะเป็นความรู้ที่มีความถูกต้องสูงจะแยกไว้เป็นส่วน precise knowledge

เมื่อมีการสอบถามหรือปรึกษาระบบผู้เชี่ยวชาญ ส่วนประกอบที่เรียกว่า Knowledge inferring and answering engine จะทำหน้าที่ติดต่อกับฐานความรู้ทั้งในส่วน precise knowledge และ probabilistic knowledge เพื่ออนุมานคำตอบที่ใกล้เคียงกับข้อสอบถามของผู้ใช้ให้มากที่สุด

การออกแบบและพัฒนาซอฟต์แวร์ในระบบทำเหมืองข้อมูลเพื่อการจำแนกนี้ ผู้วิจัยจะพิจารณาเฉพาะในส่วนประกอบหลักสองส่วนคือ Tree induction และ Probabilistic-rule generation ทั้งนี้เนื่องจากในส่วนของ Data integration and Pre-processing และ Knowledge inferring and answering engine อยู่นอกขอบเขตของการพัฒนาระบบเหมืองข้อมูลเพื่อการจำแนก (รายละเอียด

ของซอฟต์แวร์ทั้งสองส่วนจะปรากฏในรายงานการวิจัยเรื่อง "วิธการเตรียมข้อมูลอัตโนมัติก่อนการทำเหมืองข้อมูล" และ "การประมวลผลหลังกระบวนการทำเหมืองข้อมูล" ตามลำดับ โดยรายงานการวิจัยทั้งหมดนี้จะเป็นส่วนประกอบของชุดโครงการวิจัย "SUT Miner: ระบบเหมืองข้อมูลที่มีประสิทธิภาพ")

การออกแบบอัลกอริทึมเพื่อการทำเหมืองข้อมูลแบบจำแนก

ในการออกแบบอัลกอริทึมที่เป็นโมดูลหลักของระบบเหมืองข้อมูลเพื่อการจำแนก จะออกแบบสอดคล้องกับกรอบแนวคิดของระบบที่แสดงดังรูปที่ 2.1 ดังนั้นระบบนี้จะประกอบด้วยสองโมดูลคือ Tree-induction และ Probabilistic-rule generation รายละเอียดของแต่ละอัลกอริทึมแสดงได้ดังต่อไปนี้

โมดูล Tree-induction

อัลกอริทึมในการสร้างต้นไม้ตัดสินใจประกอบด้วยสองขั้นตอนหลักคือการอ่านข้อมูลทั้งหมดพร้อมทั้งกำหนดค่าเริ่มต้นตัวนับหมายเลขโหนด และขั้นตอนการสร้างต้นไม้ โดยในขั้นตอนสร้างต้นไม้นี้จะวนทำซ้ำจนกระทั่งแยกข้อมูลได้สมบูรณ์

Algorithm 1 Tree-induction

Input: a data set formatted as Prolog clauses
Output: a decision tree with node/2 and edge/3 structures

---

(1) Initialization

  (1.1) Clear temporary knowledge base (KB) by removing all information regarding the predicates node/2, edge/3 and current_node/1

  (1.2) Set node counter = 0

  (1.3) Scan data set to get information about data attributes, positive instances, negative instances, total data instances

(2) Building tree

  (2.1) Increment node counter

  (2.2) Repeat steps 2.2.1-2.2.4 until there is no more attributes left for creating decision attributes

    (2.2.1) Compute Info value of each candidate attribute
    (2.2.2) Choose the attribute that yields minimum Info to be decision node at current tree level
    (2.2.3) Assert edge/3 and node/2 information into KB
    (2.2.4) Split data instances along node branches

  (2.3) Repeat steps 2.1 and 2.2 until the lists of positive and negative instances are empty

  (2.4) Output tree structure containing node/2 and edge/3 predicates

---

อัลกอริทึม Tree-induction ทำหน้าที่รับข้อมูลเข้าในลักษณะของ Prolog clauses จากนั้นใช้ค่าแอททริบิวต์ที่ปรากฏในข้อมูลแต่ละรายการ สร้างโครงสร้าง node และ edge ของต้นไม้ตัดสินใจ เมื่ออัลกอริทึมอ่านข้อมูลเข้าครบหมดแล้วจะเริ่มกระบวนการสร้างโครงสร้างต้นไม้ (ขั้นตอนที่ 2 ของอัลกอริทึม Tree-induction) ในขั้นตอนย่อยที่ 2.2 เป็นการคัดเลือกแอททริบิวต์ที่ให้ค่า Gain สูงที่สุด โดยค่า Gain นี้จะคำนวณจากฟังก์ชัน Info ที่ทำหน้าที่คำนวณสัดส่วนของจำนวนข้อมูลที่เป็นคลาส positive -- P(p) และสัดส่วนของจำนวนข้อมูลที่เป็นคลาส negative -- P(n) ตัวแปร p หมายถึงจำนวน positive instances (เช่นข้อมูลที่มีค่า class = yes) และ n หมายถึงจำนวน negative instances (เช่นข้อมูลที่มีค่า class = no) สูตรคำนวณค่า Info และ Gain (Quinlan 1993) แสดงได้ดังนี้

$$Info(P(p),P(n)) = -P(p)\log_2 P(p) - P(n)\log_2 P(n)$$

$$Gain(Attribute) = Info(\frac{p}{p+n},\frac{n}{p+n}) - \sum_{i=1}^{v}\frac{p_i+n_i}{p+n} Info(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i})$$

ฟังก์ชัน Info(P(p), P(n)) จะให้ค่าสัดส่วนการปะปนกันของข้อมูลทั้งที่เป็น positive instances และ negative instances แอททริบิวต์ที่ให้ค่า Gain สูงที่สุดหมายถึงแอททริบิวต์นั้นเมื่อนำมาใช้เป็นโหนดของต้นไม้ตัดสินใจสามารถแยก positive instances ออกจาก negative instances ได้ดีที่สุด แอททริบิวต์ที่ถูกเลือกจะถูกบันทึกค่าเป็น edge ของต้นไม้ โครงสร้างของ node และ edge กำหนดไว้ดังนี้

node(nodeID, [Positive_Instances]-[Negative_Instancess])

edge(ParentNode, EdgeLabel, ChildNode)

โหนดแต่ละโหนดจะมีหมายเลขกำกับพร้อมทั้งระบุข้อมูล (ข้อมูลแต่ละรายการจะมีหมายเลขกำกับเช่นกัน) ในโหนดนั้นทั้งในส่วนที่เป็น positive instances และ negative instances ในส่วนของโครงสร้างกิ่งหรือ edge จะระบุข้อมูลสามส่วนคือ หมายเลขโหนดที่เป็นโหนดแม่ของกิ่งนั้น, ชื่อของ edge (คือค่าของแอททริบิวต์ที่ถูกเลือกเป็นโหนดแม่ ดังนั้นจำนวน edge ที่แตกออกจากโหนดจะมีจำนวนเท่ากับจำนวนค่าที่เป็นไปได้ของแอททริบิวต์นั้น) และหมายเลขของโหนดที่เป็นโหนดลูก

กระบวนการสร้างโหนดและ edge จะดำเนินซ้ำไปเช่นนี้จนกว่าข้อมูลที่เป็น positive instances และ negative instances จะถูกแยกออกจากกันโดยสมบูรณ์ หรือจนกระทั่งไม่มีแอททริบิวต์เหลือให้ใช้สร้างต้นไม้ได้อีกต่อไป เมื่อเสร็จกระบวนการในขั้นตอนนี้ข้อมูล node และ edge ทั้งหมดที่บันทึกไว้จะถูกส่งต่อไปยังโมดูล Probabilistic-rule generation

โมดูล Probabilistic-rule generation

อัลกอริทึมในส่วนนี้ทำหน้าที่รับข้อมูล node และ edge จากโมดูลที่ทำหน้าที่สร้างต้นไม้ตัดสินใจ รวมทั้งรับค่าความน่าจะเป็นขั้นต่ำจากผู้ใช้ ผลลัพธ์สุดท้ายที่ได้จากอัลกอริทึมคือตัวจำแนกข้อมูล ที่แสดงในลักษณะของกฎการตัดสินใจ หรือ decision rules โดยกฎนี้จะเรียงลำดับตามค่าความน่าจะเป็นสูงสุดลดหลั่งลงมาจนกระทั่งถึงกฎการตัดสินใจที่มีค่าความน่าจะเป็นต่ำสุดตามที่ผู้ใช้กำหนด รายละเอียดขั้นตอนต่างๆในอัลกอริทึมแสดงได้ดังต่อไปนี้

Algorithm 2 Probabilistic-rule generation

Input: a decision tree with node/2 and edge/3 structures, and
a probability threshold

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

(1) Traverse tree from a root node to each leaf node
  (1.1) Collect edge information and count number of data instances
  (1.2) Compute probability as a proportion
  (number of instances at leaf node) / (total data instances in a data set)
  (1.3) Assert a rule containing a triplet
  <attribute-value pair, class, probability value> into temporary KB
(2) Sort rules in the KB in descending order according to the rules' probability
(3) Remove rules that have probability less than the specified threshold
(4) Assert selected rules into the KB and return KB as an output

---

ขั้นตอนแรกของอัลกอริทึมเป็นการอ่านข้อมูลจาก node และ edge เรียงลำดับตามค่าหมายเลขโหนด โดยเริ่มจากโหนดรากที่เป็นโหนดหมายเลขศูนย์ไล่ตามแต่ละกิ่งหรือ edge ลงมาจนกระทั่งถึงโหนดใบ ในขณะที่อ่านข้อมูลในแต่ละกิ่งจะนับจำนวนข้อมูลที่ถูกแยกย่อยลงมาในแต่ละกิ่งเพื่อคำนวณค่าความน่าจะเป็นในการครอบคลุมข้อมูลของกฎการตัดสินใจ ค่าความน่าจะเป็นนี้เป็นสัดส่วนระหว่างจำนวนข้อมูลที่โหนดใบหารด้วยจำนวนข้อมูลทั้งหมด (นั่นคือหารด้วยข้อมูลที่โหนดราก) จากนั้นบันทึกกฎการตัดสินใจที่แต่ละกฎมีส่วนประกอบ 3 ส่วน คือ ค่าของแอททริบิวต์ที่เป็นปัจจัยประกอบการตัดสินใจ, คลาสของข้อมูลที่เป็นผลของการตัดสินใจ และ ค่าความน่าจะเป็นของกฎการตัดสินใจ

เมื่อแปลงโครงสร้างต้นไม้ตัดสินใจเป็นกฎการตัดสินใจได้ครบในทุกกิ่ง และทุกโหนดใบแล้ว จะมีการเรียงลำดับกฎตามค่าความน่าจะเป็น (ขั้นตอนที่ 2 ตามอัลกอริทึม) โดยเรียงจากค่ามากสุดลงมาหาค่าน้อย ต่อจากนั้นจะตัดทิ้งกฎที่มีค่าความน่าจะเป็นต่ำกว่าเกณฑ์ที่ผู้ใช้กำหนด (ขั้นตอนที่ 3 ตามอัลกอริทึม) และในขั้นตอนสุดท้ายเป็นการบันทึกกฎการตัดสินใจลงในฐานความรู้

การพัฒนาโปรแกรมเพื่อการทำเหมืองข้อมูลแบบจำแนก

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

รูปแบบของข้อมูล

ข้อมูลฝึกที่จะเป็นอินพุทของโปรแกรม จะมีลักษณะเป็นข้อความที่อยู่ในรูปแบบของข้อความที่เป็นจริง หรือ fact ตัวอย่างของข้อมูลแสดงได้ดังรูปที่ 2.2

```
%% Data weather
%
%attributes: names and their possible values
%
attribute( outlook,     [sunny, overcast, rainy] ).
attribute( temperature, [hot, mild, cool]        ).
attribute( humidity,    [high, normal]           ).
attribute( windy,       [true, false]            ).
attribute( class,       [yes, no]                ).
%data
instance(1, class=no, [outlook=sunny, temperature=hot, humidity=high, windy=false]).
instance(2, class=no, [outlook=sunny, temperature=hot, humidity=high, windy=true]).
instance(3, class=yes, [outlook=overcast, temperature=hot, humidity=high, windy=false]).
instance(4, class=yes, [outlook=rainy, temperature=mild, humidity=high, windy=false]).
instance(5, class=yes, [outlook=rainy, temperature=cool, humidity=normal, windy=false]).
instance(6, class=no, [outlook=rainy, temperature=cool, humidity=normal, windy=true]).
instance(7, class=yes, [outlook=overcast, temperature=cool, humidity=normal, windy=true]).
instance(8, class=no, [outlook=sunny, temperature=mild, humidity=high, windy=false]).
instance(9, class=yes, [outlook=sunny, temperature=cool, humidity=normal, windy=false]).
instance(10, class=yes, [outlook=rainy, temperature=mild, humidity=normal, windy=false]).
instance(11, class=yes, [outlook=sunny, temperature=mild, humidity=normal, windy=true]).
instance(12, class=yes, [outlook=overcast, temperature=mild, humidity=high, windy=true]).
instance(13, class=yes, [outlook=overcast, temperature=hot, humidity=normal, windy=false]).
instance(14, class=no, [outlook=rainy, temperature=mild, humidity=high, windy=true]).
%
```

รูปที่ 2.2 ตัวอย่างไฟล์ข้อมูลที่จะนำเข้ายังโปรแกรมเพื่อสร้างกฎการตัดสินใจ

บรรทัดแรกของข้อมูลเริ่มต้นด้วยเครื่องหมาย % หมายถึง comment โครงสร้างของไฟล์ข้อมูลจะแยกส่วนประกอบออกเป็นสองส่วนคือ ส่วนคำอธิบายแอททริบิวต์ (ได้แก่ส่วนข้อความ attribute ...) และส่วนแสดงรายละเอียดของข้อมูลแต่ละเรคคอร์ด (ได้แก่ส่วนข้อความ instance ...) ในส่วนที่อธิบายแอททริบิวต์ ภายในจะประกอบด้วยสองอาร์กิวเม้นต์ อาร์กิวเม้นต์แรกจะบอกชื่อแอททริบิวต์ อาร์กิวเม้นต์ที่สองเป็นลิสต์ของค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์นั้น

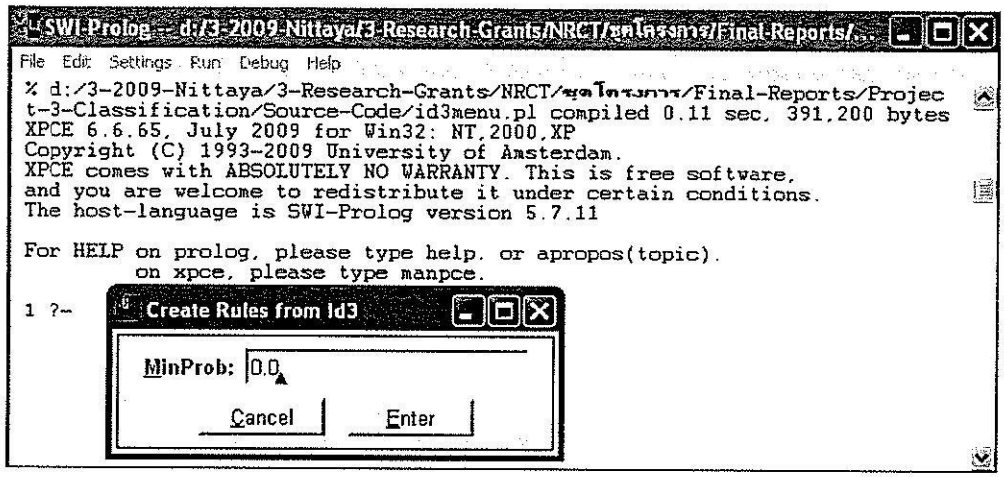
ในส่วนของข้อมูลหรือ instance จะประกอบด้วยสามอาร์กิวเม้นต์ คือ หมายเลขของข้อมูล, ค่าคลาสของข้อมูล, ลิสต์ที่ระบุค่าของแต่ละแอททริบิวต์ โดยวิธีการระบุค่าจะใช้รูปแบบ attribute-value pair หรือ ชื่อแอททริบิวต์=ค่าของแอททริบิวต์ เมื่อสร้างข้อมูลในรูปแบบที่กำหนดเสร็จแล้วจะต้องบันทึกไฟล์ให้อยู่ในรูปแบบของโปรแกรม Prolog ที่มีส่วนขยาย (file extension) เป็น .pl เช่น ตัวอย่างไฟล์ข้อมูลในรูปที่ 2.2 บันทึกอยู่ในชื่อ weather.pl ข้อมูลนี้ (Quinlan 1993) เป็นข้อมูลที่รวบรวมการตัดสินใจของนักกอล์ฟในสิบสี่วันที่ผ่านมาว่าวันใดเขาตัดสินใจออกไปเล่นกอล์ฟ (class = yes) และวันใดที่ไม่ออกไปเล่น (class = no) แอททริบิวต์ที่ใช้ประกอบการตัดสินใจประกอบด้วยสภาพท้องฟ้า (outlook) อุณหภูมิ (temperature) ความชื้น (humidity) และสภาพลมแรง (windy)

โปรแกรม Tree-induction

ระบบเหมืองข้อมูลเพื่อการจำแนก เริ่มต้นทำงานจากโปรแกรมแรก (ชื่อ id3menu) ที่เป็นส่วนสร้าง GUI โต้ตอบกับผู้ใช้ (การพัฒนาส่วน GUI ใช้ซอฟต์แวร์ XPCE ที่ถูกผนวกอยู่ใน SWI Prolog) จากนั้นจะเริ่มการสร้างต้นไม้ตัดสินใจด้วยการเรียกคำสั่ง callId3 คำสั่ง callId3 จะทำหน้าที่เรียกคำสั่ง mainId3 โดยระบุอาร์กิวเม้นต์เป็นค่าความน่าจะเป็นขั้นต่ำ ถ้าผู้ใช้ไม่ระบุค่านี้ ระบบจะกำหนดให้เป็น 0.0 ตัวอย่างการเรียกใช้ระบบด้วยคำสั่ง 'id3menu' แสดงได้ดังรูปที่ 2.3

```
id3menu:-
      new(Dialog,dialog('Create Rules from Id3')),
      send_list(Dialog, append,
      [ new(Per,text_item(minProb,'0.0')),
        button(cancel, message(Dialog, destroy)),
        button(enter, and(message(@prolog,callId3,Per?selection ),
        message(Dialog, destroy))) ]),
      send(Dialog, open).

callId3(Per):- term_to_atom(Per1,Per), mainId3(Per1).
```



รูปที่ 2.3 หน้าจอหลักของการเรียกใช้โปรแกรมสร้างต้นไม้ตัดสินใจ

คำสั่ง mainID3 ทำหน้าที่เรียกใช้ init ซึ่งเป็นส่วนสร้าง node และ edge รวมถึงตัวนับหมายเลขโหนด กระบวนการสร้างต้นไม้ตัดสินใจจะไปสิ้นสุดที่คำสั่ง addKnowledge ที่ทำหน้าที่บันทึกข้อมูล node และ edge ลงในฐานข้อมูลชั่วคราว เมื่อถึงขั้นตอนนี้ผู้ใช้สามารถตรวจสอบข้อมูลเกี่ยวกับ node และedge ได้โดยการสั่ง listing(node) และ listing(edge) ซึ่งจะได้รายละเอียดปรากฏดังรูปที่ 2.4

```
mainId3(Min) :-   init(AllAttr, EdgeList),
                  getnode(N),
                  create_edge_onelevel(N, AllAttr, EdgeList),
                  addKnowledge,
                  selectRule(Min, Res),
                  maplist(writeln, Res).
```

```
SWI-Prolog -- d:/3-2009-Nittaya/3-Research-Grants/NRCT/ชุดโครงการ/Final-Reports/...
File  Edit  Settings  Run  Debug  Help
1 ?- listing(node).
:- dynamic node/2.

node(1, [3, 4, 5, 7, 9, 10, 11, 12, 13]-[1, 2, 6, 8, 14]).
node(2, [9, 11]-[1, 2, 8]).
node(3, []-[1, 2, 8]).
node(4, [9, 11]-[]).
node(5, [3, 7, 12, 13]-[]).
node(6, [4, 5, 10]-[6, 14]).
node(7, []-[6, 14]).
node(8, [4, 5, 10]-[]).

true.

2 ?- listing(edge).
:- dynamic edge/3.

edge(0, root=nil, 1).
edge(1, outlook=sunny, 2).
edge(2, humidity=high, 3).
edge(2, humidity=normal, 4).
edge(1, outlook=overcast, 5).
edge(1, outlook=rainy, 6).
edge(6, windy=true, 7).
edge(6, windy=false, 8).

true.
```

รูปที่ 2.4 รายละเอียดที่แสดงจากการใช้คำสั่ง listing(node) และ listing(edge)

ข้อมูล node หมายเลขหนึ่งระบุว่าที่โหนดเริ่มแรกนี้มีข้อมูลที่เป็นกลุ่ม positive คือข้อมูลหมายเลข [3,4,5,7,9,10,11,12,13] ปะปนอยู่กับข้อมูลที่เป็นกลุ่ม negative คือข้อมูลหมายเลข [1,2,6,8,14] เมื่อพิจารณาประกอบกับข้อมูล edge บรรทัดที่สอง, บรรทัดที่ห้า และบรรทัดที่หก ที่ระบุว่า

edge(1, outlook=sunny, 2)

edge(1, outlook=overcast, 5)

edge(1, outlook=rainy, 6)

ทำให้เราทราบว่าแอททริบิวต์ outlook ถูกเลือกใช้เป็นโหนดในระดับแรกสุด และมีกิ่งแตกออกไปสามกิ่ง คือ กรณีแอททริบิวต์นี้มีค่าเป็น sunny, overcast และ rainy โดยกรณี outlook=sunny จากโหนดแม่ที่เป็นหมายเลขหนึ่งจะมีเส้นเชื่อมไปยังโหนดลูกที่เป็นโหนดหมายเลขสอง ในทำนองเดียวกันกรณี outlook=overcast จากโหนดแม่ที่เป็นหมายเลขหนึ่งจะมีเส้นเชื่อมไปยังโหนดลูกที่เป็นโหนดหมายเลขห้า และสุดท้ายกรณี outlook=rainy จากโหนดแม่ที่เป็นหมายเลขหนึ่งจะมีเส้นเชื่อมไปยังโหนดลูกที่เป็นโหนดหมายเลขหก ถ้าพิจารณาที่ข้อมูลของโหนดหมายเลขห้า

node(5, [3, 7, 12, 13]-[ ])

ปรากฏข้อมูลในโหนดนี้สี่เรคคอร์ดและทุกรคคอร์ดอยู่ในกลุ่ม positive (นั่นคือ class = yes) ทำให้การสร้างต้นไม้ย่อยในกิ่งนี้สิ้นสุดโดยมีโหนดหมายเลขห้าเป็นโหนดใบ ในขณะที่โหนดหมายเลขสองและหมายเลขหกยังต้องมีการสร้างต้นไม้ย่อยต่อไป

จะเห็นได้ว่าโครงสร้างต้นไม้ตัดสินใจนี้สามารถใช้เป็นโมเดลอธิบายการตัดสินใจของนักกอล์ฟได้ว่าในสถานการณ์แบบใดที่เขาตัดสินใจออกไปเล่นกอล์ฟ และในสถานการณ์แบบใดที่เขาจะไม่ออกไปเล่น แต่ต้นไม้ตัดสินใจนี้จะแปลความได้ค่อนข้างยากสำหรับผู้ใช้ที่ไม่คุ้นเคยกับโครงสร้างข้อมูลชนิดนี้ การแปลความที่ง่ายกว่านี้คือการใช้กฎการตัดสินใจ ถ้า...แล้ว

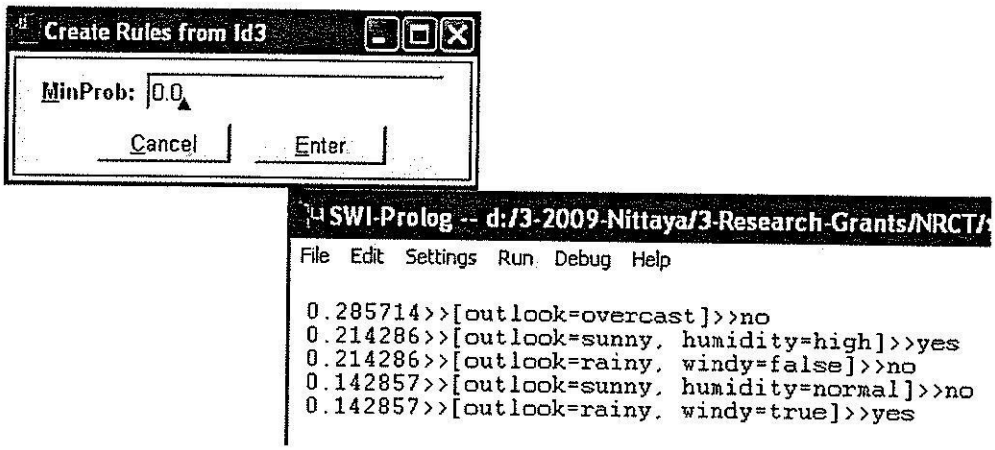
โปรแกรม Probabilistic-rule generation

การแปลงโครงสร้างต้นไม้ตัดสินใจที่ประกอบด้วย node และ edge ให้เป็นกฎการตัดสินใจ IF..THEN มีขั้นตอนการเขียนคำสั่งดังต่อไปนี้

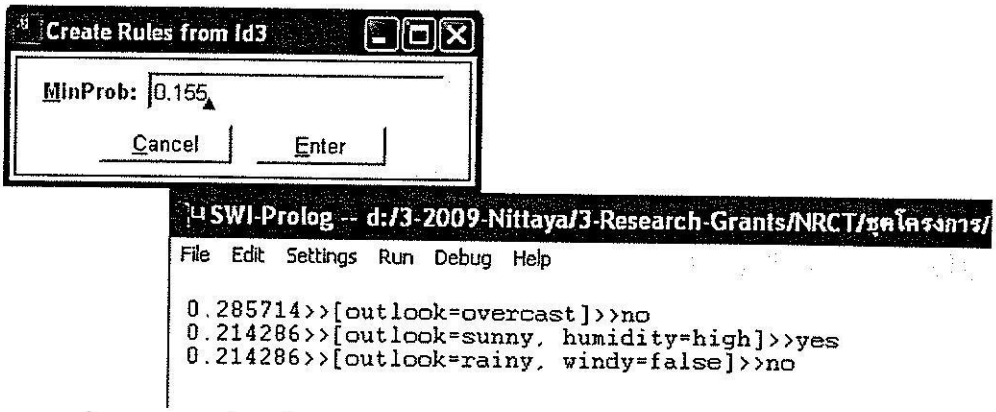
```
addKnowledge :-
        findall([A], pathFromRootToLeaf(A, _), Res),
        retractall(_>>_>>_),
        maplist(apply(assert), Res), writeln(addToKNB), nl.
selectRule(V,Res) :-
        findall(N>>X>>Class, (X>>Class>>N,N>=V), Res1),
        sort(Res1,Res2),reverse(Res2,Res).
path(A, [H| T], C) :- edge(A, H, B), path(B, T, C).
path(C, [], C) :- !.
```

```
pathFromRootToLeaf(V>>Class>>Num,C) :-
        path(1, V, C),
        node(C, Value1-Value2),
        (Value1=[]; Value2 =[]),
        (Value1=[]->length(Value2, Numb) ; length(Value1, Numb)),
        total+Total,
        Num is Numb/Total,
        (Value1=[]->Class=yes ; Class=no).
```

จากหน้าจอเริ่มต้นในรูปที่ 2.3 ถ้าผู้ใช้ไม่เปลี่ยนค่าความน่าจะเป็น (นั่นคือใช้ค่าที่โปรแกรมระบุไว้เป็น 0.0) เมื่อคลิกที่ปุ่ม Enter จะปรากฏผลลัพธ์เป็นกฎการตัดสินใจแสดงได้ดังรูปที่ 2.5 ส่วนในกรณีที่ผู้ใช้ระบุค่าความน่าจะเป็นเป็นค่าอื่นเช่น 0.155 เมื่อคลิกที่ปุ่ม Enter จะปรากฏผลลัพธ์เป็นกฎการตัดสินใจแสดงได้ดังรูปที่ 2.6



รูปที่ 2.5 โมเดลที่ได้ในลักษณะของกฎการตัดสินใจเมื่อเรียกใช้โปรแกรมด้วยค่าความน่าจะเป็น 0.0



รูปที่ 2.6 โมเดลที่ได้เมื่อเรียกใช้โปรแกรมด้วยค่าความน่าจะเป็น 0.155

การแสดงผลลัพธ์ในลักษณะของกฎการตัดสินใจ ข้อความแต่ละบรรทัดจะประกอบด้วยส่วนประกอบสามส่วนคือ ค่าความน่าจะเป็น, ปัจจัยที่ใช้ประกอบการตัดสินใจ และผลการตัดสินใจ ดังตัวอย่างกฎการตัดสินใจในรูปที่ 2.6

0.285714 >> [outlook=overcast] >> no

0.214286 >> [outlook=sunny, humidity=high] >> yes

0.214286 >> [outlook=rainy, windy=false] >> no

กฎแรกแปลความหมายได้ว่ากฎนี้มีความน่าจะเป็นในการครอบคลุมข้อมูล 0.285714 โดยมีปัจจัยประกอบการตัดสินใจคือ outlook=overcast และผลการตัดสินใจคือ no นั่นคือเมื่อท้องฟ้ามีเมฆมากนักกอล์ฟตัดสินใจที่จะไม่ออกไปเล่นกอล์ฟ

กฎในบรรทัดที่สองแปลความด้วยวิธีเดียวกันได้ว่า กฎนี้มีความน่าจะเป็นในการครอบคลุมข้อมูล 0.214286 โดยมีปัจจัยประกอบการตัดสินใจคือ outlook=sunny และ humidity=high ผลการตัดสินใจคือ yes (เครื่องหมาย , ในส่วนที่เป็นปัจจัยประกอบการตัดสินใจหมายถึง and ใช้ในกรณีที่มีปัจจัยมากกว่าหนึ่งปัจจัย)

กฎที่สามแปลความได้ว่า กฎนี้มีความน่าจะเป็นในการครอบคลุมข้อมูล 0.214286 โดยมีปัจจัยประกอบการตัดสินใจคือ outlook=rainy และ windy=false ผลการตัดสินใจคือ no

บทที่ 3
การทดสอบโปรแกรม

วิธีการทดสอบความถูกต้องและประสิทธิภาพของโปรแกรม

ระบบเหมืองข้อมูลเพื่อการจำแนก เป็นโปรแกรมที่พัฒนามาจากพื้นฐานของโปรแกรม ID3 (Quinlan 1993) ที่ใช้การสร้างต้นไม้ตัดสินใจเป็นเครื่องมือหลักในการแยกข้อมูลที่มีหลายคลาสปะปนกันให้เหลือเป็นกลุ่มข้อมูลย่อยที่เป็นคลาสเดียวกัน จากนั้นอ่านลักษณะของข้อมูลในแต่ละกลุ่มย่อยจากโครงสร้างโหนดและกิ่งของต้นไม้ตัดสินใจ วิธีการจำแนกข้อมูลเพื่อให้ได้โมเดลในลักษณะนี้ได้รับการยอมรับว่ามีความถูกต้องของโมเดลสูง และข้อเด่นของวิธีการนี้คือโมเดลเข้าใจได้ง่าย (เมื่อเปรียบเทียบกับวิธีการอื่น เช่น neural network) ดังนั้นงานวิจัยนี้จึงทดสอบความถูกต้องของโปรแกรมด้วยการทดสอบผลลัพธ์ที่ได้ (ได้แก่ โมเดลข้อมูล) กับผลลัพธ์ที่ได้จากโปรแกรม ID3 แต่เนื่องจากโมเดลที่เรียนรู้จากข้อมูลฝึกขนาดใหญ่จะมีความซับซ้อนมาก จึงไม่สะดวกที่จะทดสอบเปรียบเทียบตัวโครงสร้างของตัวโมเดลโดยตรง แต่จะใช้วิธีทดสอบความแม่นตรงของโมเดลเมื่อใช้ทำนายคลาสของข้อมูลชุดทดสอบ จากนั้นเปรียบเทียบความแม่นตรงในการทำนายคลาสโดยโมเดลที่สร้างจากโปรแกรมที่พัฒนาขึ้นเทียบกับโมเดลจากโปรแกรม ID3

ในส่วนของการทดสอบประสิทธิภาพเป็นการทดสอบการลดขนาดของโมเดล โดยใช้โมเดลทั้งหมดที่มีค่าความน่าจะเป็นตั้งแต่ 0.0 ถึง 1.0 ทดสอบความแม่นตรงในการทำนาย (predicting accuracy) เปรียบเทียบกับความแม่นตรงที่ได้เมื่อโมเดลมีขนาดเล็กลง การลดขนาดจะลดเป็นสัดส่วนจากค่าความน่าจะเป็นสูงสุด ใช้สัดส่วนตั้งแต่ 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), จนถึงขนาด 1/10(max.Prob) ตัวอย่างเช่นจากข้อมูลนักกอล์ฟ สมมติให้โมเดลในรูปแบบของกฎการตัดสินใจที่มีค่าความน่าจะเป็นกำกับ (probabilistic decision rules) ทั้งหมดที่มีค่าความน่าจะเป็นภายในช่วง [0.0-1.0] ได้แก่

Rule 1: 0.285714>>[outlook=overcast]>>no

Rule 2: 0.214286>>[outlook=sunny, humidity=high]>>yes

Rule 3: 0.214286>>[outlook=rainy, windy=false]>>no

Rule 4: 0.142857>>[outlook=sunny, humidity=normal]>>no

Rule 5: 0.142857>>[outlook=rainy, windy=true]>>yes

Rule 6: 0.011326>>[outlook=rainy, humidity=high]>>no

ค่าความน่าจะเป็นสูงสุดของกฎการตัดสินใจชุดนี้คือ 0.285714 และเมื่อต้องการลดขนาดโมเดลลงด้วยเกณฑ์ 1/10(max.Prob) จะได้ 1/10 * (0.285714) = 0.02857 ดังนั้นที่ขนาดของการลดสัดส่วนลงหนึ่งในสิบของค่าความน่าจะเป็นจะได้โมเดลเป็น Rules 1-5 โดย Rule 6 จะถูกคัดทิ้งเนื่องจากค่าความน่าจะเป็นไม่ถึงเกณฑ์

ข้อมูลที่ใช้ในการทดสอบ

การทดสอบความถูกต้องของการสร้างโมเดล ความแม่นตรงของโมเดลในการทำนายคลาสของชุดข้อมูลทดสอบ รวมถึงการทดสอบประสิทธิภาพของโมเดลที่มีการลดขนาดลง จะใช้ชุดข้อมูลมาตรฐานจาก UCI Repository (www.ics.uci.edu/~mlearn/MLRepository.html) จำนวน 6 ชุดข้อมูล ประกอบด้วย ข้อมูล Monk, Post-operative, Breast cancer, Vote, Hepatitis, Mushroom ข้อมูลแต่ละชุดประกอบด้วย training data และ test data ดังนั้นวิธีการทดสอบความแม่นตรงจึงใช้วิธี hold out หรือวิธีที่แยกข้อมูลทดสอบออกจากข้อมูลฝึก เพื่อให้การทดสอบโมเดลเป็นการทดสอบกับข้อมูลใหม่โดยแท้จริง รายละเอียดของข้อมูลที่ใช้ทั้งหกชุดข้อมูลสรุปได้ดังตารางที่ 3.1

ตารางที่ 3.1 รายละเอียดของข้อมูลที่ใช้ทดสอบความถูกต้องและประสิทธิภาพของโปรแกรม

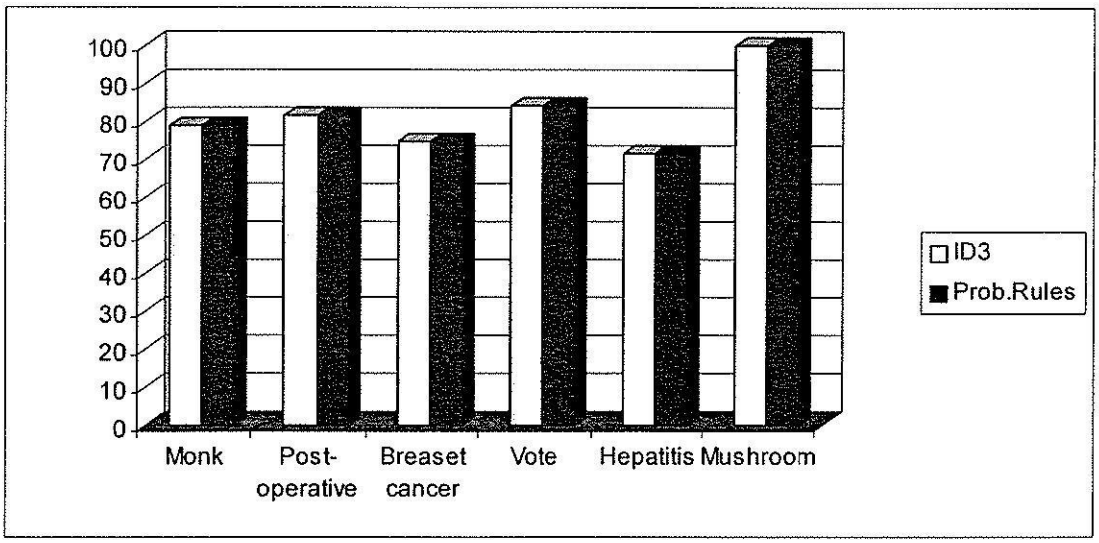
ชื่อชุดข้อมูล	จำนวนข้อมูลฝึก (instances)	จำนวนข้อมูลทดสอบ (instances)	จำนวน predicting attributes	จำนวนคลาสใน goal attribute
Monk	124	432	6	2
Post-operative	50	36	8	2
Breast cancer	191	95	9	2
Vote	300	135	16	2
Hepatitis	103	52	19	2
Mushroom	5416	2708	22	2

ผลการทดสอบ

การทดลองในขั้นตอนแรกเป็นการทดสอบความถูกต้องของโปรแกรมสร้างกฎการตัดสินใจ โดยเปรียบเทียบกับโปรแกรมID3 วิธีการเปรียบเทียบใช้ค่าความแม่นตรง (accuracy) ในการทำนายคลาสข้อมูลทดสอบของโมเดลที่ได้จากทั้งสองโปรแกรม ผลการทดสอบแสดงดังตารางที่ 3.2 และผลการทดสอบเดียวกันนี้แสดงเป็นภาพได้ดังรูปที่ 3.1

ตารางที่ 3.2 ผลการทดสอบความแม่นตรงของโมเดล Probabilistic decision rules เปรียบเทียบกับโมเดล ID3

ชื่อชุดข้อมูล	ความแม่นตรง (accuracy)	
	Probabilistic decision rules	ID3
Monk	79.03%	79.03%
Post-operative	81.65%	81.65%
Breast cancer	74.73%	74.73%
Vote	84.33%	84.33%
Hepatitis	71.42%	71.42%
Mushroom	100%	100%

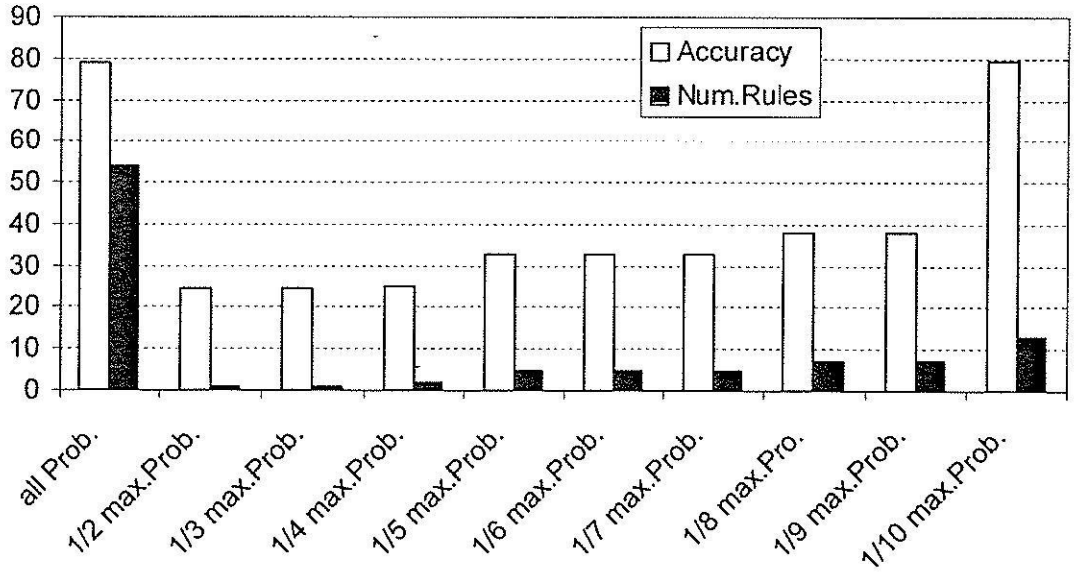


รูปที่ 3.1 กราฟเปรียบเทียบความแม่นตรง (accuracy) ของ Probabilistic decision rules และ ID3

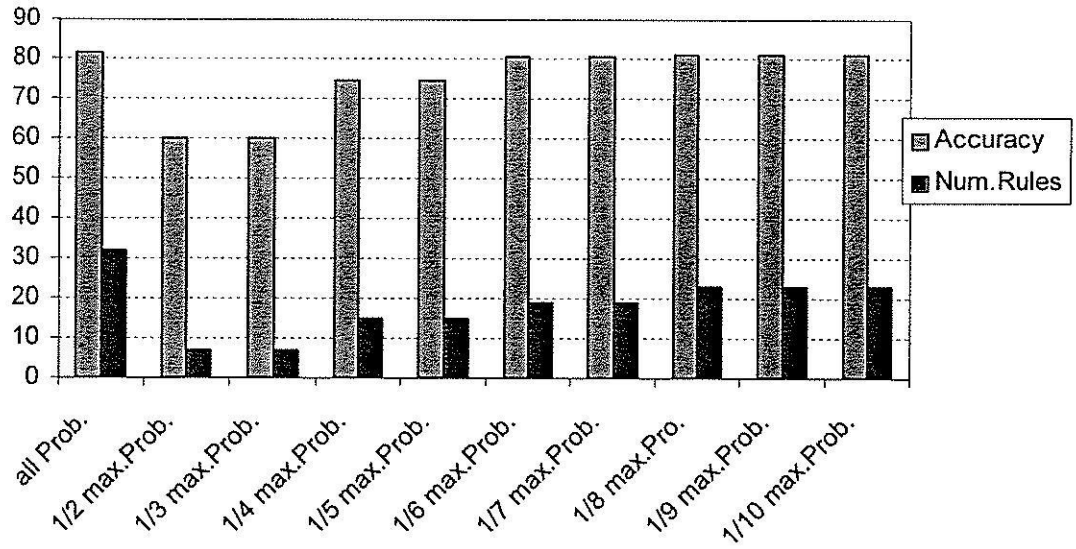
การทดสอบในขั้นตอนที่สองเป็นการทดสอบประสิทธิภาพของโมเดลที่แสดงในลักษณะของกฎการตัดสินใจที่ค่าความน่าจะเป็นกำกับ หรือ Probabilistic decision rules ที่มีการลดขนาดของโมเดลลงมาตามสัดส่วนต่างๆกัน วิธีการวัดประสิทธิภาพจะพิจารณาจากขนาดของโมเดล (วัดจากจำนวนของกฎ) และความแม่นตรงของโมเดลในการทำนายคลาสของข้อมูล ผลการทดสอบการสร้างและทดสอบโมเดลกับชุดข้อมูลมาตรฐานทั้งหกชุดสรุปผลได้ดังตารางที่ 3.3 และแสดงผลการทดสอบกับชุดข้อมูลมาตรฐานแต่ละข้อมูลเป็นภาพกราฟได้ดังรูปที่ 3.2-3.7

ตารางที่ 3.3 ความแม่นตรงของโมเดล Probabilistic decision rules เมื่อมีการลดขนาดของโมเดล

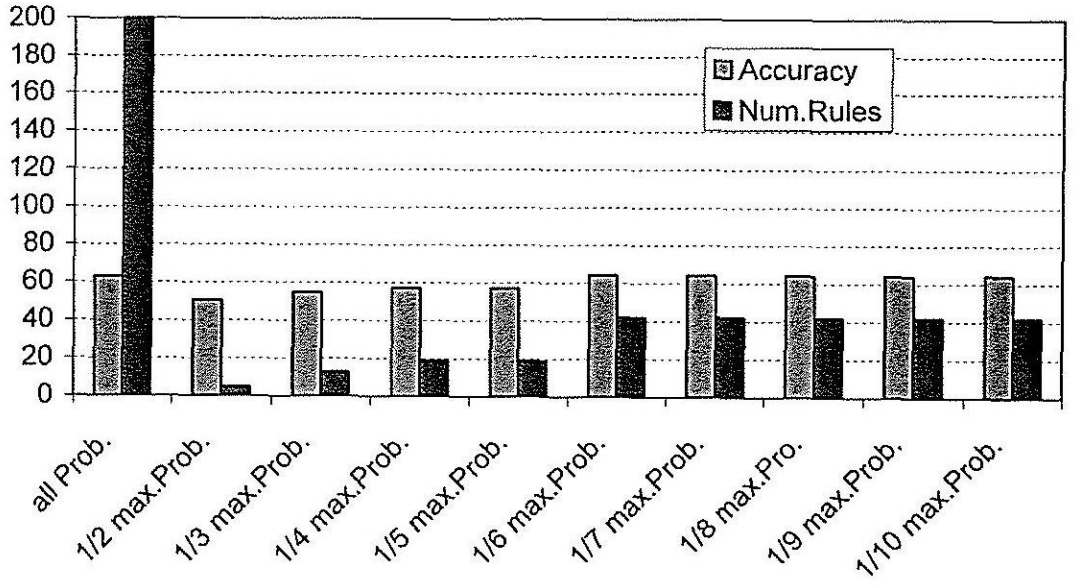
Monk		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max Prob	1/6 max Prob	1/7 max Prob	1/8 max Prob	1/9 max Prob	1/10 max Prob
	Accuracy	79.03	24.76	24.76	25	32.63	32.63	32.63	38.19	38.19	79.6
	# rules	54	1	1	2	5	5	5	7	7	1
Post-operative		All Prob.	1/2 max Prob	1/3 max Prob	1/4 max Prob	1/5 max 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	80.9
	# rules	32	7	7	15	15	**19**	19	23	23	2
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	64.2
	# rules	200	5	13	19	19	**42**	42	42	42	4
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.
	# rules	47	1	**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.
	# rules	16	3	**4**	4	4	4	4	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	10
	# rules	25	3	3	5	5	**8**	8	8	9	



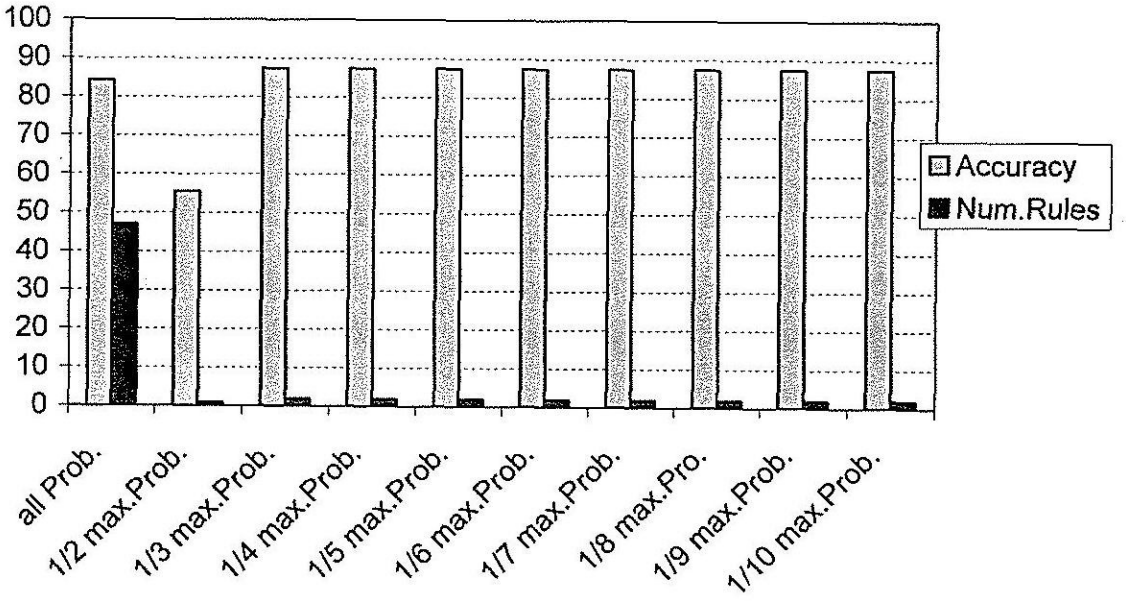
รูปที่ 3.2 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Monk



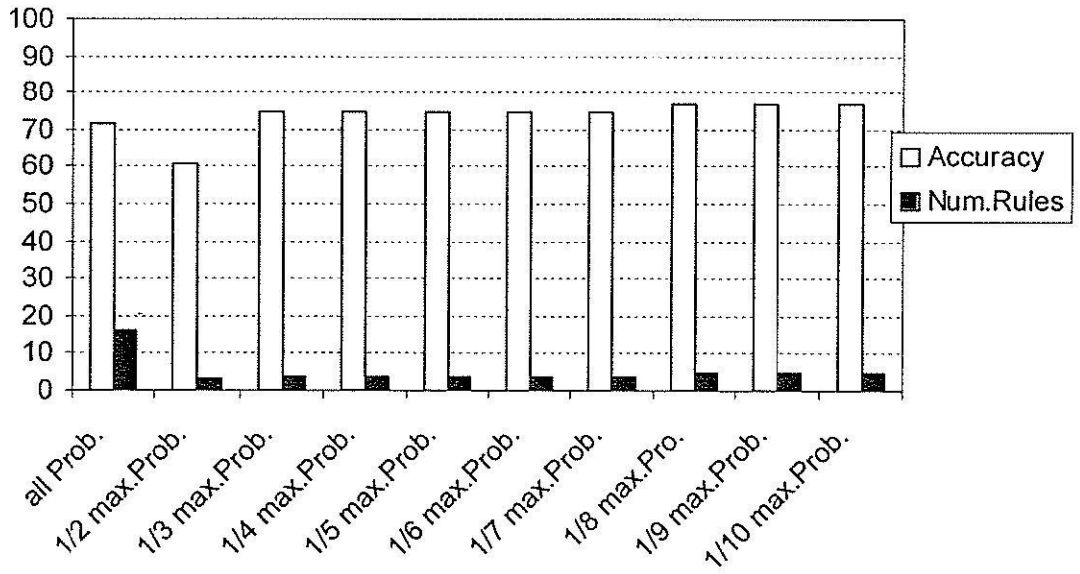
รูปที่ 3.3 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Post-operative



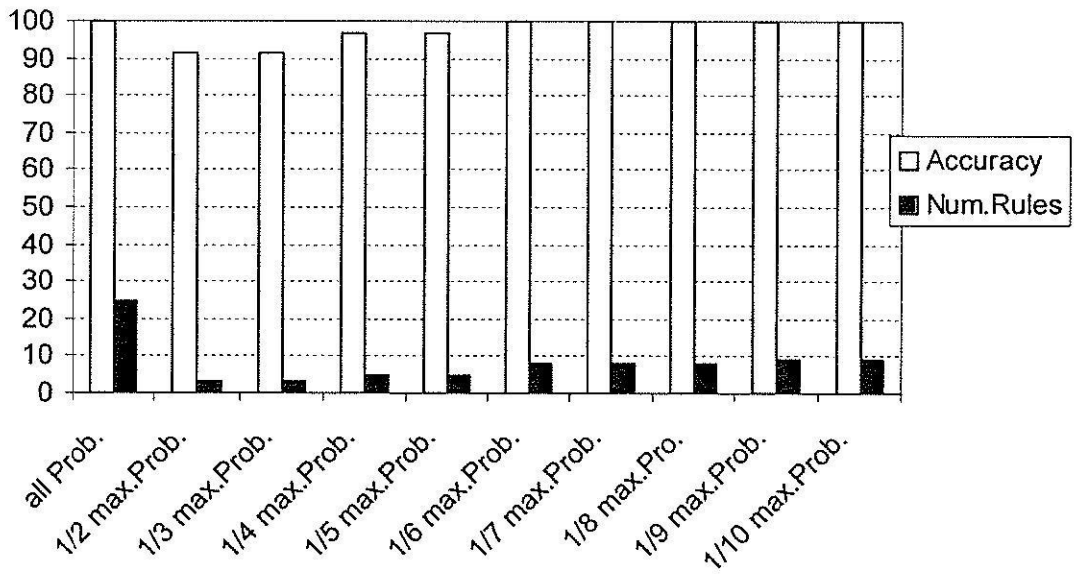
รูปที่ 3.4 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Breast cancer



รูปที่ 3.5 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Vote



รูปที่ 3.6 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Hepatitis



รูปที่ 3.7 ความแม่นตรงของโมเดล Probabilistic decision rules ที่ขนาดต่างๆ กันของชุดข้อมูล Mushroom

อภิปรายผล

การทดสอบความถูกต้องในขั้นตอนการสร้างโมเดลของโปรแกรมการทำเหมืองข้อมูลเพื่อการจำแนก ปรากฏผลว่าโมเดลที่ได้ในลักษณะของ probabilistic decision rules ให้ค่าความแม่นตรงเท่ากับโมเดลที่ได้จากโปรแกรม ID3 ดังแสดงในตารางที่ 3.2 และรูปที่ 3.1 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 3.3 ยืนยันข้อสมมุติฐานนี้ นั่นคือ ในจำนวนข้อมูลที่ใช้ทดสอบหกชุด มีข้อมูลถึงสี่ชุด (ได้แก่ข้อมูล Monk, Breast cancer, Vote, Hepatitis) ที่ให้ผลการทำนายที่มีความแม่นตรงสูงขึ้นเมื่อคัดเลือกโมเดลให้มีเฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง

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

จากผลการทดสอบดังกล่าวทำให้สามารถสรุปได้ว่าการลดขนาดของโมเดล โดยคัดเลือกไว้เฉพาะกฎการตัดสินใจที่มีค่าความน่าจะเป็นสูง จะช่วยให้ได้โมเดลที่สามารถนำไปใช้

ทำนายข้อมูลได้ดีขึ้น โมเดลที่ได้จะไม่มีลักษณะ overfitting ลักษณะการคัดเลือกกฎเช่นนี้เทียบเคียงได้กับการทำ post-pruning ของเทคนิค decision-tree induction แต่วิธีการทำจะง่ายกว่าและแนวคิดไม่ซับซ้อน

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

บทที่ 4
บทสรุป

สรุปผลการวิจัย

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) เนื่องจากโปรแกรมจะใช้ข้อมูลนี้ฝึกกระบวนการสร้างโมเดล เพื่อใช้โมเดลนี้ทำนายค่าของแอททริบิวต์เป้าหมายได้อย่างแม่นยำ การทำเหมืองข้อมูลแบบจำแนกจึงเป็นกระบวนการค้นหาแอททริบิวต์ที่สามารถนำมาใช้ช่วยทำนายค่าเป้าหมาย (หรือจำแนกค่าของแอททริบิวต์เป้าหมาย) ได้อย่างแม่นยำที่สุด เทคนิคในการค้นหา แอททริบิวต์ดังกล่าวมีได้หลายเทคนิค เช่น ใช้การสร้าง decision tree เพื่อให้โครงสร้างข้อมูลที่ได้เป็นเครื่องมือในการจำแนกค่าแอททริบิวต์เป้าหมาย หรือใช้การคำนวณหาข้อมูลที่อยู่

ใกล้ที่สุดแล้วตัดสินค่าแอททริบิวต์เป้าหมายตามข้อมูลนั้น 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 จากผลการทดลองยังพบว่าโมเดลที่มีขนาดเล็กสามารถทำนายแอททริบิวต์เป้าหมายในชุด

ข้อมูลทดสอบได้ดีกว่าโมเดลที่ยังไม่ได้ลดขนาดลง สาเหตุหลักของการเพิ่มค่าความแม่นตรงนี้เนื่องจากโมเดลที่มีขนาดเล็กลงจะช่วยลดผลกระทบในเรื่อง overfitting ของโมเดล

ข้อเสนอแนะ

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

Algorithm 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

---

ในประเด็นของการปรับปรุงประสิทธิภาพของงานวิจัยนี้ให้ใช้งานได้เต็มประสิทธิภาพกับข้อมูลจริงที่อาจจะมีขนาดใหญ่มาก แต่โมเดลที่ใช้ประโยชน์ได้จริงปรากฏอยู่ในข้อมูลส่วนเล็กๆ เท่านั้น เช่น ข้อมูลรหัสพันธุกรรม อาจใช้แนวทางของเทคนิคการค้นหาโมเดลจากข้อมูลกลุ่มย่อยหลายๆกลุ่ม แล้วคัดเลือกกลุ่มย่อยที่ให้ความเป็นไปได้สูงสุดในการผลิตโมเดลที่เป็นประโยชน์ จากนั้นจึงประยุกต์ใช้อัลกอริทึมที่เสนอในงานวิจัยนี้สร้างโมเดลในรูปแบบของต้นไม้ตัดสินใจ

บรรณานุกรม

C. Blake, E. Keogh, and C.J. Merz. UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA, 1998.

Marko Bohanec and Ivan Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15:223-250, 1994.

Leo Breiman, Jerome Freidman, Richard Olshen, and Charles Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.

William W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 988-994, Chambery, France, 28 August-3 September 1993.

P.R. Cohen and D. Jensen. Overfitting explained. *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, 1997.

Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A further study of pruning methods in decision tree induction. *AI&Statistics-95: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 211-218, Ft. Lauderdale, Florida, 4-7 January 1995.

Jerome H. Freidman. A recursive partitioning decision rule for nonparametric classifiers. *IEEE Transactions on Computers,* C-26: 404-408, April 1977.

S.B. Gelford and C.S. Ravishankar. A tree-structured piecewise-linear adaptive filter. *IEEE Transactions on Information Theory*, 39(6): 1907-1922, November 1993.

Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*, second edition. Morgan Kaufmann, 2006.

Hyunsoo Kim and G.J. Koehler. An investigation on the conditions of pruning an induced decision tree. *European Journal of Operational Research,* 77(1):82, August 1994.

John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2): 227-243, 1989.

John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.

John Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221-234, 1987.

John Ross Quinlan. *C4.5: Programs for machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

John Ross Quinlan and Ronald Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80(3): 227-248, March 1989.

Cullen Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10: 153-178, 1993.

Jan L. Talmon and P. McNair. The effect of noise and biases on the performance of machine learning algorithms. *International Journal of Bio-Medical Computing*, 31(1): 45-57, July 1992.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2006.

C.S. Wallace and J.D. Pattrick. Coding decision trees. *Machine Learning*, 11(1): 7-22, April 1993.

David H. Wolpert. On overfitting avoidance as bias. *Technical Report SFI TR 92-03-5001*, The Santa Fe Institute, 1992.

ภาคผนวก

ภาคผนวก ก

รหัสต้นฉบับของโปรแกรมเพื่อการทำเหมืองข้อมูลแบบจำแนก

```
/* ============  Classification : main program ========= */
%
% To run the program, call this procedure:
%                                              id3menu.
%
%--------------  MENU ----------------

id3menu :-
      new(Dialog,dialog('Create Rules from Id3')),
      send_list(Dialog, append,
            [ new(D1, text_item(datafile,'post-operative.pl')),
              new(Per,text_item(minProb,'0.0')),
              button(cancel, message(Dialog, destroy)),
              button(enter, and(message(@prolog,callId3,
                        D1?selection,
                        Per?selection ),
              message(Dialog, destroy)))  % enter&destroy
              ]),
      send(Dialog, open).
%
callId3(Dfile,Per):-
      term_to_atom(Per1,Per),
      consult(Dfile),
      mainId3(Per1).
%
%--------------- ID3 -----------------

:- dynamic current_node/1,node/2,edge/3,hasClass/2.

mainId3(Min):-
      init(AllAttr,EdgeList),
      getnode(N),
      create_edge_onelevel(N,AllAttr,EdgeList),
      addKnowledge,
      selectRule(Min,Res),
      maplist(writeln,Res).

init(AllAttr,[root-nil/PB-NB]):-
      writeln(limitUpto400nodes),
      retractall(hasClass(_,_)),
      attribute( class,[ Y1, Y2]),
      assert(hasClass(Y1,Y2)),
      retractall(node(_,_)),
      retractall(current_node(_)),
      retractall(edge(_,_,_)),
      assert(current_node(0)) ,
      hasClass(C1,C2),
      findall(X,attribute(X,_),AllAttr1),
      delete(AllAttr1,class,AllAttr),
      findall(X2,instance(X2,class=C1,_),PB),
      findall(X3,instance(X3,class=C2,_),NB),
      length(PB,N1),
      length(NB,N2),
      N is N1+N2,
      retractall(total+_),
      apply(assert,[total+N]).
```

```
getnode(X):-
      current_node(X),
      X1 is X+1,
      retractall(current_node(_)),
      assert(current_node(X1)),
      X1 <400. % limit at 400 nodes

create_edge_onelevel(_,_,[]):-!.
create_edge_onelevel(_,[],_):-!.

create_edge_onelevel(N,AllAttr,EdgeList):-
      create_nodes(N,AllAttr,EdgeList).

create_nodes(N,AllAttr,[H1-H2/PB-NB|T]):-
            getnode(N1),
            assert(edge(N,H1=H2,N1)),
            assert(node(N1,PB-NB)),
            append(PB,NB,AllInst),
            ( (PB\==[], NB\==[])->
                  (cand_node(AllAttr,AllInst,AllSplit),
                   min_cand(AllSplite,[V,MinAttr,Split]),
                   delete(AllAttr,MinAttr,Attr2),
                   create_edge_onelevel( N1,Attr2,Split))
                  ;   true ),
            create_nodes(N,AllAttr,T).

create_nodes(_,_,[]):-!.
create_nodes(_,[],_):-!.

%
% ------------------------------
check1 :- node(X,Y),write(node/X+Y),fail ; true.
check2 :- edge(X,Y,Z),node(Z,N),writeln(edge/X+Y+Z*N),fail ; true.
check3 :- edge(X,Y,Z),node(X,L),node(Z,N),
            writeln(edge/X*L+Y+Z*N),fail ; true.
check4 :- X>>Class>>N,writeln(X>>Class>>N),fail ; true.

addKnowledge :-
      findall([A],pathFromRootToLeaf(A,_),Res),
      retractall(_>>_>>_),
      maplist(apply(assert),Res),
      write(addToKNB).

selectRule(V,Res):-
      findall(N>>X>>Class,(X>>Class>>N,N>=V),Res1),
      sort(Res1,Res2),
      reverse(Res2,Res).

path(A,[H|T],C):-
      edge(A,H,B),
      path(B,T,C).

path(C,[],C):-!.
```

```
pathFromRootToLeaf(V>>Class>>Num,C):-
      path(1,V,C),
      node(C,Value1-Value2),
      (Value1=[];Value2=[]),
      (Value1=[]->length(Value2,Numb);length(Value1,Numb)),
      total+Total,
      Num is Numb/Total,hasClass(C1,C2),
      (Value1=[]->Class=C1;Class=C2).

min_cand([H|T], Min) :-
      min_cand(T, H, Min).

min_cand([], Min, Min).
min_cand([H|T], Min0, Min) :-
      H=[V,_,_],
      Min0=[V0,_,_],
      ( V<V0 ->Min1=H;Min1=Min0),
      % Min1 is min(H, Min0),
      min_cand(T, Min1, Min).

cand_node([H|T],CurInstL,[[Val,H,SpliteL]|OtherAttr]):-
      info(H,CurInstL,Val,SpliteL),
      cand_node(T,CurInstL,OtherAttr).

cand_node([],_,[]):-!.
cand_node(_,[],[]).

concat3(A,B,C,R):-
      atom_concat(A,B,R1), atom_concat(R1,C,R).

info(A,CurInstL,R,Splite):-
      attribute(A,L),
      maplist(concat3(A,=),L,L1),%make a good form
      suminfo(L1,CurInstL,R,Splite). %L1=[size-small,size-large]

%suminfo(+[color=red,color=blue],+[1,2,3,4],-info,-SplitList).
%
suminfo([H|T],CurInstL,R,[Split|ST]) :-
      AllBag=CurInstL,
      hasClass(C1,C2),
      term_to_atom(H1,H),
      findall(X1,(instance(X1,_,L1),member(X1,CurInstL),
                  member(H1,L1)),BagGro),
      findall(X2,(instance(X2,class=C1,L2),member(X2,CurInstL),
                  member(H1,L2)),BagPos),
      findall(X3,(instance(X3,class=C2,L3),member(X3,CurInstL),
                  member(H1,L3)),BagNeg),
      (H11=H22) =H1,
      length(AllBag,Nall),length(BagGro,NGro),length(BagPos,NPos),
      length(BagNeg,NNeg),
      Split=H11-H22/BagPos-BagNeg,
      suminfo(T,CurInstL,R1,ST),
      (  NPos is 0 *->L1 = 0; L1 is (log(NPos/NGro)/log(2)) ),
      (  0 is NNeg *->L2 = 0; L2 is (log(NNeg/NGro)/log(2)) ),
      (  NGro is 0 -> R= 999;
           R is (NGro/Nall)*(-(NPos/NGro)*L1-(NNeg/NGro)*L2)+R1 ).
suminfo([],_,0,[]).
```

```
%% Data:  'Post_operative.pl'
%
%   class yes = patient sent to ICU
%   class no = patient not in critical condition: sent to general floor
%                                        or prepared to go home

attribute( internalTemp,       [mid, high, low] ).
attribute( surfaceTemp,        [mid,high, low] ).
attribute( oxygenSaturation,   [excellent, good, fair, poor] ).
attribute( bloodPressure,      [high, mid, low] ).
attribute( tempStability,      [stable, mod_stable, unstable] ).
attribute( coreTempStability,  [stable,mod_stable, unstable] ).
attribute( bpStability,        [stable, mod_stable, unstable] ).
attribute( comfort,            [5, 7, 10, 15] ).
attribute( class,   [ home, ward]).

%data
instance(1, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=15] ).
instance(2, class=home, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable, bpStability=stable,
            comfort=10] ).
instance(3, class=ward, [internalTemp=high, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10] ).
instance(4, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=unstable, bpStability=mod_stable, comfort=15] ).
instance(5, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable, bpStability=stable,
            comfort=10] ).
instance(6, class=home, [internalTemp=high, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=15] ).
instance(7, class=home, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable,
            bpStability=mod_stable, comfort=5] ).
instance(8, class=home, [internalTemp=high, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=unstable,
            bpStability=stable, comfort=10]).
instance(9, class=home, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10] ).
instance(10, class=home, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10] ).
instance(11, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(12, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10] ).
instance(13, class=ward, [internalTemp=high, surfaceTemp=high,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=unstable, coreTempStability=stable,
            bpStability=unstable, comfort=15] ).
```

```
instance(14, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10] ).
instance(15, class=home, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=unstable, bpStability=stable, comfort=15]).
instance(16, class=ward, [internalTemp=high, surfaceTemp=high,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=unstable, coreTempStability=stable,
            bpStability=unstable, comfort=10] ).
instance(17, class=ward, [internalTemp=low, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=mod_stable, comfort=15]).
instance(18, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10] ).
instance(19, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=15] ).
instance(20, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10] ).
instance(21, class=ward, [internalTemp=low, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=15] ).
instance(22, class=home, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=unstable, coreTempStability=stable,
            bpStability=unstable, comfort=10] ).
instance(23, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=15] ).
instance(24, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10] ).
instance(25, class=ward, [internalTemp=high, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10] ).
instance(26, class=ward, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10] ).
instance(27, class=ward, [internalTemp=high, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=low, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10] ).
instance(28, class=ward, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10] ).
instance(29, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=15]).
instance(30, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(31, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(32, class=ward, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(33, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
```

```
            tempStability=unstable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10]).
instance(34, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(35, class=ward, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10]).
instance(36, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(37, class=home, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(38, class=home, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(39, class=home, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=7]).
instance(40, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
instance(41, class=ward, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(42, class=home, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(43, class=home, [internalTemp=high, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(44, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(45, class=ward, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(46, class=home, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(47, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(48, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=low, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(49, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=mod_stable, bpStability=mod_stable, comfort=10]).
instance(50, class=ward, [internalTemp=low, surfaceTemp=high,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(51, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=unstable, coreTempStability=stable,
            bpStability=unstable, comfort=10]).
instance(52, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
```

```
instance(53, class=ward, [internalTemp=high, surfaceTemp=high,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10]).
instance(54, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(55, class=ward, [internalTemp=high, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=15]).
instance(56, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=unstable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
instance(57, class=ward, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(58, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
instance(59, class=ward, [internalTemp=mid, surfaceTemp=high,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(60, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable, bpStability=stable,
            comfort=10]).
instance(61, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(62, class=home, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(63, class=ward, [internalTemp=high, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=unstable,
            bpStability=unstable, comfort=10]).
instance(64, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(65, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(66, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(67, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable, bpStability=stable,
            comfort=10]).
instance(68, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=low, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(69, class=ward, [internalTemp=low, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(70, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
instance(71, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=high,
```

```
            tempStability=stable, coreTempStability=stable, bpStability=stable,
            comfort=10]).
instance(72, class=ward, [internalTemp=mid, surfaceTemp=low,
            oxygenSaturation=excellent, bloodPressure=high,
            tempStability=stable, coreTempStability=stable,
            bpStability=mod_stable, comfort=10]).
instance(73, class=ward, [internalTemp=low, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(74, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=mod_stable, comfort=10]).
instance(75, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(76, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=unstable,
            bpStability=stable, comfort=10]).
instance(77, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=high, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(78, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(79, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=stable, comfort=10]).
instance(80, class=ward, [internalTemp=high, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=unstable, comfort=5]).
instance(81, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid, tempStability=stable,
            coreTempStability=stable, bpStability=unstable, comfort=10]).
instance(82, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(83, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=15]).
instance(84, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=15]).
instance(85, class=ward, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=excellent, bloodPressure=mid,
            tempStability=unstable, coreTempStability=stable,
            bpStability=stable, comfort=10]).
instance(86, class=home, [internalTemp=mid, surfaceTemp=mid,
            oxygenSaturation=good, bloodPressure=mid, tempStability=unstable,
            coreTempStability=stable, bpStability=stable, comfort=15]).


% ========= End of Data File ===========
```

ภาคผนวก ข

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

- N. Kerdprasop and K. Kerdprasop (2010). Probabilistic knowledge discovery from medical databases. *Proceedings of the 14th International Annual Symposium on Computational Science and Engineering (ANSCSE 14)*, Mae Fah Luang University, Chiang Rai, Thailand, March 23-26.

- N. Kerdprasop and K. Kerdprasop (2006). Density estimation technique for data stream classification. *Proceedings of the 17th International Workshop on Database and Expert Systems Applications (DEXA)*, Krakow, Poland, September 4-8, pp.662-666

- N. Wongprachanukul, N. Kerdprasop, and K. Kerdprasop (2005). A comparative study of method for pruning decision trees. *Proceedings of 31st Congress on Science and Technology of Thailand*, Suranaree University of Technology, Nakhon Ratchasima, Thailand, October 18-20.

- K. Kerdprasop, N. Kerdprasop and N. Wongprachanukul (2005). การศึกษาเปรียบเทียบวิธีการลดความซับซ้อนของโมเดลข้อมูล. *Proceedings of the Research Network Development of Higher Education Alliance in Nakhon Ratchasima*, Suranaree University of Technology, Nakhon Ratchasima, Thailand, June 24, pp.68-70.

Proceedings of the 14th International Annual Symposium on Computational Science and Engineering, Mae Fah Luang University, Chiang Rai, 23-26 March 2010.

PROBABILISTIC KNOWLEDGE DISCOVERY FROM MEDICAL DATABASES

Nittaya Kerdprasop and Kittisak Kerdprasop
Data Engineering and Knowledge Discovery (DEKD) Research Unit,
School of Computer Engineering, Suranaree University of Technology,
Nakhon Ratchasima, Thailand

ABSTRACT

Medical knowledge discovery is an emerging area within the data-mining field that attracts many new researchers from diverse disciplines. During the past decade numerous learning techniques had been employed to discover useful knowledge from health examination and clinical data. Unlike past efforts that simply concentrated on the deployment of well-known learning techniques on medical data sets, our new approach expands the learning algorithm to deal with uncertain knowledge. We devise an algorithm to generate probabilistic knowledge from the induced decision tree. The implementation of the proposed algorithm is demonstrated via second-order predicates and a meta-programming approach. Experimental results on several medical domains emphasize the simple form of knowledge representation and the potential of incorporating learning results as background knowledge in the knowledge-base system.

KEYWORDS

Medical knowledge mining, Probabilistic knowledge induction.

1. INTRODUCTION

The automated learning of models from patient data and biomedical records has become more and more essential since the extensive computerization in healthcare industry and the significant advancement in genomic and proteomic technologies during the last decade. Medical and clinical databases have been created and constantly growing at an exponential rate. The development of an automatic and intelligent data analysis tool is an obvious solution to the data-flooding problem in medical domains [4], [12], [13].

Knowledge extraction from huge amount of health databases is expected to ease the medical decision-making process. The ultimate goal of knowledge extraction is to generate the most accurate and useful knowledge and represent it in an understandable format. Such goal is, however, difficult to accomplish due to the learning complexity of knowledge induction methods and the nonconformity of the database contents. Most of the time knowledge discovery from medical databases results in reporting large number of irrelevant knowledge [6], [9]. We thus focus our study on this issue and devise a technique to extract a limited number of knowledge that is most likely relevant to the specific domain.

In medical knowledge mining, interpretability of results is an important feature of the data analysis tool. Medical practitioners need a system that can produce accurate results in an

understandable form. Therefore, knowledge represented as rules has been widely used for knowledge discovery in medical applications. Classification rule induction is an approach commonly used for building diagnosis models [2], [5], [7], [14]. Association rule mining is an induction method applied for exploring patterns that are frequently occur in medical data [3], [10]. Classification and association mining methods are two major techniques for rule generation that work successfully in many domains. Nevertheless, in medical applications these learning techniques tend to generate a lot of rules. Too many rules, some are redundant and uninteresting, cause problems to the medical practitioners because a truly relevant one can be easily overlooked.

We thus propose a rule induction method based on the decision-tree structure that adopts the probability concept to select the most probable applicable rules. The outline of this paper is as follows. After the introductory section, we present in Section 2 the general framework and the algorithms of our proposed method. The implementation and experimental results are illustrated in Sections 3. Section 4 concludes this paper with the discussion on further research direction.

2. FRAMEWORK AND METHOD FOR PROBABILISTIC KNOWLEDGE INDUCTION

Our knowledge induction system (Figure 1) is based on the decision-tree induction method [11]. Decision tree induction is a popular method for mining knowledge from data and representing the result as a classifier tree. Popularity is due to the fact that mining result in a form of decision tree is interpretability, which is more concern among casual users than a sophisticated method but lack of understandability. A decision tree is a hierarchical structure with each node contains decision attribute and node branches corresponding to different attribute values of the decision node. The goal of building decision tree is to partition data with mixing classes down the tree until each leaf node contains data with pure class.

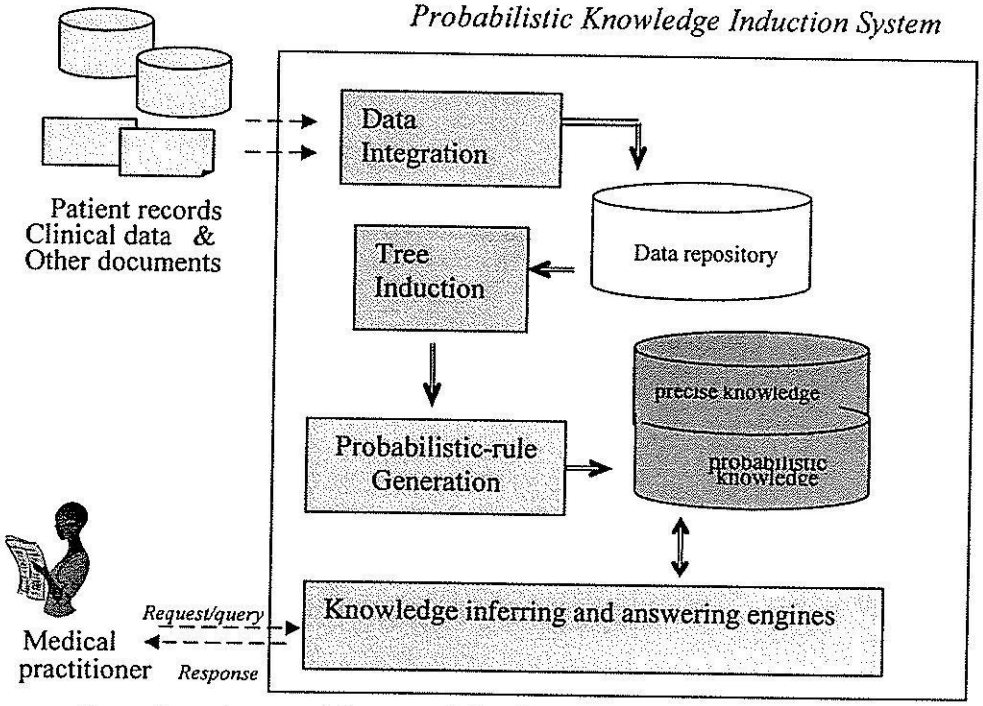


Figure 1. A general framework for the tree-based probabilistic knowledge induction.

In our system framework, we increase interpretability of the knowledge mining results by transforming the decision tree structure into a small set of decision rules. After a complete decision tree has been created, we calculate the probability of case occurrence augmented with each leaf node. In the phase of decision rule generation, these probability values will be sorted. Rules within the top ranking part will be displayed to assist medical practitioner for making decision. In the designed framework, probabilistic knowledge induction system is composed of four main components: data integration, tree induction, probabilistic-rule generation, and the knowledge inferring and answering engines. Data integration component is responsible for collecting data from different sources, cleaning and format transforming. These data are to be used by the tree induction component.

Given the induced tree, the probabilistic-rule generation component traverses each tree branch to calculate the likelihood of path occurrence. This likelihood is interpreted as the probability of event and associated to the rule generated from the path traversal. The generated probabilistic rules are then sorted. Rules at the top ranking (specified by the given minimum probability) are stored in the knowledge base as the probabilistic knowledge and could be used for recommendation or answering query to the medical practitioner. Algorithms for knowledge induction based on tree structure (Algorithm 1), probabilistic-rule generation from decision tree (Algorithm 2), and probabilistic knowledge inferring to answer the most probable class decision on new case (Algorithm 3) are given in the following.

Algorithm 1 Knowledge induction

Input: a data set formatted as Prolog clauses
Output: a decision tree with node and edge structures

(1) Initialization

  (1.1) Clear temporary knowledge base (KB) by removing all information regarding the predicates node, edge and current_node

  (1.2) Set node counter = 0

  (1.3) Scan data set to get information about data attributes, positive instances, negative instances, total data instances

(2) Building tree

  (2.1) Increment node counter

  (2.2) Repeat steps 2.2.1-2.2.4 until there is no more attributes left for creating decision attributes

    (2.2.1) Compute Info value of each candidate attribute

    (2.2.2) Choose the attribute that yields minimum Info to be decision node

    (2.2.3) Assert edge and node information into KB

    (2.2.4) Split data instances along node branches

  (2.3) Repeat steps 2.1 and 2.2 until the lists of positive and negative instances are empty

  (2.4) Output tree structure containing node and edge predicates

Algorithm 2 Probabilistic knowledge generation

Input: a decision tree with node and edge structures, and a probability threshold
Output: a set of probabilistic rules ranking from the highest probability

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

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

(1.2) Compute probability as a proportion

(number of instances at leaf node) / (total data instances in a data set)

(1.3) Assert a rule containing a triplet (attribute-value pair, class, probability value) into KB

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

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

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

Algorithm 3 Probabilistic knowledge inferring

Input: a KB containing probabilistic knowledge, and a new case with unknown class value
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 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 voting scheme

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we present the implementation technique of our proposed tree-based probabilistic-knowledge induction framework using logic programming paradigm. Prolog code is based on the syntax of SWI Prolog (www.swi-prolog.org).

Data format. In logic programming, program and data take the same format, i.e. all are in Prolog clausal form. For the purpose of demonstration, we use the health examination data of 86 patients after the operation. The general conditions such as blood pressure and temperature are observed to determine whether the patient is in good condition and should be sent home shortly (class=home), or the condition is quite moderate and should stay at the hospital ward for further follow up (class=ward). Even though binary classification is a typical task in medical domains, the code presented in this section can be easily modified to classify data with more than two classes. The post-operative data set (downloadable from the UCI repository [1]) in Prolog clausal form is shown some part as the following.

```
attribute( internalTemp,       [mid, high, low]).
attribute( surfaceTemp,        [mid, high, low]).
attribute( oxygenSaturation,   [excellent, good, fair, poor]).
attribute( bloodPressure,      [high, mid, low]).
attribute( tempStability,      [stable, mod-stable, unstable]).
attribute( coreTempStability,  [stable, mod-stable, unstable]).
attribute( bpStability,        [stable, mod-stable, unstable]).
attribute( comfort,            [5,7,10,15]).
attribute( class,              [home, ward]).
instance(1,class=ward,[internalTemp=mid, surfaceTemp=low, oxygenSaturation=excellent,
                       bloodPressure=mid, tempStability=stable, coreTempStability=stable,
                       bpStability=stable, comfort=15]).
...
```

Main module. The three algorithms, explained in the previous section, are called by the main module, which is the top-level of our program implementation. The Prolog coding of main module is as follows:

```
main:-init(AllAttr,EdgeList),
      getnode(N),
      create_edge_onelevel(N,AllAttr,EdgeList),
      addKnowledge,
      write(chooseMinProb),
      read(Min),
      selectRule(Min,Res),
      maplist(writeln,Res).
```

The built-in predicate `maplist` is a second-order predicate [8] provided in the library of SWI Prolog. Its implementation is declared recursively as the following. The predicates `init` and `getnode` in main module invoke the following initialization process. The predicates `assert` and `retractall` are also second-order predicates responsible for asserting and removing, respectively, information in the knowledge base. Another second-order predicate `apply` repeatedly assert clauses into the knowledge base. The built-in second-order predicate `findall` searches for all solutions that satisfy the constrained predicates.

Probabilistic-rule generation. In main module, the predicates `addKnowledge` and `selectRule(Min,Res)` are invoked to compute probability along each tree branch to generate probabilistic rules and then select only rules that could occur at the probability level higher than the specified threshold. Prolog coding of this module is as follows:

```
addKnowledge :- findall([A],pathFromRootToLeaf(A,_),Res),
                retractall(_>>_>>_),
                maplist(apply(assert),Res).

selectRule(V,Res):- findall(N>>X>>Class,(X>>Class>>N,N>=V),Res1),
                    sort(Res1,Res2),
                    reverse(Res2,Res).
```

```
pathFromRootToLeaf(V>>Class>>Num,C):- path(1,V,C),
                node(C,Value1-Value2),
                (Value1=[]; Value2=[]),
                (Value1=[]->length(Value2,Numb); length(Value1,Numb)),
                total+Total,
                Num is Numb/Total,
               (Value1=[]-> Class=home; Class=ward).
```

Running results on probabilistic-rule induction. For the demonstration purpose, we show the final result of probabilistic rule induction with a specified minimum threshold 0.02. Each result has been formatted as probability >> rule's conditions (shown as attribute-value pairs) >> decision on class value (either home or ward).

```
0.0930233>>[comfort=10, bloodPressure=high, surfaceTemp=low]>>home

0.0581395>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=stable, surfaceTemp=mid,
            tempStability=unstable]>>home

0.0465116>>[comfort=10, bloodPressure=high, surfaceTemp=mid,
            bpStability=mod_stable]>>home

0.0348837>>[comfort=15, bpStability=unstable, surfaceTemp=mid]>>home

0.0348837>>[comfort=15, bpStability=stable, internalTemp=mid,
            tempStability=stable]>>home

0.0348837>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=unstable, surfaceTemp=mid,
            tempStability=stable]>>home

0.0348837>>[comfort=10, bloodPressure=low]>>home

0.0348837>>[comfort=10, bloodPressure=high, surfaceTemp=mid,
            bpStability=stable, oxygenSaturation=excellent]>>home

0.0232558>>[comfort=15, bpStability=unstable, surfaceTemp=high]>>home

0.0232558>>[comfort=15, bpStability=mod_stable]>>home

0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=unstable,
            tempStability=unstable, bpStability=stable]>>ward

0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=stable, surfaceTemp=low]>>home

0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=mod_stable,
            surfaceTemp=high]>>home

0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=low, surfaceTemp=low, tempStability=stable]>>home

0.0232558>>[comfort=10,    bloodPressure=mid,    coreTempStability=stable,
            internalTemp=high]>>home
```

4. CONCLUSION

Modern healthcare organizations generate huge amount of electronic data stored in heterogeneous databases. Data collected by hospitals and clinics are not yet turned into useful knowledge due to the lack of efficient analysis tools. We thus propose a rapid prototyping of an

automatic knowledge-mining tool to induce probabilistic knowledge from medical data. The induced knowledge is to be integrated into the knowledge base of a medical decision support system. Thus, in our design the knowledge base will be composed of precise knowledge as well as a minimal set of induced probabilistic knowledge. Discovered knowledge can also facilitate the reuse of knowledge base among decision-support applications within organizations that own heterogeneous clinical and health databases. Direct application of medical probabilistic knowledge base is for medical related decision-making. Other indirect but obvious application of such knowledge is to pre-process other data sets by grouping it into focused subset containing only relevant data instances.

The main contribution of this work is our implementation of knowledge mining engines based on the concept of higher-order Horn clauses using Prolog language. Higher-order programming has been originally appeared in functional languages in which functions can be passed as arguments to other functions and can also be returned from other functions. This style of programming has soon been ubiquitous in several modern programming languages such as Perl, PHP, and JavaScript. Higher order style of programming has shown the outstanding benefits of code reuse and high level of abstraction. This paper illustrates higher order programming techniques in SWI-Prolog. The powerful feature of meta-level programming in Prolog facilitates the reuse of mining results represented as rules to be flexibly applied as conditional clauses in other applications.

The plausible extensions of our current work are to add constraints into the knowledge mining method in order to limit the search space and therefore yield the most relevant and timely knowledge, and due to the uniform representation of Prolog's statements as a clausal form, mining from the previously mined knowledge should be implemented naturally. The probabilistic knowledge induction and inferring techniques presented in this paper can be applied to the development of probabilistic databases. We also plan to extend our system to work with stream data that normally occur in modern medical organizations.

REFERENCES

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Irvine, CA (2007) http://archive.ics.uci.edu/ml/
2. Bojarczuk, C.C. et al.: A Constrained-Syntax Genetic Programming System for Discovering Classification Rules: Application to Medical Data Sets. Artificial Intelligence in Medicine 30, 27--48 (2004)
3. He, Y., Tang, Y., Zhang, Y., Sunderraman, R.: Adaptive Fuzzy Association Rule Mining for Effective Decision Support in Biomedical Applications. Int. J. Data Mining and Bioinformatics 1, 1, 3--18 (2006)
4. Kononenko, I.: Machine Learning for Medical Diagnosis: History, State of the Art and Perspective. Artificial Intelligence in Medicine 1, 89--109 (2001)
5. Kretschmann, E. et al.: Automatic Rule Generation for Protein Annotation with the C4.5 Data Mining Algorithm Applied on SWISS-PROT. Bioinformatics 17, 10, 920--926 (2001)
6. Li, J., Fu, A., Fahey, P.: Efficient Discovery of Risk Patterns in Medical Data. Artificial Intelligence in Medicine (2008) doi:10.1016/j.artmed.2008.07.008
7. Mugambi, E. et al.: Polynomial-Fuzzy Decision Tree Structures for Classifying Medical Data. Knowledge-Based System 17, 2-4, 81--87 (2004)
8. Nadathur, G., Miller, D.: Higher-Order Horn Clauses. J. ACM 37, 777--814 (1990)
9. Ordonez, C.: Comparing Association Rules and Decision Trees for Disease Prediction. In: Proc. Int. Workshop on Healthcare Information and Knowledge Management, pp.17--24 (2006)
10. Ohsaki, M., Sato, Y., Yokoi, H., Yamaguchi, T.: A Rule Discovery Support System for Sequential Medical Data in the Case Study of a Chronic Hepatitis Dataset. In: Proc. ECML/PKDD-2003 Discovery Challenge Workshop, http://www.lisp.vse.cz/challenge/ecml pkdd2003
11. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1, 81--106 (1986)

12. Roddick, J.F. et al.: Exploratory Medical Knowledge Discovery: Experiences and Issues. ACM SIGKDD Explorations Newsletter 5, 1, 94--99 (2003)
13. Shillabeer, A., Roddick, J.F.: Establishing a Lineage for Medical Knowledge Discovery. In: Proc. 6th Australasian Conf. on Data Mining and Analytics, pp.29--37 (2007)
14. Zhou, Z., Jiang, Y.: Medical Diagnosis with C4.5 Rule Preceded by Artificial Neural Network Ensemble. IEEE Transactions on Information Technology in Biomedicine 1, 37--42 (2003)

ACKNOWLEDGMENTS

This research has been funded by grants from the National Research Council of Thailand (NRCT) and the second author is supported by the Thailand Research Fund (TRF, grant number RMU5080026). DEKD has been fully supported by Suranaree University of Technology.

Proceedings of 17th International Workshop on Database and Expert Systems Applications (DEXA), Krakow, Poland, 4-8 September 2006.

DENSITY ESTIMATION TECHNIQUE FOR DATA STREAM CLASSIFICATION

Nittaya Kerdprasop and Kittisak Kerdprasop
Data Engineering and Knowledge Discovery Research Unit
School of Computer Engineering, Suranaree University of Technology, Thailand
{nittaya, kerdpras}@sut.ac.th

ABSTRACT

Density estimation is an important pre-processing step in the problem of data stream classification in which the number of data is overwhelming and the exact data distribution is unknown. We simplify the problem by employing a statistical sampling technique to obtain an approximate solution. With the proposed method, an unbounded large data set can be sampled in a number of random configurations, and that data can be used to describe the data set as a whole. The efficiency of the method depends largely on the ability to draw samples effectively which in turn depends on how close we can estimate the target density. We use finite mixture models to represent the probability density functions of the data stream. Then, we apply the EM algorithm twice to learn the model parameters. The efficiency of our estimation technique has been shown in the experimental results.

1. Introduction

The recent advances in hardware and software have enabled the rapid generation of continuous stream of information such as customer click streams, telephone records, web page visits, and so on. An accurate and timely analysis over an unbounded stream of data poses a new challenge to researchers and practitioners in the area of data mining. Data stream is defined as massive amounts of data continuously generated at a rapid rate, possibly time-varying and unpredictable [1, 6].

Major characteristics of data streams are the continuously online arrival of data elements, uncontrolled order of such elements upon arrival, variable sizes, and a one-time processing of an element before it is discarded or archived due to the massive size of data that far exceeds the storage capacity. Therefore, the process of data stream analysis is required to examine each data element at most once and produce the analysis results as fast as possible. The requirements of timely analysis and efficient memory usage constrain most data mining algorithms to sacrifice accuracy of the analysis results for the fast and feasible processing.

A direct solution to the problem of data stream mining is the development of approximate algorithms [3, 9]. However, the large volumes of data continuously arriving in a stream could eventually make the algorithms inefficient. A more practical solution is to apply a data reduction technique along with the approximation algorithms. Data stream analysis by sampling a set of representatives out of the incoming stream is a natural solution for producing approximate results. Nevertheless, in the context of continuous data stream in which the data size is unknown, simply applying random sampling method cannot give reliable approximation.

We thus propose an estimation technique to approximate the form of the data stream distribution. Correct density estimation can lead us to an optimal performance on the subsequent analysis phase such as data classification or discriminant analysis. Our approach employs the Expectation-Maximization (EM) algorithm [4] to learn the parameters of the density function. We run EM for only a few iterations to obtain a simplified estimator, and then run EM to full convergence for a more accurate one. The two estimators are the basis of our density estimation approach.
The paper is organized as follows. Section 2 presents the theoretical background of parameter estimation technique commonly used in finite mixture models. Section 3 explains the proposed method to estimate density together with the technique to obtain a sample set from the data stream.

Section 4 shows the experimental results of density estimator assessment and the classification performance. Section 5 concludes our work with a discussion for future research.

2. Parameter estimation via EM

In situation where one has to build a statistical model from massive amounts of data, a common practice is to use a random sample of the examples or observations in the data. But in the domain of data stream in which data may be generated from different sources and the exact data distribution is unknown, finite mixture models are appropriate and more powerful in representing arbitrary complex probability density functions.

We consider multi-dimensional data stream as finite mixture models of Gaussian or normal probability density functions. A finite mixture model [5, 7, 8] is a combination of distributions written as

$$p(x \mid \theta) = \sum_{i=1}^{K} \alpha_i p(x \mid \theta_i), \quad with \sum_{i=1}^{K} \alpha_i = 1, \qquad (1)$$

where K is the number of mixture components, $\alpha_i \geq 0$ are the mixing proportions and add up to one. The $p(x \mid \theta_i)$ denotes the multivariate normal distribution with mean vector μ and covariance matrix Σ, that is $\theta_i = (\mu_i, \Sigma_i)$, and $\theta = \{\theta_1, \ldots, \theta_K, \alpha_1, \ldots, \alpha_K\}$ is the complete set of parameters needed to specify the mixture.

Given a set of n independent and identically distributed observations $X = \{x_1, \ldots, x_n\}$, the log-likelihood corresponding to a K-component mixture is

$$\begin{aligned} \log p(X \mid \theta) &= \log \prod_{j=1}^{n} p(x_j \mid \theta) \\ &= \sum_{j=1}^{n} \log \sum_{i=1}^{K} \alpha_i p(x_j \mid \theta_i). \end{aligned} \qquad (2)$$

The maximum likelihood (ML) estimate of the parameter value is

$$\hat{\theta} = \arg\max_{\theta} \{\log p(X \mid \theta)\}. \qquad (3)$$

The EM algorithm is the usual choice for obtaining ML estimates of the mixture parameters. The EM algorithm is an iterative procedure to find the maximum of likelihood function in incomplete data problems. For finite mixtures, the missing data $Z = \{z_1, \ldots, z_n\}$ is a set of n labels associated with the n observations indicating the mixture component from which the observation x is drawn. Let $Y = X \cup Z$ be a full data set. The probability distribution of Z depends on X and the unknown parameter θ. Let $p(Y \mid \theta)$ denote the joint density of the complete data. The EM algorithm starts with some initial parameter estimate $\theta^{(0)}$. The algorithm repeats the expectation (E) and maximization (M) steps until convergence. Let $\theta^{(t-1)}$ denote the current parameter value. Then, in the subsequent t^{th} iteration, the algorithm works as follows.

E-steps: Computes the conditional expectation of the complete data log-likelihood,

$$Q(\theta \mid \theta^{(t-1)}) = \mathrm{E}[\log p(Y \mid \theta) \mid \theta^{(t-1)}, X]$$
$$= \sum_{\text{all possible } Y} \log p(Y \mid \theta) p(Z \mid \theta^{(t-1)}, X) \qquad (4)$$

M-steps: Replace $\theta^{(t)}$ that maximizes the function Q, that is $\theta^{(t)}$ satisfies

$$Q(\theta^{(t)} \mid \theta^{(t-1)}) \geq Q(\theta \mid \theta^{(t-1)}), \ \textit{for all } \theta.$$

Given an initial parameter $\theta^{(0)}$, the EM algorithm produces a sequence $\{\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots\}$ that converges to a stationary point of the likelihood function.

3. Density estimation over data stream

To analyze data stream we assume that the observed data have a normal distribution. Given a specific number of models or components, the EM algorithm is applied twice to obtain the E and E' distributions (as shown in Figure 1).

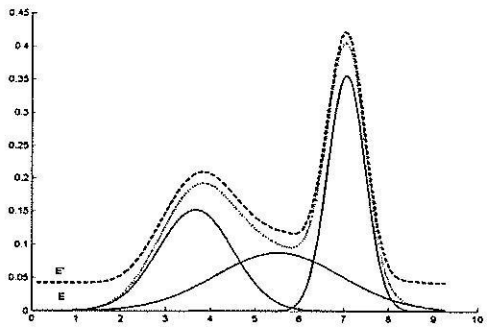


Figure 1. A proposed method to estimate density and obtain sample from the estimated distributions.

The idea of the proposed method is illustrated in Figure 1. The target function, $f(x)$, is represented as a one-dimensional 3-Gaussian mixture models (the three solid lines at the bottom) from which we want to draw samples. The proposal density $E(x)$ is estimated from a few iterations of the EM technique with the upper bound requirement that $E(x) > f(x)$ for all x. Full iterations of EM produces $E'(x)$, which is the approximation of the unknown target density.

The EM algorithm requires a pre-specified number of K components to be incorporated into the mixture models. According to our proposed method, a suitable number should be selected by a user. To cope with multi-dimensional problem, we propose to use a statistical method – principal component analysis (PCA) – to reduce the complicated problem to a simpler two-dimensional problem. That is, we take into account only the first and second major components of the data set. The two-dimensional data are used to train the EM algorithm to estimate parameters μ and Σ of the Gaussian mixture models. The estimated Gaussian distributions are E and E'.

From the estimated density E and the rough approximate E', we perform rejection sampling [11] with the decision criteria $\{E(x)/(\sqrt{d} * E'(x))\} \geq u$, when u is a uniform variable distributed between 0 and 1, and d is a dimensionality of the data. A broad distance of E and E' (such as at $x =$ 1 in Figure 1) represents a rejecting area, whereas a narrow distance (such as at $x = 6.5$) is an

acceptance one. The input from stream data has been taken one by one. The data item that satisfies the criteria will be included in the sample until the specified sample size is completely filled up. Then the sampled data set contains representatives of the whole stream. Any analysis methods can now be performed on this set. Our algorithm is shown as follows.

Algorithm Density Estimation and Sampling from Data Stream

Input: a d-dimensional data set D with N observations,

an integer K to specify the number of models,

a required sample size SS.

Output: a sample set S drawn from the mixture models.

Steps:

1. If $d > 0$ then Apply PCA to obtain 1st and 2nd principal components, to get a two-dimensional data set X
2. Set max_iteration = max{50, dK}
3. ($E(X)$, $E'(X)$) = *Density_Estimator* (X, K, max_iteration) *// Density estimation with EM //*
4. Set count = 0
5. While count < SS *// Rejection sampling steps //*
6. Sample x from $E(X)$
7. Generate u from U(0,1)
8. If $u \le E(x)/(\sqrt{d} * E'(x))$ then Accept x, add it to S, and increment count
9. Return S

Density_Estimator (X, K, *max_iteration*)

1. Initialize parameter $\theta = (\mu, \Sigma)$ for each of K Gaussian models by running K-means
2. Initialize the prior probabilities $P(m_k)$ of each model m to $1/K$, $k = 1,..., K$
3. Repeat
4. Compute the probability $P(m_k^{(i)} \mid x_n, \theta^{(i)}) = \dfrac{P(m_k^{(i)} \mid \theta^{(i)}) p(x_n \mid \mu_k^{(i)}, \Sigma_k^{(i)})}{\sum_j P(m_j^{(i)} \mid \theta^{(i)}) p(x_n \mid \mu_j^{(i)}, \Sigma_j^{(i)})}$
5. Update means μ_k, variances Σ_k, and priors P (■)

$$\mu_k^{(i+1)} = \frac{\sum_{n=1}^{N} x_n P(m_k^{(i)} \mid x_n, \theta^{(i)})}{\sum_{n=1}^{N} P(m_k^{(i)} \mid x_n, \theta^{(i)})}, \qquad \Sigma_k^{(i+1)} = \frac{\sum_{n=1}^{N} P(m_k^{(i)} \mid x_n, \theta^{(i)}) (x_n - \mu_k^{(i+1)})(x_n - \mu_k^{(i+1)})^T}{\sum_{n=1}^{N} P(m_k^{(i)} \mid x_n, \theta^{(i)})},$$

$$P(m_t^{(i+1)} \mid \theta^{(i+1)}) = \frac{1}{N} \sum_{n=1}^{N} P(m_k^{(i)} \mid x_n, \theta^{(i)})$$

6. Until the *max_iteration* has been reached or the joint likelihood of all data with respect to all the models is greater than the lower boundary criterion $CL(\theta)$

$$L(\theta) \ge CL(\theta) = \sum_{k=1}^{K} P(m_k \mid X, \theta) \log p(X \mid \theta) = \sum_{k=1}^{K} \sum_{n=1}^{N} P(m_k \mid x_n, \theta) \log p(x_n \mid \theta)$$

7. Return $\theta_i = (\mu_k, \Sigma_k)$ for $k = 1, ..., K$, and a rough $\theta_i' = (\mu^5{}_k, \Sigma^5{}_k)$ from 5 iterations

---

4. Experimental evaluation

To validate the utility of our proposed method, we firstly evaluate our density estimation and sampling method on synthetic Gaussian mixture data and then apply the method to the problem of classification on real data set obtained from the UCI Machine Learning Library [2].

The objective of our initial experiments is to empirically evaluate the closeness of the estimated density to the real one. The closeness is determined by comparing the Euclidean distance of the estimated mean vector $\hat{\mu}$ to the original mean vector μ, and comparing the estimated covariance matrix $\hat{\Sigma}$ to the original covariance matrix Σ. We use a synthetic data generator to produce two-dimensional Gaussian mixtures. The number of mixture models, number of points in each model, original mean vector and covariance matrix are input parameters.

Table 1. Experimental results of sampling from various mixing of Gaussian models, compared against the uniform random sampling which always assumes a single Gaussian model. The efficiency of the sampling methods is evaluated on the basis of the closeness of the estimated $\theta_i = (\mu_i, \Sigma_i)$ to the original means and covariance matrices of the generative models. The μ-differences and Σ-differences are averaged from K mixture models.

Number of Mixture Models	Sampling from Estimated Distributions		Uniform Random Sampling	
	μ-difference	Σ-difference	μ-difference	Σ-difference
2	0.000113	0.000901	0.088772	0.144793
6	0.000425	0.001527	0.090213	0.231109
8	0.000961	0.001599	0.098055	0.271645
12	0.000987	0.001938	0.200137	0.430098
14	0.001017	0.001991	0.299873	0.456131
16	0.001025	0.002007	0.300159	0.513772
18	0.001328	0.002031	0.330011	0.720001
20	0.001414	0.002508	0.460101	0.935644

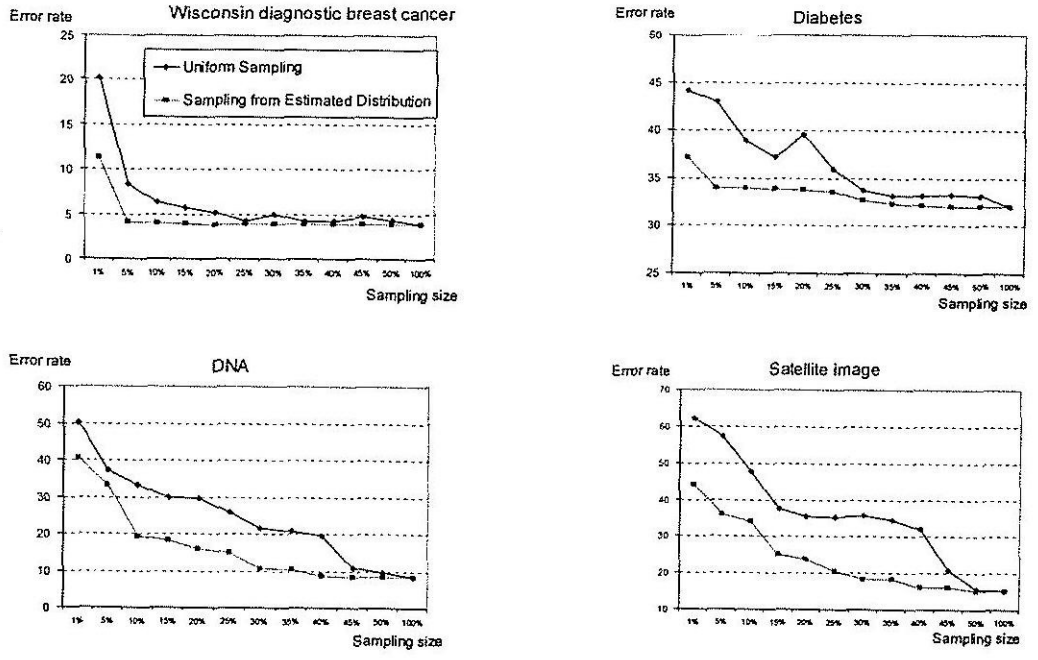


Figure 2. Classification results on four data sets

In our experiments, we vary the number of models from 2 to 20 with 50 to 1,000 data points in each model. To properly initialize the component means for the θ-parameter learning, we find the approximate mean points by running max{50, *dK*} iterations of k-means algorithm [10]. Component elements and main diagonal covariance matrix elements are also initialized accordingly, and off-diagonal matrix elements are constrained to zero. Some of our experimental results on the accuracy of our density estimator compared with the simple uniform sampling are illustrated in Table 1. The experimental results confirm the applicability of the EM approach toward the problem of θ-parameter approximation. The estimated means and variances are very close to the original parameter values.

To verify the utility of the proposed method on the real-world data sets, we run the C4.5 classification algorithm [10] on various sampled data from the UCI repository. We test our algorithm on four data sets: Wisconsin diagnostic breast cancer (466 observations, 2 classes), diabetes (512 observations, 2 classes), DNA (2000 observations, 3 classes), and satellite image (4435 observations, 6 classes). In each data set, we evaluate the error rate of the classification on separate test data.

We simulate a data stream by generating several sample sizes from each data set. In our experiments we observe the performance of classification on increasing sample sizes varied from 1%, 5%, 10%, 15%, ... , 50%, and the complete data set. The experimental results are shown in Figure 2. The results reveal the efficiency of the proposed method that less than 30% of the population is sufficient for the accurate classification over data stream.

5. Conclusions

In this paper we propose a technique of density estimation to analyze major characteristics of data stream. We also propose a sampling algorithm to efficiently draw representative samples from data containing mixture models. We apply the expectation-maximization technique to estimate the parameters of the mixture models. The algorithm *Density_Estimator* produces two density

functions, E and E'. The distance of E and E' at each sampling point is the decision criteria for either sample acceptance or rejection. A narrow distance along the two estimated densities tends to the acceptance case if the distance ratio is greater than the generated uniform random variable from the interval [0, 1]. The experimental results verify the utility of the proposed method. The classification experimentations on real-world data also confirm the efficiency of our method. We plan to further our study on skewed data in which the distributions are not uniformly distributed.

Acknowledgements

This research has been supported by grants from the Thailand Research Fund (TRF, MRG4780170), and the National Research Council. The Data Engineering and Knowledge Discovery Research Unit is fully supported by the research grants from Suranaree University of Technology.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems", *Proc. ACM PODS*, 2000.

[2] C. Blake and C. Merz, *UCI Repository of Machine Learning Databases*, 1998.

[3] G. Coremode and S. Muthukrishnan, "What's hot and what's not: Tracking most frequent items dynamically", *Proc. ACM PODS*, 2003.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society B*, 39, 1977, 1-22.

[5] M.A.T. Figueiredo and A.K. Jain, "Unsupervised learning of finite mixture models", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(3), 2002, 381-396.

[6] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data stream: A review", *SIGMOD Record*, 34(2), 2005, 18-26.

[7] J.M. Marin, K. Mengersen, and C. Robert, "Bayesian modelling and inference on mixtures of distributions", *Handbook of Statistics 25*, Elsevier-Science, 2005.

[8] G. McLachlan and D. Peel, *Finite Mixture Models*, John Wiley and Sons, 2000.

[9] S. Muthukrishnan, "Data streams: Algorithms and applications", *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2003.

[10] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2000.

[11] J. Von Neumann, "Various techniques used in connection with random digits", *Applied Mathematics Series*, 12, National Bureau of Standards, 1951.

Proceedings of 31st Congress on Science and Technology of Thailand, Suranaree University of Technology, Nakhon Ratchasima, Thailand, 18-20 October 2005.

การศึกษาเพื่อเปรียบเทียบประสิทธิภาพของเทคนิคการตัดกิ่งต้นไม้ตัดสินใจ

A COMPARATIVE STUDY OF METHODS FOR PRUNING DECISION TREES

นฤพนต์ ว่องประชานุกูล, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

Narupon Wongprachanukul, Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit, School of Computer Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand,
E-mail address: narupon@nsrc.or.th, nittaya@sut.ac.th, kerdpras@sut.ac.th

บทคัดย่อ: งานวิจัยนี้เป็นการศึกษาเพื่อเปรียบเทียบประสิทธิภาพของเทคนิคการตัดกิ่งต้นไม้ตัดสินใจที่มีชื่อเสียงสองวิธีคือ Reduced-error pruning และ Error-based pruning โดยมีจุดมุ่งหมายเพื่อวิเคราะห์ค่าความเที่ยงตรงในการจำแนกคลาสข้อมูล เวลาที่ใช้ในการสร้างโมเดล และขนาดของต้นไม้ตัดสินใจ เราทำการทดลองกับสิบชุดข้อมูลด้วยเทคนิคการตัดกิ่งเหล่านี้ เพื่อลดขนาดของต้นไม้และแก้ไขปัญหา "overfitting" ต้นไม้ที่ได้รับการตัดกิ่งแล้วจะใช้เวลาในการสร้างลดลงเนื่องจากขนาดที่เล็กลง และยังคงสามารถจำแนกข้อมูลใหม่ได้อย่างถูกต้อง

Abstract: We make a comparative study of two well-known pruning methods, reduced-error pruning and error-based pruning. The predictive accuracy, the time taken to build the model, and size of the pruned trees are evaluated for each pruning method. We conduct the experiments on ten data sets. Pruning methods aim at simplifying decision trees to avoid overfitting problem. The pruned trees result in faster classification and do not decrease their predictive accuracy.

Introduction: Decision tree is one of the tools used for data mining. The main application area is classification task. The model is built from a set of records, called training set. Each record consists of a number of attribute-value pairs. One of these attributes represents class of the record. We also have a test set for evaluating the performance of a decision tree.

When a decision tree is built, many of the branches may be overly expanded due to noise or outliers in the training set. The built model is too complex, since it tries to classify all records in the training set including noise and outliers. This problem is called "overfitting". We use tree pruning method to remove the least reliable branches, generally resulting in faster classification and improvement in the ability of the tree to correctly classify unknown data.

We study the performance of the post-pruning approach. A tree node is pruned by removing its branches from a fully grown tree (T_{max}) [1]. In the following subsections, we summarize the concepts of two pruning methods whose performances are evaluated in this paper.

Reduced-error pruning (REP): This method is probably the simplest pruning technique. It uses the pruning set to evaluate the goodness of a subtree of the complete tree. It starts with T_{max} and runs the test data through it. For each internal node, the number of

classification errors is counted if the subtree is kept compared with if the subtree is pruned. The decision on whether or not to prune the subtree is based on which alternative yields a minimum error.

A pruning operation involves replacing a subtree by a leaf. REP will perform this operation if it does not increase the total number of classification errors. Traversing the tree in a bottom-up strategy ensures that the result is the smallest pruned tree that has minimum error on the pruning data.

Error-based pruning (EBP): This method is implemented by the well-known decision tree inducer C4.5 [3]. Unlike REP, EBP uses the training set for building and simplifying trees. It visits nodes of T_{max} according to a bottom-up traversal strategy and uses the certainty factor (CF) parameter to control the pruning. CF is used to estimate the upper limit of the probability that an error occurs over the population at a leaf.

The subtree replacement is performed if the error estimate for the expected leaf is not greater than the sum of the error estimates for the current leaf nodes of the subtree. EBP also performs a pruning operation called "subtree raising" that replaces a subtree with its most populated branch if this does not increase the estimated error.

Methodology: We conducted experiments and used the decision tree on ten data sets from UCI Machine Learning Repository [2] with the above pruning methods and use the decision tree inducer C4.5. Model accuracy was tested with ten-fold cross-validation technique. The main characteristics of the data sets are presented in Table 1.

Table 1. The main characteristics of the data sets used for experiments

Data set	No. of Instances	No. of Attributes	No. of Nominal attributes	No. of Numeric attributes	Missing values	No. of Classes
Anneal	898	38	32	6	yes	5
Audiology	226	69	69	0	yes	24
Glass	214	9	0	9	no	7
Glass-2	163	9	0	9	no	2
Hepatitis	155	19	13	6	yes	2
Ionosphere	351	34	0	34	no	2
Iris	150	4	0	4	no	3
Labor	57	16	8	8	yes	2
Soybean	683	35	35	0	yes	19
Vote	435	16	16	0	yes	2

Results, Discussion and Conclusion: We compare the predictive accuracy, the time taken to build the model and size of the pruned trees with the unpruned trees. These results are reported in Table 2.

Table 2. Accuracy, time taken to build the model and size of the pruned trees compare with the unpruned trees

	Time (s)			Tree sizes			Accuracy (%)		
Data set	REP	EBP	unpruned	REP	EBP	unpruned	REP	EBP	unpruned
Anneal	0.55	1.32	2.14	113	78	155	92.87	90.98	93.10
Audiology	0.11	0.22	0.22	47	54	62	71.24	77.43	76.55
Glass	0.11	0.22	0.17	17	59	59	71.50	66.82	65.89
Glass-2	0	0.05	0.05	15	17	17	74.23	80.37	80.37
Hepatitis	0.05	0.11	0.06	13	21	31	81.94	78.71	78.06
Ionosphere	1.65	2.14	1.70	13	35	35	90.60	88.03	88.32
Iris	0	0.06	0	9	9	9	94.67	96.0	96.0
Labor	0	0.05	0	7	5	22	82.46	73.68	78.95
Soybean	0.27	0.55	0.38	120	93	175	87.55	91.51	91.36
Vote	0.06	0.06	0.06	9	11	37	95.63	96.32	96.32

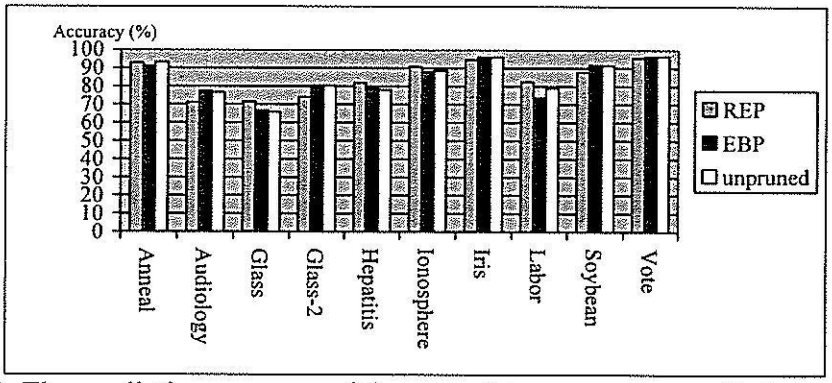


Figure 1. The predictive accuracy of the pruned trees compares with the unpruned trees

Both REP and EBP reduce the size of a fully grown tree by removing some unnecessary branches and do not significantly decrease the predictive accuracy of most final trees. REP produces the pruned tree in the shortest period of time, since it must not estimate classification errors. These experiments are still preliminary and need more systematic and extensive studies including additional comparative studies to other pruning methods.

References: [1]. Esposito, F., Malerba, D., and Semeraro, G. *A Comparative Analysis of Methods for Pruning Decision Trees.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, 5, 476-491. 1997.

[2]. Merz, C.J., and Murphy, P.M. *UCI Repository of machine learning databases.* http://www.ics.uci.edu/~mlearn/MLRepository.html. 1996.

[3]. Quinlan, J.R. *C4.5: Programs for Machine Learning.* 1993.

Keywords: decision trees, pruning methods, reduced-error pruning, error-based pruning.

ชื่องานวิจัย : **การศึกษาเปรียบเทียบวิธีการลดความซับซ้อนของโมเดลข้อมูล**

คณะผู้วิจัย : ผู้ช่วยศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ, ผู้ช่วยศาสตราจารย์ ดร.นิตยา เกิดประสพ และ นายนฤพนต์ ว่องประชานุกูล

ผู้นำเสนอผลงานวิจัย : นายนฤพนต์ ว่องประชานุกูล

สังกัด : หน่วยปฏิบัติการวิจัยวิศวกรรมข้อมูลและการค้นหาความรู้,

สาขาวิชาวิศวกรรมคอมพิวเตอร์, สำนักวิชาวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีสุรนารี

ที่อยู่สำหรับติดต่อ : 111 ถนนมหาวิทยาลัย, ต.สุรนารี, อ.เมือง, จ.นครราชสีมา 30000

โทรศัพท์: 044-224432 อีเมล์: nittaya@ccs.sut.ac.th

กลุ่มวิชา : กลุ่มงานวิจัยด้านวิศวกรรมศาสตร์

<u>วัตถุประสงค์ :</u>

งานวิจัยนี้มีจุดมุ่งหมายที่จะศึกษาเปรียบเทียบเทคนิคต่างๆ ที่ใช้ในการลดความซับซ้อนของโมเดลข้อมูล โดยมุ่งเน้นที่โมเดลประเภทต้นไม้ตัดสินใจที่นิยมใช้มากในงานทำเหมืองข้อมูลประเภทการจำแนกข้อมูลอัตโนมัติและการแสดงลักษณะร่วมของข้อมูล โครงสร้างต้นไม้ตัดสินใจที่มีขนาดใหญ่เกินไปจะซับซ้อนเข้าใจยากและนำไปสู่ปัญหาสำคัญคือ เป็นโมเดลที่จำเพาะมากเกินไป (overfitting) การหาวิธีลดความซับซ้อนโดยคงความเที่ยงตรงของโมเดล จะเป็นประโยชน์ต่อการพัฒนาอัลกอริทึมสังเคราะห์โมเดลเพื่อการจำแนกที่มีประสิทธิภาพและเที่ยงตรงสูง

<u>วิธีการ :</u>

เทคนิคหลักของการลดความซับซ้อนที่จะใช้ในการศึกษาวิจัยนี้ จะประกอบด้วยเทคนิคที่ใช้วิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติเพื่อตัดบางส่วนของโครงสร้างต้นไม้ที่ไม่ก่อประโยชน์ทิ้งไป เทคนิคที่จะนำมาใช้ศึกษา

เปรียบเทียบคือ การกำหนดเงื่อนไขแบบฮิวริสติกเพื่อกำหนดระดับที่เหมาะสมในการตัดโครงสร้างต้นไม้ ผลจากการศึกษาวิเคราะห์เปรียบเทียบจะช่วยให้สามารถพัฒนาเทคนิคใหม่ที่ช่วยให้การสร้างและกำหนดความซับซ้อนของโมเดลในลักษณะต้นไม้ตัดสินใจมีประสิทธิภาพมากขึ้น การลดความซับซ้อนจะส่งผลให้ลดเวลาในการสังเคราะห์โครงสร้างต้นไม้ นอกจากนี้ยังช่วยลดเนื้อที่หน่วยความจำที่ต้องใช้ในการเก็บแต่ละกิ่งของโครงสร้างต้นไม้ ประโยชน์โดยตรงของการใช้หน่วยความจำลดลงคือช่วยให้โปรแกรมสังเคราะห์โมเดลทำงานกับข้อมูลขนาดใหญ่ได้

<u>ผลที่ได้ :</u>

ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลมาตรฐานที่นิยมใช้ในการศึกษาเปรียบเทียบประสิทธิภาพการทำเหมืองข้อมูล รายละเอียดของข้อมูลแสดงได้ดังตารางที่ 1 ผลการทดสอบเปรียบเทียบประสิทธิภาพการลดความซับซ้อนของโมเดลและคุณภาพของโมเดลที่ได้ระหว่างวิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติและวิธีเชิงฮิวริสติกที่ใช้พื้นฐานจากทฤษฎีสารสนเทศ แสดงได้ดังตารางที่ 2 และ 3 ตามลำดับ

ตารางที่ 1 รายละเอียดข้อมูลที่ใช้ในการทดลอง

ชื่อชุดข้อมูล	จำนวนข้อมูลฝึก	จำนวนข้อมูลทดสอบ	จำนวนแอททริบิวต์	จำนวนแอททริบิวต์ชนิดสัญลักษณ์	จำนวนแอททริบิวต์ชนิดตัวเลข
Adult	32,561	16,281	15	8	6
Credit-German	666	334	21	13	7
Hepatitis	103	52	20	13	6
Mushroom	5,416	2,708	23	22	-
Vote	300	135	17	16	-

ตารางที่ 2 ผลการทดสอบเปรียบเทียบประสิทธิภาพการลดความซับซ้อนของโมเดลระหว่างวิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติและวิธีเชิงฮิวริสติกที่ใช้พื้นฐานจากทฤษฎีสารสนเทศ แสดงด้วยเปอร์เซ็นต์การลดจำนวนโหนดในโครงสร้างต้นไม้ที่เป็นโมเดลผลลัพธ์

ชื่อชุดข้อมูล / วิธีลดความซับซ้อนของโมเดล	Adult	Credit-German	Hepatitis	Mushroom	Vote
วิธีเชิงฮิวริสติกที่ใช้พื้นฐานจากทฤษฎีสารสนเทศ	22.12%	34.27%	26.95%	35.72%	41.12%
วิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติ	12.96%	25.18%	36.54%	38.78%	47.83%

ตารางที่ 3 ผลการทดสอบเปรียบเทียบประสิทธิภาพของโมเดลที่ได้จากการลดความซับซ้อนด้วยวิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติและวิธีเชิงฮิวริสติกที่ใช้พื้นฐานจากทฤษฎีสารสนเทศ

ชื่อชุดข้อมูล / วิธีลดความซับซ้อนของโมเดล	Adult	Credit-German	Hepatitis	Mushroom	Vote
วิธีเชิงฮิวริสติกที่ใช้พื้นฐานจากทฤษฎีสารสนเทศ	82.13%	74.25%	76.92%	95.79%	91.85%
วิธีวิเคราะห์ความสัมพันธ์ในเชิงสถิติ	82.99%	75.15%	86.54%	98.71%	97.04%

<u>สรุปผลการทดลอง :</u>

การลดความซับซ้อนของโมเดลจากทั้งสองวิธีการจากผลการทดลองจะเห็นได้ว่าวิธีการทางสถิติให้ผลลัพธ์เป็นโมเดลที่มีความเที่ยงตรงสูงกว่า แต่ขนาดของโมเดลจะใหญ่กว่าโมเดลที่ได้จากวิธีการเชิง ฮิวริสติก แนวทางการพัฒนาในอนาคตคือการสร้างวิธีการลดความซับซ้อนที่มีคุณภาพสูงเทียบเท่ากับวิธีการเชิงสถิติแต่ให้ขนาดของโมเดลที่เล็กลง

ประวัติผู้วิจัย

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา ประเทศสหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของกระทรวงวิทยาศาสตร์ฯ โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "The application of inductive logic programming to support semantic query optimization" หลังสำเร็จการศึกษาได้ปฏิบัติราชการในตำแหน่งอาจารย์ประจำสาขาคอมพิวเตอร์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ต่อมาในปีพุทธศักราช 2543 ได้มาปฏิบัติงานในตำแหน่งอาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จนถึงปัจจุบัน งานวิจัยที่ทำในขณะนี้คือการพัฒนาระบบเหมืองข้อมูลประสิทธิภาพสูงที่สามารถทนต่อข้อมูลรบกวน และการเพิ่มความสามารถในการจัดการความรู้ของระบบเหมืองข้อมูล