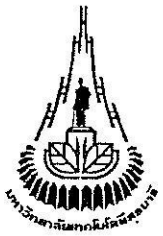
รหัสโครงการ SUT7-705-48-24-10



รายงานการวิจัย

วิธีการเตรียมข้อมูลอัตโนมัติก่อนการทำเหมืองข้อมูล

(An automated approach to pre- data mining)

ได้รับทุนอุดหนุนการวิจัยจาก
มหาวิทยาลัยเทคโนโลยีสุรนารี

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

รหัสโครงการ SUT7-705-48-24-10

รายงานการวิจัย

วิธีการเตรียมข้อมูลอัตโนมัติก่อนการทำเหมืองข้อมูล

(An automated approach to pre- data mining)

ผู้วิจัย

หัวหน้าโครงการ

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พ.ศ. 2548 และ 2549

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

พฤศจิกายน 2552

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณมหาวิทยาลัยเทคโนโลยีสุรนารี และสำนักงานคณะกรรมการวิจัยแห่งชาติที่ได้จัดสรรงบประมาณในการทำวิจัยให้ในปีงบประมาณ 2548 และ 2549 การดำเนินงานของโครงการวิจัยนี้ยังได้รับงบประมาณบางส่วน รวมถึงความร่วมมือในการดำเนินงานจากทีมงานของหน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ (Data Engineering and Knowledge Discovery -- DEKD -- Research Unit) สุดท้ายผู้วิจัยขอขอบคุณผู้ทรงคุณวุฒิที่ได้เสียสละเวลาตรวจข้อเสนอโครงการ และตรวจร่างรายงานการวิจัยฉบับสมบูรณ์

บทคัดย่อภาษาไทย

การสร้างแบบจำลองข้อมูลในลักษณะของโมเดลที่มีความแม่นตรง เป็นจุดมุ่งหมายหลักของกระบวนการทำเหมืองข้อมูล การจะได้ผลผลิตสุดท้ายเป็นโมเดลที่แม่นตรงจะต้องใช้การค้นหาจากข้อมูลที่ได้รับการเก็บรวมรวมและผ่านกระบวนการเตรียมข้อมูลมาอย่างดี ข้อมูลที่ดีและมีคุณภาพจะเป็นปัจจัยสำคัญ ที่ช่วยให้โปรแกรมทำเหมืองข้อมูลสามารถค้นหาโมเดลที่ดีได้ภายในระยะเวลาอันสั้น แต่การเตรียมข้อมูลเป็นงานที่เสียเวลามากและมักจะต้องทำด้วยมือ หรือใช้โปรแกรมพื้นฐานอื่นๆ ช่วยเช่นโปรแกรมสเปรดชีตหรือแผ่นตารางทำการ งานวิจัยนี้จึงถูกเสนอขึ้นเพื่อพัฒนาเครื่องมืออัตโนมัติที่จะช่วยในกระบวนการเตรียมข้อมูล กระบวนการนี้จะประกอบด้วยสามขั้นตอนย่อยได้แก่ การแปลงรูปแบบข้อมูล การปรับปรุงข้อมูลให้สมบูรณ์ การคัดเลือกและลดขนาดข้อมูล ผู้วิจัยได้พัฒนาเทคนิคและเครื่องมือในการแปลงข้อมูลที่เป็นข้อความให้เป็นรูปแบบฮอร์นคลอส หรือข้อความแบบฮอร์นที่จะสามารถประมวลผลได้โดยโปรแกรมเชิงตรรกะ ในกรณีที่ข้อมูลมีบางค่าสูญหาย เครื่องมือที่พัฒนาขึ้นสามารถจัดการได้ด้วยสามแนวทางคือ การแทนที่ค่าสูญหายด้วยค่าคงที่บางค่า การแทนที่ค่าที่สูญหายด้วยค่าส่วนใหญ่ หรืออาจจะตัดทิ้งแอททริบิวต์ที่มีค่าสูญหายในสัดส่วนที่สูงมาก ด้านการลดขนาดข้อมูลงานวิจัยนี้ใช้เทคนิคการสุ่มเป็นพื้นฐานหลัก เครื่องมือที่พัฒนาขึ้นเพื่อการสุ่มข้อมูลประกอบด้วยเทคนิคการสุ่มแบบใส่ค่ากลับคืน การสุ่มแบบไม่ใส่ค่ากลับคืน และการสุ่มตามความหนาแน่นของกลุ่มข้อมูล จากการทดสอบเครื่องมือต่างๆที่พัฒนาขึ้นพบว่าสามารถลดเวลาการเตรียมข้อมูลลงได้มากและช่วยให้ผู้วิเคราะห์ข้อมูลทำความเข้าใจ รวมถึงทำการสำรวจลักษณะต่างๆ ภายในข้อมูลได้อย่างมีประสิทธิภาพ

บทคัดย่อภาษาอังกฤษ

Modeling data accurately is the main focus of data mining process. Reliable model is the final product of the search process through the well-prepared and properly collected data records. High quality of the original data can greatly help the data mining tools to discover better models in less time. Data preparation is, however, an extremely time-consuming process and traditionally has been done manually or semi-automatically with some simple tools such as spreadsheet. This project is thus proposed to automate the tedious data preparation tasks. These tasks are comprised of data transformation, data cleaning, data selection and reduction. We develop techniques to transform textual data into Horn clauses that are ready to be processed by logic-based mining programs. All missing values in the data set are handled by several methods, i.e. replace with some constants, replace with majority values, or remove feature in which its values are highly missing. Data reduction is achieved via a sampling method. We implement three sampling methods: random sampling with replacement, random sampling without replacement, and density-biased sampling. The developed data preparation tools have been proved beneficial to data analysts as they can reduce the preparation time and gain more insight in their data.

สารบัญ

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

ความสำคัญและที่มาของปัญหาการวิจัย

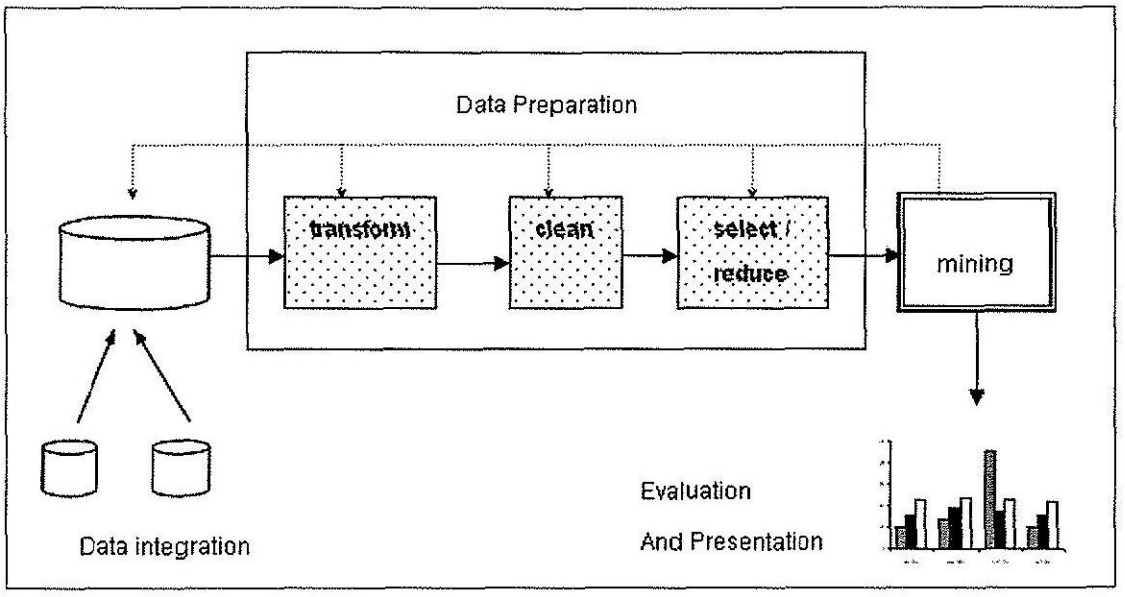
การทำเหมืองข้อมูล (data mining) คือ กระบวนการค้นหาแนวโน้ม รูปแบบร่วม ความสัมพันธ์ หรือความรู้ใหม่อื่นๆ จากข้อมูลจำนวนมาก การทำเหมืองข้อมูลจัดได้ว่าเป็นเทคโนโลยีใหม่ที่ช่วยให้การวิเคราะห์ข้อมูลทำได้โดยอัตโนมัติ และมีประสิทธิภาพสูงขึ้นกว่าที่เคยเป็นมา จึงได้รับความสนใจนำไปใช้อย่างแพร่หลายในทุกวงการ กระบวนการทำเหมืองข้อมูลนี้เปรียบเสมือนการทำเหมืองแร่ที่เราใช้เครื่องจักรคัดแยกแร่ที่เป็นที่ต้องการออกจากกองหิน กรวด ดินที่ปะปนมากับสายแร่ เพียงแต่ในกระบวนการทำเหมืองข้อมูล สิ่งที่เราได้จากกองข้อมูลมหาศาล คือ ความรู้ (knowledge) ที่ซ่อนอยู่ในกองข้อมูล ดังนั้นในระยะหลังจึงเริ่มมีนักวิจัยบางกลุ่มเสนอให้ใช้ชื่อเรียกเทคโนโลยีเช่นนี้ให้ถูกต้องว่า knowledge mining เพราะสิ่งที่เราต้องการจากกระบวนการทำเหมือง คือ ความรู้ แต่ในรายงานการวิจัยนี้จะยังคงใช้คำว่า data mining

ความรู้ที่ได้นี้จะช่วยให้เราเข้าใจลักษณะของข้อมูล และเข้าใจปัจจัยที่ทำให้เกิดลักษณะบางอย่างขึ้นในข้อมูลบางกลุ่ม ซึ่งจะช่วยให้เราสามารถทำนายแนวโน้มของข้อมูลใหม่ที่จะเกิดขึ้นในอนาคตได้อย่างถูกต้องมากขึ้น รวมถึงเข้าใจความสัมพันธ์ที่เชื่อมโยงข้อมูลแต่ละกลุ่มย่อยเข้าด้วยกัน

การทำเหมืองข้อมูลมีลักษณะเป็นสหศาสตร์ที่ต้องการความรู้จากหลากหลายสาขา เช่น database technology, artificial intelligence, machine learning, statistics, information theory, pattern recognition, knowledge-based systems, knowledge acquisition, information retrieval, high-performance computing และ data visualization การทำเหมืองข้อมูลเป็นศาสตร์แขนงใหม่ที่เริ่มเป็นที่รู้จักในช่วงปลายทศวรรษที่ 1980 Piatetsky-Shapiro and Frawley (1991) ได้รวบรวมงานวิจัยในยุคแรกของสาขานี้ไว้ในหนังสือชื่อ Knowledge Discovery in Databases และได้เริ่มมีการรวมตัวของนักวิจัยในสาขานี้ จัดการประชุมทางวิชาการเพื่อเสนอความก้าวหน้าของงานวิจัยภายใต้ชื่อการประชุม International Conference on Knowledge Discovery and Data Mining (KDD) ตั้งแต่ปี 1995 และการประชุมนี้ได้จัดต่อเนื่องมาทุกปีจนถึงปัจจุบันและได้กลายเป็นการประชุมที่สำคัญในหมู่นักวิจัยด้านการทำเหมืองข้อมูลและสาขาอื่นๆที่เกี่ยวข้องกับการทำเหมืองข้อมูล

ปัจจุบันศาสตร์ทางด้าน data mining ได้รับความสนใจจากนักวิจัยจำนวนมากทั่วโลกในสาขาต่างๆ ที่เกี่ยวข้อง ค้นคว้าเพื่อหาแนวทางที่จะทำให้กระบวนการทำเหมืองข้อมูลมีประสิทธิภาพสูงขึ้นและทำงานกับข้อมูลขนาดใหญ่มากได้

โดยทั่วไปกระบวนการทำเหมืองข้อมูลจะประกอบด้วย 6 ขั้นตอนย่อย ดังแสดงในรูปที่ 1.1



รูปที่ 1.1 ขั้นตอนต่างๆในกระบวนการทำเหมืองข้อมูล

การทำเหมืองข้อมูลจัดเป็นกระบวนการ เนื่องจากเมื่อเริ่มตั้งแต่รับข้อมูลที่เป็นอินพุทจนกระทั่งได้โมเดลหรือความรู้ที่เป็นเอาท์พุท จะต้องผ่านขั้นตอนย่อยอีกหกขั้นตอน ได้แก่

- Data integration เป็นขั้นตอนที่รวมข้อมูลที่อาจจะกระจายอยู่ในหลายแหล่งมาไว้ที่เดียวกัน
- Data transformation ทำหน้าที่แปลงไฟล์ข้อมูลที่อาจจะมีรูปแบบแตกต่างกัน ให้อยู่ในรูปแบบที่โปรแกรม mining สามารถเรียกใช้และนำมาประมวลผลได้
- Data cleaning เป็นการปรับปรุงข้อมูลด้วยการกำจัดหรือแก้ไขข้อมูลที่ผิดพลาด และอาจเติมบางส่วนของข้อมูลที่ขาดหายไปให้ครบถ้วนสมบูรณ์
- Data selection / reduction คัดเลือกเฉพาะลักษณะที่สำคัญของข้อมูล (feature selection) และลดขนาดของข้อมูล (data reduction) เพื่อให้โปรแกรม mining สามารถประมวลผลได้
- Data mining เป็นหัวใจสำคัญของกระบวนการทำเหมืองข้อมูล ขั้นตอนนี้เป็นการประยุกต์เทคนิคและวิธีการต่างๆ ที่จะสามารถช่วยให้ค้นหารูปแบบข้อมูลหรือแพทเทิร์น, โมเดลของข้อมูล, ความสัมพันธ์ภายในกลุ่มข้อมูล และความรู้อื่นๆ ที่เป็นประโยชน์จากข้อมูล
- Pattern evaluation and knowledge presentation เป็นการประเมินคุณภาพผลลัพธ์ที่ได้จากการทำ mining ว่ามีความน่าสนใจ หรือมีประโยชน์เพียงใด รวมถึงประเมิน

รูปแบบผลลัพธ์ว่าซับซ้อนเกินกว่าที่จะทำความเข้าใจหรือไม่ ถ้าผลลัพธ์ไม่ผ่านการประเมิน กระบวนการทำเหมืองข้อมูลสามารถย้อนกลับไปปรับเปลี่ยนชุดของข้อมูล หรือเปลี่ยนพารามิเตอร์บางตัวของโปรแกรม mining เมื่อผลลัพธ์ถึงเกณฑ์ที่จะนำใช้งานได้ ผลลัพธ์นั้นจะถูกแสดงออกมาให้ผู้ใช้เห็นในลักษณะต่างๆ เช่น แสดงเป็นข้อความที่จัดรูปแบบแล้ว เป็นสมการคณิตศาสตร์ เป็นภาพ หรือ แผนภูมิ

ขั้นตอนการวิเคราะห์หาโมเดล หรือ mining ถือเป็นขั้นตอนหลักของกระบวนการทำเหมืองข้อมูล แต่โมเดลหรือความรู้ที่ค้นหามาได้จะมีคุณภาพและเป็นประโยชน์เพียงใดนั้น ปัจจัยสำคัญที่สุดคือข้อมูลซึ่งเป็นอินพุทของกระบวนการทำเหมืองข้อมูล ข้อมูลจะต้องอยู่ในรูปแบบที่เหมาะสมและครบถ้วนสมบูรณ์ นอกจากนี้คุณภาพและปริมาณของข้อมูลจะมีผลโดยตรงต่อผลลัพธ์ที่จะได้จากการทำเหมืองข้อมูล คุณภาพของข้อมูลสะท้อนมาจากลักษณะต่างๆ (attributes, or features) ที่ประกอบกันขึ้นเป็นข้อมูลหนึ่งรายการ ถ้าลักษณะต่างๆ ที่รวบรวมและคัดเลือกไว้เป็นลักษณะหลักที่จะสามารถตัดสินพฤติกรรมหรือตัวแบบที่จัดเป็นโมเดลของข้อมูล ผลลัพธ์ของการทำเหมืองข้อมูลก็จะได้ความรู้ที่เป็นประโยชน์

นอกจากคุณภาพแล้ว ปริมาณหรือจำนวนของข้อมูลจะเป็นอีกปัจจัยสำคัญที่ช่วยยืนยันว่าผลลัพธ์ที่ได้นั้นมีความแม่นตรง (accuracy) เพียงใด ในทางอุดมคติจำนวนข้อมูลยิ่งมากเท่าไร จะช่วยให้ผลลัพธ์มีความถูกต้องน่าเชื่อถือมากขึ้นเท่านั้น แต่ในทางปฏิบัติแล้วเนื่องจากทรัพยากรของระบบคอมพิวเตอร์ (ขนาดของหน่วยความจำ และ ความเร็วของหน่วยประมวลผล) มีจำกัด และการทำ mining เพื่อนำความรู้ไปใช้ประโยชน์ควรจะได้ผลลัพธ์ภายในเวลาที่เหมาะสม การลดขนาดของข้อมูลด้วยสัดส่วนที่ถูกต้องจึงเป็นแนวทางแก้ปัญหาของข้อจำกัดดังกล่าว

การทำเหมืองข้อมูลให้มีประสิทธิภาพ จะต้องเริ่มตั้งแต่การเตรียมข้อมูลให้ได้ข้อมูลที่มีคุณภาพ นักวิจัยจำนวนมาก เช่น Aha (1992), Redman (1992), Wang et al. (1995), Wand and Wang (1996), Kennedy et al. (1998), Weiss and Indurkhya (1998), Pyle (1999) และ Ballou and Tayi (1999) ได้อธิบายลักษณะของข้อมูลที่มีคุณภาพและเสนอแนะเทคนิคที่สามารถนำมาใช้ช่วยเพิ่มคุณภาพข้อมูล ลักษณะที่มักจะพบบ่อยในกลุ่มของข้อมูลที่ไม่มีคุณภาพ ได้แก่ ข้อมูลบางส่วนขาดหายไป (missing values) มีผู้เสนอแนวทางจัดการกับกรณีเช่นนี้ไว้หลายแนวทางดังปรากฏในงานวิจัยของ Friedman (1977), Breiman et al. (1984) และ Quinlan (1989)

ประโยชน์ของการทำเหมืองข้อมูลจะเห็นได้ชัดเจนเมื่อข้อมูลมีปริมาณมาก แต่ปัญหาที่เกิดตามมาคือระบบ data mining ไม่สามารถรองรับข้อมูลขนาดมหาศาลนั้นได้ แนวทางแก้ปัญหานี้คือ ใช้วิธีลดขนาดข้อมูล เทคนิคการลดขนาดข้อมูลมักจะกระทำในสองแนวทาง คือ ลดลักษณะ

(attributes, features) ที่อธิบายข้อมูลแต่ละรายการให้เหลือเฉพาะลักษณะหลักที่สำคัญ งานวิจัยในแนวทางนี้จะอยู่ในกลุ่มที่เรียกว่า feature subset selection รายละเอียดปรากฏในหนังสือและงานวิจัยจำนวนมาก (Neter et al., 1996; Dash & Liu, 1997; Kohavi & John, 1997; Dash et al.,1997; Liu & Motoda, 1998)

แนวทางที่สองในการลดขนาดของข้อมูล ใช้วิธีการลดจำนวนรายการข้อมูล (data instances) ให้เหลือเฉพาะข้อมูลที่สามารถเป็นตัวแทนของข้อมูลกลุ่มใหญ่ได้ เทคนิคที่นิยมใช้ลดจำนวนข้อมูลคือ การสุ่ม (sampling) การจัดกลุ่ม(clustering) และการใช้ฮิสโตแกรม(histogram) John and Langley (1996) ได้ศึกษา static และ dynamic sampling ในขณะที่ Josien และคณะ (2001) ได้ทดลองศึกษาพฤติกรรมของโปรแกรม data mining เมื่อข้อมูลถูกลดขนาดลง Kivinen and Mannila (1994) ได้วิเคราะห์ในเชิงทฤษฎีถึงจำนวนข้อมูลที่สามารถเป็นตัวแทนของข้อมูลกลุ่มใหญ่ได้ Barbara และคณะ (1997) รวมถึง Devore and Peck (1997) ได้นำเทคนิคฮิสโตแกรมมาใช้วิเคราะห์การกระจายของข้อมูลเพื่อสุ่มข้อมูลจากแต่ละกลุ่ม

กระบวนการเริ่มต้นในการทำเหมืองข้อมูลจะเป็นการเตรียมข้อมูล (data preparation) งานวิจัยและพัฒนาทั้งหลายที่เกี่ยวข้องกับการเตรียมข้อมูล มักจะทำเพียงเฉพาะบางส่วนของการเตรียมข้อมูล เช่น แก้ไขในกรณีข้อมูลบางส่วนสูญหาย (missing values) หรือเสนอเทคนิคเฉพาะบางเทคนิคเพื่อให้ข้อมูลมีขนาดที่เหมาะสมกับโปรแกรม mining แต่โครงการวิจัยที่เสนอนี้จะ implement ทั้งสามขั้นตอนของการเตรียมข้อมูลคือ data transformation, data cleaning และ data reduction ข้อมูลที่ผ่านการเตรียมแล้วจะสามารถถูกส่งต่อให้กับส่วนทำเหมืองข้อมูล (mining) ต่อไปได้

วัตถุประสงค์ของการวิจัย

เพื่อพัฒนาซอฟต์แวร์ที่จะทำหน้าที่เตรียมและคัดข้อมูลให้มีรูปแบบ คุณภาพ และขนาดที่เหมาะสมกับโปรแกรม mining ในกระบวนการทำเหมืองข้อมูล ซอฟต์แวร์นี้จะประกอบด้วยส่วนประกอบย่อยภายในสามส่วน คือ ส่วน transformation ทำหน้าที่แปลงรูปแบบข้อมูลให้ตรงกับรูปแบบที่โปรแกรม mining ประมวลผลได้ ส่วนประกอบที่สองคือส่วน cleaning ทำหน้าที่ตรวจสอบความครบถ้วนสมบูรณ์ของข้อมูล ถ้ามีข้อมูลบางส่วนหายไปจะต้องสามารถเติมข้อมูลให้สมบูรณ์ได้หรือถ้าข้อมูลมีปริมาณมากสามารถเลือกที่จะตัดรายการข้อมูลที่ไม่สมบูรณ์ทิ้งได้ และส่วนประกอบที่สามคือส่วน reduction ทำหน้าที่ลดขนาดข้อมูล การลดขนาดข้อมูลนี้สามารถทำได้ทั้งการลด feature และการลดรายการข้อมูลด้วยเทคนิคการสุ่ม

ขอบเขตของการวิจัย

โครงการวิจัยนี้มีจุดมุ่งหมายหลักที่จะพัฒนาซอฟต์แวร์ ที่ทำหน้าที่จัดการเกี่ยวกับการเตรียมข้อมูลเพื่อให้สามารถส่งต่อไปยังส่วน mining ได้โดยอัตโนมัติ ในส่วนของการแปลงรูปแบบข้อมูลจะจำกัดขอบเขตของงานเฉพาะการแปลงรูปแบบไฟล์ names file และ data file ที่ใช้ใน UCI repository ให้เป็นรูปแบบ Horn clause ที่สามารถใช้กับโปรแกรม Prolog ได้ รูปแบบของแฟ้มข้อมูลที่ได้จะสามารถใช้ได้กับระบบเหมืองข้อมูล SUT Miner ที่ทีมผู้วิจัยได้พัฒนาขึ้น

การพัฒนาซอฟต์แวร์ต่างๆที่ต้องใช้ในระบบเหมืองข้อมูล SUT Miner จะพิจารณาเฉพาะกรณีที่ข้อมูลเป็นชนิดข้อความ (nominal, categorical) ถ้าหากข้อมูลบางแอททริบิวต์ปรากฏเป็นตัวเลข จะต้องมีการแปลงตัวเลขให้เป็นข้อความ (เรียกว่าการทำ discretization) ซึ่งในขั้นตอนการแปลงตัวเลขเป็นข้อความนี้อยู่นอกเหนือขอบเขตของงานวิจัย

ไฟล์ที่ปรับเปลี่ยนรูปแบบแล้วจะถูกส่งต่อไปให้ส่วน data cleaning ทำการตรวจเช็คและเติมข้อมูลในกรณีที่มีข้อมูลบางส่วนไม่สมบูรณ์ จากนั้นจะเข้าสู่ขั้นตอนการลดขนาดข้อมูลด้วยโปรแกรม data selection/reduction การทำ data cleaning ในงานวิจัยนี้จะเน้นเฉพาะกรณี missing values โดยจะไม่พิจารณากรณี noisy data เนื่องจากถ้าข้อมูลมีปริมาณมากพอ สามารถจัดการกับ noise ได้ด้วยเทคนิค density-biased sampling ที่เป็นฟังก์ชันหนึ่งในขั้นตอน data selection/reduction ที่ผู้ใช้สามารถเลือกใช้งานได้

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

ประโยชน์ที่ได้รับจากการวิจัย

ซอฟต์แวร์ pre-data mining ที่พัฒนาขึ้นนี้มีจุดมุ่งหมายเพื่อใช้ประโยชน์ในงานเตรียมข้อมูลเบื้องต้นก่อนที่จะถึงขั้นตอนการวิเคราะห์ข้อมูลอัตโนมัติ สามารถใช้งานได้จริงกับข้อมูลที่รวบรวมจากงานประเภทต่างๆ ถึงแม้ข้อมูลที่รวบรวมมาจะมีข้อมูลที่บันทึกไม่ครบถ้วน ซอฟต์แวร์นี้ก็ยังสามารถเตรียมข้อมูลเพื่อการสังเคราะห์โมเดลได้ ข้อมูลที่ผ่านการเตรียมแล้วจะสามารถนำไปวิเคราะห์ได้ทั้งในแบบการจำแนกประเภทข้อมูล การจัดกลุ่มข้อมูล การหาความสัมพันธ์ภายในกลุ่มข้อมูล รวมถึงงานวิเคราะห์ข้อมูลในรูปแบบอื่นๆ ดังนั้นซอฟต์แวร์เหมืองข้อมูลที่พัฒนาขึ้นนี้ จึงใช้ประโยชน์ได้กับทุกวงการที่เกี่ยวข้องกับงานวิเคราะห์ข้อมูล โดยจะช่วยให้งานวิเคราะห์ข้อมูลทำ

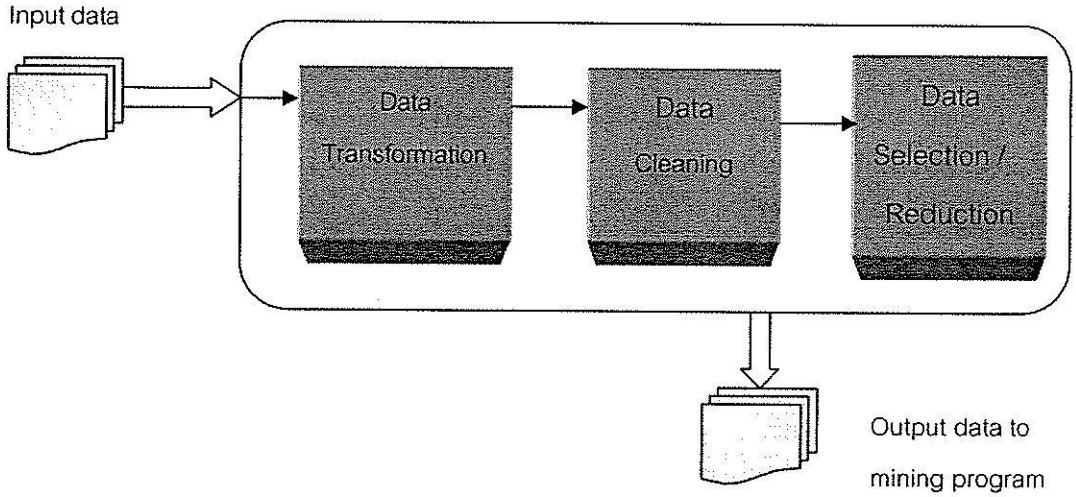
ได้สะดวกและรวดเร็วขึ้น และผู้ใช้ไม่จำเป็นต้องเป็นผู้มีความเชี่ยวชาญเฉพาะในด้านการวิเคราะห์ข้อมูล ประโยชน์จึงเกิดกับทุกหน่วยงานที่มีการเก็บรวบรวมข้อมูลและวิเคราะห์ข้อมูล

ข้อมูลที่ผ่านกระบวนการ pre-data mining แล้ว จะถูกนำไปใช้เป็นข้อมูลเข้าให้กับโปรแกรมวิเคราะห์ข้อมูลอื่นๆของระบบ SUT Miner ซอฟต์แวร์ SUT Miner ที่พัฒนาขึ้นนี้ผู้วิจัยมีเจตนาที่จะเผยแพร่เป็นสาธารณะ (free ware) ผ่านเว็บไซต์ของหน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ (Data Engineering and Knowledge Discovery, DEKD) มหาวิทยาลัยเทคโนโลยีสุรนารี ผู้สนใจในทุกหน่วยงานสามารถนำไปใช้ในงานวิเคราะห์ข้อมูลได้ ซึ่งคาดว่าจะช่วยให้ประเทศชาติประหยัดเงินตราซื้อซอฟต์แวร์สำเร็จรูปเพื่อการวิเคราะห์ข้อมูลลงได้จำนวนมาก

บทที่ 2
วิธีดำเนินการวิจัย

กรอบแนวคิดของงานวิจัย

โครงการวิจัยนี้มีวัตถุประสงค์หลักในการออกแบบและพัฒนาซอฟต์แวร์ ที่ทำหน้าที่เตรียมข้อมูล ให้โปรแกรมทำเหมืองข้อมูลสามารถประมวลผลได้อย่างมีประสิทธิภาพ กรอบแนวคิดเชิงโครงสร้างของซอฟต์แวร์แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างซอฟต์แวร์เตรียมข้อมูลก่อนการทำเหมืองข้อมูล

กระบวนการเตรียมข้อมูลโดยทั่วไปจะประกอบด้วยสามงานหลักคือ การแปลงข้อมูล (data transformation) การปรับปรุงข้อมูล (data cleaning) และการคัดเลือกข้อมูล (data selection/reduction) ซึ่งการทำงานในขั้นตอนย่อยเหล่านี้อาจสลับลำดับแตกต่างจากที่แสดงในแผนภาพข้างต้นได้ ขั้นตอนต่างๆในการดำเนินงานเพื่อพัฒนาซอฟต์แวร์ของโครงการวิจัยนี้ประกอบด้วย

ขั้นตอนที่ 1 รวบรวมเอกสารอ้างอิงที่เกี่ยวข้อง และคัดเลือกข้อมูล

รวบรวมเอกสารเกี่ยวกับเทคนิคการจัดการกรณี missing values และเทคนิคการสุ่มข้อมูล (sampling techniques) รวมถึงคัดเลือกข้อมูลเพื่อใช้ทดสอบซอฟต์แวร์ที่สร้างขึ้น ข้อมูลที่ใช้จะต้องมีขนาดใหญ่เพียงพอที่จะเอื้อให้สามารถทดสอบประสิทธิภาพของส่วน reduction ได้ และเป็นข้อมูลที่มีบางส่วนขาดหายไปเพื่อให้สามารถทดสอบส่วน cleaning ได้ ข้อมูลที่จะคัดเลือกเพื่อใช้ในงานวิจัยนี้ส่วน

หนึ่งจะค้นหามาจาก UCI Repository (http://archive.ics.uci.edu/ml/datasets.html) ซึ่งเป็นแหล่งรวมข้อมูลที่นิยมใช้ในการทำ machine learning

ขั้นตอนที่ 2 ออกแบบโปรแกรม

วางแผนและออกแบบโปรแกรมทั้งสามส่วนของซอฟต์แวร์ และส่วนเชื่อมต่อที่จะประสานการทำงานของทุกส่วน รวมทั้งออกแบบและพัฒนาส่วนที่จะติดต่อกับผู้ใช้ (graphical user interface, GUI)

ขั้นตอนที่ 3 ออกแบบ พัฒนา และทดสอบโปรแกรมส่วน data transformation

โปรแกรมแปลงข้อมูลหรือ data transformation ทำหน้าที่แปลงไฟล์ข้อมูลให้ใช้ได้กับระบบการทำเหมืองข้อมูล ข้อมูลเริ่มต้นที่จะเข้ามาสู่ซอฟต์แวร์ทำเหมืองข้อมูลเป็นข้อมูลที่ประกอบด้วยสองไฟล์ย่อยคือ names file และ data file รูปแบบนี้มักจะปรากฏกับข้อมูลใน UCI repository โปรแกรมแปลงข้อมูลที่พัฒนาขึ้นจะสามารถรวม names file และ data file จากนั้นเปลี่ยนรูปแบบข้อมูลแต่ละรายการให้เป็น Horn clauses ที่ภาษาโปรล็อกสามารถประมวลผลได้

ขั้นตอนที่ 4 ออกแบบ พัฒนา และทดสอบโปรแกรมส่วน data cleaning

อัลกอริทึมสังเคราะห์ความรู้ (learning or mining algorithm) จำนวนมากไม่สามารถทำงานได้ถ้า input file มีข้อมูลไม่ครบ โปรแกรม data cleaning จึงถูกสร้างขึ้นเพื่อทำหน้าที่เติมข้อมูลที่ขาดหายไปเพื่อให้ input file สมบูรณ์ ดังนั้นโปรแกรม data cleaning จะต้องมีความสามารถตรวจหา missing values ใน input file ได้โดยอัตโนมัติ และบรรจุค่าทดแทนส่วนที่ขาดหายไป แนวทางที่ใช้ในงานวิจัยนี้มีได้ 3 แนวทางคือ

(1) ทดแทนด้วยค่าคงที่ที่สร้างขึ้นมาใหม่, หรือ

(2) ทดแทนด้วยค่าส่วนใหญ่ของแอททริบิวต์นั้นๆ, หรือ

(3) ผู้ใช้อาจเลือกตัดแอททริบิวต์ที่มี missing values ออกไป

ขั้นตอนที่ 5 ออกแบบ พัฒนา และทดสอบโปรแกรมส่วน data selection / reduction

โปรแกรม data reduction จะทำหน้าที่ลดขนาดข้อมูลให้สามารถประมวลผลได้ในขั้น mining การลดขนาดจะใช้เทคนิคการสุ่มข้อมูล (sampling) เป็นหลัก โดยผู้ใช้สามารถกำหนดวิธีการสุ่มข้อมูลได้หลายลักษณะ ได้แก่

- การสุ่มจากข้อมูลดิบโดยไม่ใส่ข้อมูลกลับคืน (random sampling without replacement)

- การสุ่มจากข้อมูลดิบโดยใส่ข้อมูลกลับคืน (random sampling with replacement) ซึ่งวิธีการนี้จะทำให้ข้อมูลที่ถูกสุ่มปรากฏซ้ำได้
- การสุ่มตามความหนาแน่นของข้อมูล (density-biased sampling) วิธีการนี้จะพิจารณาความหนาแน่นของกลุ่มข้อมูลประกอบการสุ่ม ข้อมูลที่มีความหนาแน่นสูงจะถูกคัดเลือกไว้ หลักเกณฑ์ที่ใช้พิจารณาความหนาแน่นของข้อมูล จะประกอบด้วยสองพารามิเตอร์คือ จำนวนแอททริบิวต์ และสัดส่วนความคล้ายของค่าในแอททริบิวต์ โดยสัดส่วนความคล้ายนี้จะคำนวณจากจำนวนข้อมูลที่มีค่าในแอททริบิวต์ตรงกันหารด้วยจำนวนข้อมูลทั้งหมด ผู้ใช้สามารถกำหนดจำนวนแอททริบิวต์ที่จะถูกนำมาเปรียบเทียบค่าและกำหนดเกณฑ์ขั้นต่ำของความคล้าย

ขั้นตอนที่ 6 เชื่อมต่อโปรแกรมทั้งสามส่วนของซอฟต์แวร์เตรียมข้อมูล

เมื่อโปรแกรมทั้งสามส่วนเสร็จสมบูรณ์ จะทดสอบความถูกต้องด้วยข้อมูลที่คัดเลือกไว้ในขั้นตอนแรก โดยใช้ข้อมูลจาก UCI repository เมื่อได้ข้อมูลที่ผ่านกระบวนการเตรียมข้อมูลแล้ว ข้อมูลนั้นจะถูกส่งไปทดสอบการใช้งานได้จริง ด้วยการรันบนระบบเหมืองข้อมูล ในงานวิจัยนี้ทดสอบข้อมูลที่ผ่านกระบวนการเตรียมข้อมูลแล้ว ด้วยการประมวลผลกับโปรแกรมทำเหมืองข้อมูลแบบจำแนก (classification mining program)

การออกแบบและพัฒนาวิธีการแปลงและปรับปรุงข้อมูล

การแปลงรูปแบบไฟล์ข้อมูล (data transformation) จำแนกเป็นสองงานย่อยคือ การรวมสองไฟล์คือ names file และ data file ให้เป็นข้อมูลไฟล์เดียว จากนั้นแปลงข้อมูลให้อยู่ในรูปแบบ Horn clauses ซึ่งเป็นรูปแบบข้อความในภาษาโปรล็อก

ตัวอย่างของข้อมูลในรูปแบบของ UCI repository แสดง names file และ data file ได้ดังรูปที่ 2.2 รายการข้อมูลจะปรากฏใน data file ข้อมูลหนึ่งบรรทัดคือหนึ่งเรคคอร์ด แต่ละแอททริบิวต์คั่นด้วยเครื่องหมาย ‘,’ หรือเรียกว่ารูปแบบ CSV (comma separated value) ส่วนที่เป็นคำอธิบายโครงสร้างข้อมูลจะปรากฏใน names file บรรทัดแรกของ names file จะเป็นค่าที่เป็นไปได้ทั้งหมดของคลาส บรรทัดอื่นต่อจากนั้นจะเป็นคำอธิบายชื่อแอททริบิวต์และค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์นั้น เมื่อข้อมูลเดียวกันนี้ถูกเปลี่ยนให้อยู่ในรูปแบบ Horn clauses จะมีลักษณะดังรูปที่ 2.3

```
yes, no.

outlook: sunny, overcast, rain.
temperature: hot, mild, cool.
humidity: high, normal.
windy: true, false.
```

names file

```
sunny, hot, high, false, no
sunny, hot, high, true, no
overcast, hot, high, false, yes
rain, mild, high, false, yes
rain, cool, normal, false, yes
rain, cool, normal, true, no
overcast, cool, normal, true, yes
sunny, mild, high, false, no
sunny, cool, normal, false, yes
rain, mild, normal, false, yes
sunny, mild, normal, true, yes
overcast, mild, high, true, yes
overcast, hot, normal, false, yes
rain, mild, high, true, no
```

data file

รูปที่ 2.2 ตัวอย่างไฟล์ข้อมูลในรูปแบบ UCI repository

```
% Golf data set
%
%       Header file for attribute declaration
attribute( outlook,     [sunny, overcast, rainy] ).
attribute( temperature, [hot, mild, cool]        ).
attribute( humidity,    [high, normal]           ).
attribute( windy,       [true, false]            ).
attribute( class,       [yes, no]                ).
%       Data instances
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 2.3 ตัวอย่างไฟล์ข้อมูลในรูปแบบ Horn clauses

ข้อมูลในรูปแบบ Horn clauses จะยังคงรักษาโครงสร้างของไฟล์ข้อมูลที่ประกอบด้วยส่วนอธิบายแอททริบิวต์ และส่วนรายละเอียดข้อมูล เพียงแต่ข้อมูลจะถูกเปลี่ยนจากข้อความปกติเป็นข้อความที่เรียกว่า clause ในภาษาโปรล็อกที่มีข้อกำหนดว่าแต่ละ clause จะสิ้นสุดด้วยเครื่องหมาย '.' และข้อมูลหนึ่งรายการจะถูกแปลงเป็น Horn clause หนึ่งคลอส ขั้นตอนในการแปลงรูปแบบข้อมูลจาก UCI repository เป็น Horn clauses จะเริ่มจากการแปลงโครงสร้างข้อมูลใน names file ให้เป็นคลอส attribute(AttributeName, [List-of-AttributeValues]) เมื่อแปลงส่วนโครงสร้างเสร็จแล้ว จึงจะเริ่มอ่าน data file และแปลงข้อมูลแต่ละรายการเป็นคลอส instance(ID, Class= Value, [AttributeName=Value]) รายละเอียดการทำงานแสดงได้ดังอัลกอริทึมที่ 1

Algorithm 1. Data transformation

Input: names file and data file
Output: a data file as Horn clauses

(1) Open names file and read the value list in the first line
(2) Write to the output file the clause 'attribute(class, [value list]).'
(3) While not end-of-file do
(3.1) Read attribute name
(3.2) Read attribute-value list
(3.3) Write to the output file the clause 'attribute(attribute name, [value list]).'
(4) Close names file
(5) Open data file
(6) Set instance counter I = 1
(7) While not end-of-file do
(7.1) Read each value that appears in CSV format
(7.2) Match attribute name with corresponding attribute value
(7.3) Construct attribute-value pair format (i.e. attributeName=value)
(7.4) Remove last attribute-value pair to be class-value
(7.5) Write a clause 'instance(I, class-value, [attribute-value list]).' to output file
(7.6) Increment I
(8) Close data file
(9) Return the output file

ข้อมูลที่ผ่านขั้นตอนการแปลงไฟล์ข้อมูลแล้ว จะถูกส่งต่อมายังส่วนปรับปรุงข้อมูล (data cleaning) การปรับปรุงข้อมูลจะเกิดขึ้นเมื่อข้อมูลปรากฏ missing values ซึ่งในรูปแบบของข้อมูล UCI repository เมื่อข้อมูลบางแอททริบิวต์หายไปหรือไม่ปรากฏค่า จะปรากฏสัญลักษณ์ '?' ในตำแหน่งของข้อมูลที่แอททริบิวต์นั้น ตัวอย่างของข้อมูลที่มี missing values แสดงได้ดังรูปที่ 2.4

```
attribute( outlook,     [sunny, overcast, rainy] ).
attribute( temperature, [hot, mild, cool]        ).
attribute( humidity,    [high, normal]           ).
attribute( windy,       [true, false]            ).
attribute( class,       [yes, no]                ).
%       Data instances
instance(1, class=no, [outlook=sunny, temperature=hot, humidity=?, windy=?]).
instance(2, class=no, [outlook=sunny, temperature=hot, humidity=high, windy=true]).
instance(3, class=yes, [outlook=overcast, temperature=hot, humidity=high, windy=?]).
...
```

รูปที่ 2.4 ตัวอย่างไฟล์ข้อมูลที่ปรากฏ missing values

การปรากฏ missing values จะมีผลเสียต่อกระบวนการทำเหมืองข้อมูล ทำให้ได้โมเดลที่ความเชื่อมั่นลดลง ข้อมูลที่เป็นตัวแทนที่ดีจึงไม่ควรมีค่า missing วิธีจัดการกับกรณีข้อมูลหายไปในงานวิจัยนี้จะสร้างทางเลือกให้ผู้ใช้เลือกได้สามแนวทางคือ (1) ทดแทนค่าที่หายไปด้วยข้อความ 'missing' และเมื่อในโมเดลที่เป็นเอาท์พุทจากการทำเหมืองข้อมูล ปรากฏคำว่า 'missing' ผู้ใช้จะสามารถแปลความหมายได้ว่าหมายถึงการไม่ปรากฏค่าในแอททริบิวต์, (2) ทดแทนค่าที่หายไปด้วยค่าส่วนใหญ่ของแอททริบิวต์ ในกรณีนี้เมื่อผู้ใช้แปลผลโมเดลจะไม่ทราบว่าโมเดลนั้นได้จากการวิเคราะห์ข้อมูลที่มี missing values, และ (3) ผู้ใช้สามารถเลือกตัดทิ้งแอททริบิวต์ที่มี missing values ปรากฏอยู่มาก โดยทางเลือกที่สามนี้จะใช้การเรียกโมดูล feature selection (อัลกอริทึมที่ 3) รายละเอียดการทำงานในส่วนปรับปรุงข้อมูลแสดงได้ดังอัลกอริทึมที่ 2

Algorithm 2 Data cleaning

Input: a data file containing Horn clauses with some missing values
Output: a data file that its contents had been cleaned

```
(1) Show the GUI of data cleaning component
(2) Get the response from user's choice
(3) If choice = 'replace with missing', then
(3.1)       Read each data instance
(3.2)       Replace attributeName=? with attributeName=missing
(4) If choice = 'replace with majority', then
(4.1)       Scan the whole data set
(4.2)       Count the majority value of each attribute
(4.3)       Read each data instance
(4.3)       Replace attributeName=? with attributeName=majority
(5) If choice = 'remove attribute with missing values', then
            Call feature selection module to remove unwanted attributes
(6) Return the output file that all the missing cases have been cleaned
```

การออกแบบและพัฒนาวิธีการคัดเลือกข้อมูล

การคัดเลือกข้อมูลในงานวิจัยนี้หมายถึง การคัดเลือกแอททริบิวต์ (feature selection) และการคัดเลือกข้อมูลบางรายการด้วยวิธีการสุ่ม (random sampling) การคัดเลือกแอททริบิวต์จะมีประโยชน์ในการลดขนาดของโมเดล และการสุ่มจะช่วยลดขนาดของข้อมูล และอาจช่วยให้ได้ข้อมูลที่เป็นตัวแทนที่ดีเพื่อการสังเคราะห์โมเดลที่มีความแม่นตรงสูงขึ้น

การคัดเลือกแอททริบิวต์จะใช้วิธีวิเคราะห์การกระจายของข้อมูล แล้วแสดงเป็นภาพในลักษณะของฮิสโตแกรม (histogram) จากนั้นจะให้ผู้ใช้ระบุชื่อแอททริบิวต์ที่ต้องการคัดเลือกไว้ ขั้นตอนการคัดเลือกแอททริบิวต์แสดงได้ดังอัลกอริทึมที่ 3

Algorithm 3 Feature selection

Input: a data file that contains complete Horn clauses
Output: a data file with reduced features

(1) Show the GUI of feature selection component
(2) Get the user's response to obtain the desired attribute names
(3) Scan the input data file
(3.1) Remove unwanted attribute clauses from the header file
(3.2) Remove unwanted 'attributeName=value' from every instance clauses
(3.3) Write the new clauses to the output file
(4) Return the output file

ในส่วนของการคัดเลือกรายการข้อมูลด้วยวิธีการสุ่ม (random sampling) ผู้ใช้สามารถเลือกวิธีการสุ่มได้ว่าจะใช้การสุ่มแบบมีการใส่ข้อมูลคืนกลับ (random sampling with replacement) หรือการสุ่มแบบไม่ใส่ข้อมูลคืนกลับ (random sampling without replacement) การสุ่มแบบใส่ข้อมูลคืนกลับมักจะใช้ในกรณีที่ข้อมูลที่จะถูกสุ่มมีปริมาณไม่มากนัก และการใส่ข้อมูลคืนกลับมีผลให้ข้อมูลบางรายการถูกสุ่มซ้ำได้มากกว่าหนึ่งครั้ง ส่วนการสุ่มแบบไม่ใส่ข้อมูลคืนกลับจะใช้เมื่อชุดข้อมูลที่จะถูกสุ่มมีปริมาณมาก การสุ่มทั้งสองแบบนี้เป็นการสุ่มแบบอิสระนั่นคือข้อมูลทุกตัวมีโอกาสเท่ากันที่จะถูกเลือก วิธีการสุ่มข้อมูลแสดงได้ดังอัลกอริทึมที่ 4

Algorithm 4 Data sampling

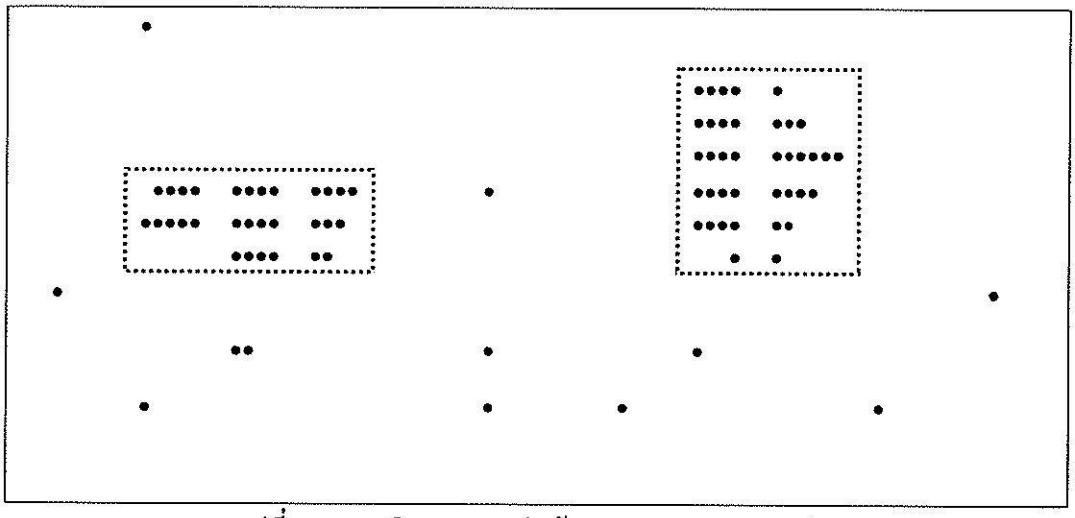
Input: a data file
Output: a data file with reduced instances

(1) Show the GUI of random sampling component
(2) Get the user's response to obtain the desired random sampling methods and the percentage of sample size
(3) Set the instance counter, I = 0
(4) Compute the number of instances to be sampled, S
(5) If choice = 'random sampling with replacement', then
(5.1) Generate random number, N, N ∈ [1..TotalInstances]
(5.2) Write the instance N to the output file
(5.3) Increment I
(5.4) If I<S, then repeat step (5)
(6) If choice = 'random sampling without replacement', then
(6.1) Generate random number, N, N ∈ [1..TotalInstances]
(6.2) If instance N does not appear in the input file,
(6.2.1 Then re-generate the random number N
(6.2.2) Otherwise write the instance N to the output file
(6.3) Delete instance N from the input file
(6.4) Increment I
(6.5) If I<S, then repeat step (6)
(7) Assert header (attribute clauses) to the output file
(8) Return the output file

การคัดเลือกข้อมูลตามความหนาแน่น

การคัดเลือกข้อมูลด้วยการสุ่มในแบบ random sampling จะใช้ลดขนาดข้อมูลได้ดีในกรณีที่ข้อมูลมีการการกระจายอย่างสม่ำเสมอ แต่ในกรณีที่ข้อมูลมีการกระจายไม่สม่ำเสมอ เช่นมีข้อมูลหนาแน่นในบางช่วง แต่ในช่วงอื่นๆมีข้อมูลเบาบางมาก ถ้าให้ใช้วิธี random sampling (ทั้งในแบบ with replacement และ without replacement) ที่ข้อมูลแต่ละตัวมีโอกาสถูกคัดเลือกเท่าๆกัน ข้อมูลที่ถูกสุ่มมาอาจจะไม่คงลักษณะการกระจายตัวเหมือนข้อมูลดั้งเดิมก่อนที่จะถูกสุ่ม ในงานวิจัยนี้จึงเสนอแนวทางการสุ่มข้อมูลแบบที่เรียกว่า การสุ่มตามความหนาแน่น (density-biased sampling) เพื่อประโยชน์ในการคงลักษณะการกระจายตัวของข้อมูล และจะทำให้ได้โมเดลที่สอดคล้องกับรูปแบบข้อมูล

การสุ่มข้อมูลตามความหนาแน่นมีแนวคิดที่แสดงเป็นแผนภาพได้ดังรูปที่ 2.5 ในรูปแสดงข้อมูลแต่ละรายการด้วยสัญลักษณ์ '•' จากรูปจะเห็นได้ว่าข้อมูลมีการเกาะกลุ่มเป็นสองกลุ่มใหญ่ (ตีกรอบกลุ่มข้อมูลด้วยเส้นประ) ในขณะที่ข้อมูลที่อยู่นอกกรอบกระจายตัวอย่างเบาบาง ดังนั้นถ้าต้องการสุ่มข้อมูลตัวแทนจากข้อมูลชุดนี้ ข้อมูลที่ถูกเลือกเป็นตัวแทน ควรจะเป็นข้อมูลจากสองกลุ่มใหญ่นี้ การสุ่มข้อมูลจากกลุ่มที่มีความหนาแน่นสูงจะมีประโยชน์ในการกำจัดข้อมูลที่ไม่เข้ากลุ่ม หรือ outliers และในบางครั้งสามารถกำจัดข้อมูลที่เป็นข้อมูลที่ผิดพลาด หรือ noisy data การกำจัด noisy data และ outliers จะช่วยให้ได้โมเดลที่มีความถูกต้องสูงและลดปัญหา overfitting



รูปที่ 2.5 แนวคิดของการสุ่มข้อมูลตามความหนาแน่น

วิธีการสุ่มข้อมูลตามความหนาแน่น จะให้ผู้ใช้กำหนดเกณฑ์ขั้นต่ำว่าจะพิจารณาความหนาแน่นด้วยค่าในกี่แอททริบิวต์ (M , Minimum number of attributes) และจะต้องการสุ่มข้อมูลที่มีความหนาแน่นขั้นต่ำเท่าไร (D, Density) โดยที่ $D \in [0..1]$ จากนั้นเริ่มต้นทำงานด้วยการเปิดไฟล์และอ่านข้อมูลทีละรายการเพื่อตรวจสอบข้อมูลที่อยู่ใกล้เคียง การวัดความใกล้เคียงใช้การเปรียบเทียบค่าในแต่ละแอททริบิวต์ ถ้ามีค่าคล้ายกันอย่างน้อย M แอททริบิวต์ถือว่าเป็นข้อมูลที่เกาะกลุ่มอยู่ใกล้กัน ตรวจสอบข้อมูลใกล้เคียงเช่นนี้กับข้อมูลทุกรายการ จากนั้นนับจำนวนว่ามีข้อมูลกี่รายการที่จัดว่าอยู่ใกล้เคียง คำนวณค่าข้อมูลใกล้เคียงให้เป็นค่าสัดส่วนโดยหารด้วยจำนวนข้อมูลทั้งหมดในไฟล์ ทำการตรวจสอบเช่นนี้กับข้อมูลทุกรายการ จากนั้นคัดเลือกไว้เฉพาะข้อมูลที่มีค่า D ถึงเกณฑ์ที่ระบุ แล้วสุ่มจากข้อมูลในกลุ่มนี้ให้ได้จำนวนข้อมูลตามที่ต้องการ ขั้นตอนเหล่านี้แสดงในอัลกอริทึมที่ 5

Algorithm 5 Density-biased sampling

Input: a data file, sample size S,
minimum number of matched attributes M, minimum density D

Output: a data file with reduced instances

(1) Show the GUI of density-biased sampling component

(2) Get the user's response to obtain the sample size S,
minimum number of matched attributes M, density threshold D

(3) Set the instance counter, I = 0

(4) Compute the number of instances to be sampled, SS

/* Compute similar instances and their density values */

(5) Open data file and read data instance

(6) For each data instance do

(6.1) Scan data file to collect instances, Ins, with matched attributes ≥ M

(6.2) Compute density, Den, as proportion of Ins to total instances in data file

(6.3) If Den ≥ D, then record this instance in temporary file F

/* Sampling from the dense area */

(7) If instances in F ≤ SS, then output file is F

(8) Otherwise,

(8.1) Generate random number, N, N ∈ [1..TotalInstancesInF]

(8.2) Write the instance N to the output file

(8.3) Increment I

(8.4) If I<SS, then repeat step (8.1)

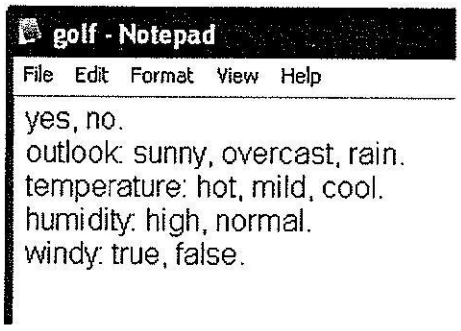
(9) Assert header (attribute clauses) to the output file

(10) Return the output file

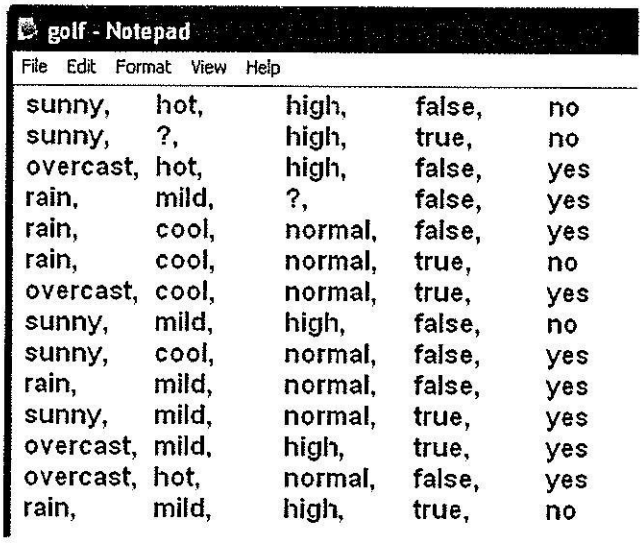
บทที่ 3
การทดสอบโปรแกรม

การทดสอบ Data transformation

การทดสอบความถูกต้องของโปรแกรมในการแปลงรูปแบบแฟ้มข้อมูล ได้ทดสอบกับข้อมูลจาก UCI repository จำนวนหลายชุดข้อมูล แต่ในรายงานนี้จะนำเสนอผลการทดสอบกับข้อมูลเพียงชุดเดียวเท่านั้นเนื่องจากผลการทดสอบเป็นเช่นเดียวกันทั้งหมด ชุดข้อมูลที่ใช้เป็นตัวอย่างคือข้อมูล golf (Quinlan, 1993) ที่เป็นข้อมูลการตัดสินใจของนักกอล์ฟว่าจะออกไปเล่น/ไม่ออกไปเล่นกอล์ฟ การตัดสินใจพิจารณาจากสภาพท้องฟ้า อุณหภูมิ ความชื้นในบรรยากาศ และความแรงของลม ข้อมูลที่ใช้มีจำนวน 14 รายการ เป็นข้อมูลที่เป็นข้อความ (categorical, nominal) ทั้งหมด และเพื่อให้สามารถทดสอบการจัดการกับข้อมูลสูญหายได้ กำหนดให้ข้อมูลสองรายการมีข้อมูลไม่ครบถ้วน (แทนด้วยสัญลักษณ์ '?') ได้แก่ข้อมูลรายการที่สองที่มีค่าในแอททริบิวต์ที่สองหายไป และข้อมูลรายการที่สี่ที่มีค่าในแอททริบิวต์ที่สามหายไป ข้อมูลในส่วน names file (golf.names) และ data file (golf.data) แสดงดังรูปที่ 3.1



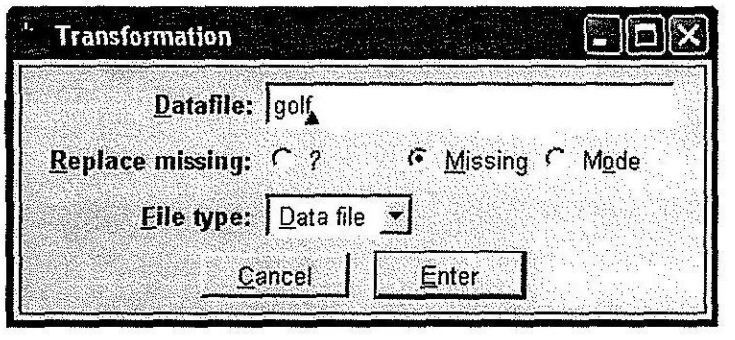
golf.names



golf.data

รูปที่ 3.1 ข้อมูลที่ใช้ทดสอบการทำงานของโปรแกรมเตรียมข้อมูล

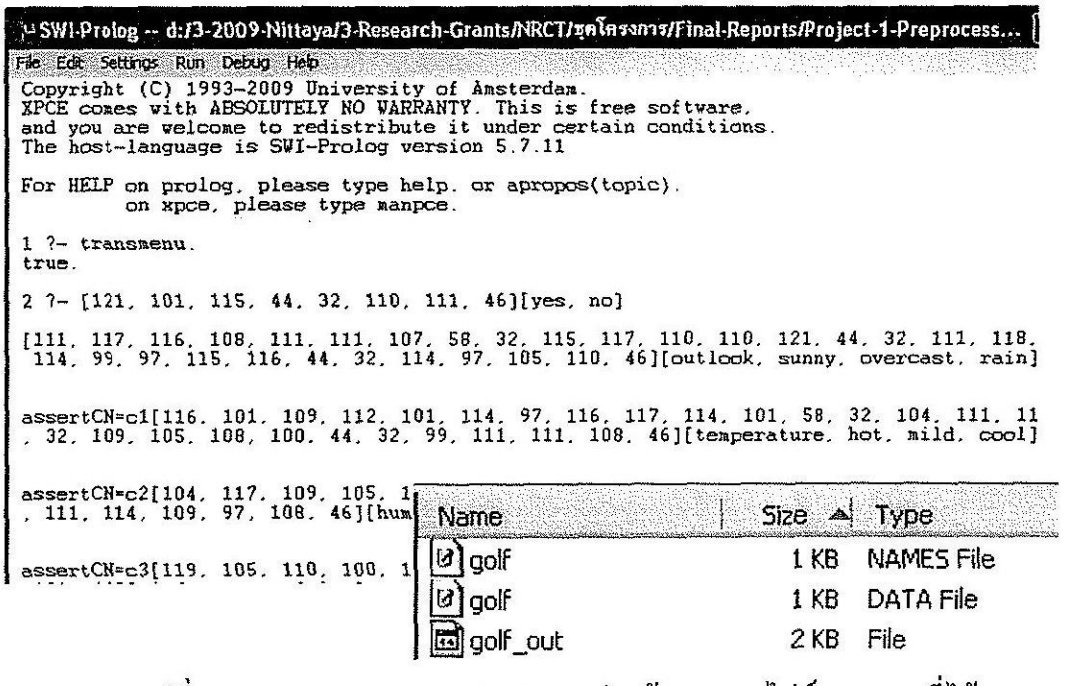
การเตรียมข้อมูลเริ่มต้นจากการเรียกใช้โปรแกรม data transformation ที่จะปรากฏหน้าจ
โต้ตอบกับผู้ใช้ดังแสดงในรูปที่ 3.2 ผู้ใช้จะใส่ชื่อไฟล์โดยไม่ต้องระบุส่วนขยาย พร้อมกันนั้นจะสามาร
สั่งงานในขั้นตอนของการเติมข้อมูลสูญหาย (data cleaning) โดยเลือกวิธีการจัดการกับกรณีข้อมูลสูญหา
ได้สามรูปแบบ คือ แทนด้วย '?' หรือแทนด้วยข้อความ 'missing' หรือแทนด้วยค่าส่วนใหญ่ของข้อมู
(mode)



รูปที่ 3.2 จอภาพส่วน Data transformation

การทดสอบ Data cleaning

จากจอภาพเริ่มต้นดังรูปที่ 3.2 ที่ผู้ใช้เลือกที่จะแทนค่าที่หายไปด้วยข้อความ 'missing' เมื่อคลิกที่ปุ่ม Enter ในหน้าต่างของ SWI Prolog จะแสดงการทำงานในระหว่างการแปลงข้อมูลดัง รูปที่ 3.3 และเมื่อการแปลงข้อมูลเสร็จสิ้น จะปรากฏไฟล์ข้อมูลชื่อ 'golf_out' ในไดเรกทอรี เดียวกับข้อมูล golf.names และ golf.data



รูปที่ 3.3 จอภาพของ SWI Prolog ขณะแปลงข้อมูล และไฟล์ golf_out ที่ได้

ข้อมูลในไฟล์ golf_out ที่ได้จากการรวมไฟล์ golf.names และ golf.data รวมทั้งได้มีการแทนที่ค่าสูญหายในข้อมูลรายการที่สองและรายการที่สี่ด้วยข้อความ 'missing' แสดงได้ดังรูปที่ 3.4 แต่ถ้าเปลี่ยนวิธีการจัดการกับข้อมูลสูญหายให้แทนที่ด้วยค่าส่วนใหญ่ (นั่นคือผู้ใช้เลือก mode) จะปรากฏข้อมูลใน golf_out ดังรูปที่ 3.5 ที่นำค่าส่วนใหญ่ในแอททริบิวต์นั้นมาแทนที่

golf_out - Notepad

File Edit Format View Help

```
% file golf_out
name(golf).
missingT(missing).
attribute(class, [yes, no]).
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(temperature, [missing, hot, mild, cool]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
instance(1, class=no, [outlook=sunny, temperature=hot, humidity=high, windy=false]).
instance(2, class=no, [outlook=sunny, temperature=missing, humidity=high, windy=true]).
instance(3, class=yes,[outlook=overcast, temperature=hot, humidity=high, windy=false]).
instance(4, class=yes,[outlook=rain, temperature=mild, humidity=missing, windy=false]).
instance(5, class=yes,[outlook=rain, temperature=cool, humidity=normal, windy=false]).
instance(6, class=no, [outlook=rain, 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=rain, 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=rain, temperature=mild, humidity=high, windy=true]).
```

รูปที่ 3.4 ข้อมูลในไฟล์ golf_out ที่ค่าที่สูญหายถูกแทนที่ด้วยข้อความ 'missing'

golf_out - Notepad

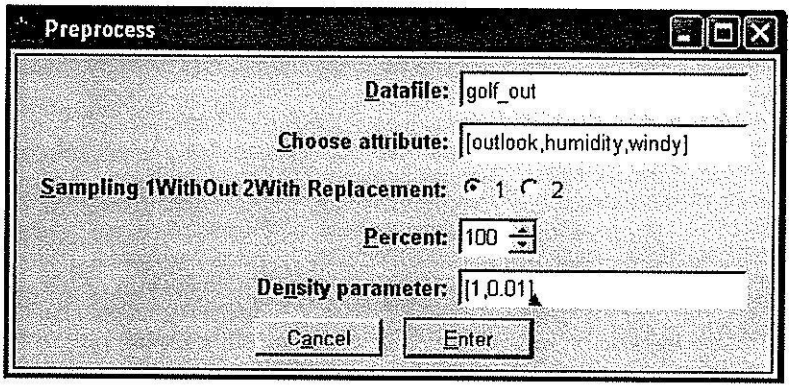
File Edit Format View Help

```
% file golf_out
name(golf).
missingT(missing).
attribute(class, [yes, no]).
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(temperature, [missing, hot, mild, cool]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
instance(1, class=no, [outlook=sunny, temperature=hot, humidity=high, windy=false]).
instance(2, class=no, [outlook=sunny, temperature=mild, humidity=high, windy=true]).
instance(3, class=yes,[outlook=overcast, temperature=hot, humidity=high, windy=false]).
instance(4, class=yes,[outlook=rain, temperature=mild, humidity=normal, windy=false]).
instance(5, class=yes,[outlook=rain, temperature=cool, humidity=normal, windy=false]).
instance(6, class=no, [outlook=rain, 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=rain, 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=rain, temperature=mild, humidity=high, windy=true]).
```

รูปที่ 3.5 ข้อมูลในไฟล์ golf_out ที่ค่าที่สูญหายถูกแทนที่ด้วยค่าส่วนใหญ่

การทดสอบ Feature selection

เมื่อมีการเรียกใช้ฟังก์ชัน feature selection ในโมดูล preprocess ที่เป็นโมดูลสำหรับคัดเลือกข้อมูล จะปรากฏหน้าจอดังรูปที่ 3.6 ในหน้าจอโต้ตอบกับผู้ใช้จะปรากฏกรอบข้อความให้ผู้ใช้พิมพ์ชื่อไฟล์ที่มีรูปแบบข้อมูลเป็น Horn clauses ซึ่งจากตัวอย่างก่อนหน้าไฟล์ที่ใช้จะเป็นชื่อ golf_out ในบรรทัดต่อมาจะเป็นส่วนที่ให้ผู้ใช้ระบุชื่อแอททริบิวต์ที่ต้องการ รายชื่อแอททริบิวต์จะพิมพ์อยู่ภายในวงเล็บ [] ซึ่งแทนโครงสร้างลิสต์ในภาษาโปรล็อก จากภาพระบุชื่อแอททริบิวต์ [outlook, humidity, windy] โดยตัดทิ้งแอททริบิวต์ temperature ในบรรทัดที่สามและสี่เป็นการระบุวิธีสุ่มข้อมูลและปริมาณข้อมูลที่ต้องการ ถ้าผู้ใช้ต้องการทำ feature selection อย่างเดียวสามารถระบุค่าเปอร์เซนต์เป็น 100 ข้อมูลที่ได้หลังจากทำ feature selection แสดงได้ดังรูปที่ 3.7



รูปที่ 3.6 จอภาพส่วน Feature selection และ Data sampling

File Edit Format View Help

```
%Density Parameter=[1,0.01] Sampling[Percent,Type]=[100,1]

%Want 14 records, but has 14 records
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
attribute(class, [yes, no]).
instance(1,  class=yes,  [outlook=overcast,  humidity=high,     windy=true]).
instance(2,  class=yes,  [outlook=rain,      humidity=missing,  windy=false]).
instance(3,  class=no,   [outlook=sunny,     humidity=high,     windy=true]).
instance(4,  class=yes,  [outlook=sunny,     humidity=normal,   windy=true]).
instance(5,  class=yes,  [outlook=overcast,  humidity=normal,   windy=false]).
instance(6,  class=yes,  [outlook=overcast,  humidity=normal,   windy=true]).
instance(7,  class=no,   [outlook=sunny,     humidity=high,     windy=false]).
instance(8,  class=yes,  [outlook=rain,      humidity=normal,   windy=false]).
instance(9,  class=no,   [outlook=sunny,     humidity=high,     windy=false]).
instance(10, class=yes,  [outlook=sunny,     humidity=normal,   windy=false]).
instance(11, class=yes,  [outlook=rain,      humidity=normal,   windy=false]).
instance(12, class=yes,  [outlook=overcast,  humidity=high,     windy=false]).
instance(13, class=no,   [outlook=rain,      humidity=normal,   windy=true]).
instance(14, class=no,   [outlook=rain,      humidity=mild,     windy=true]).
```

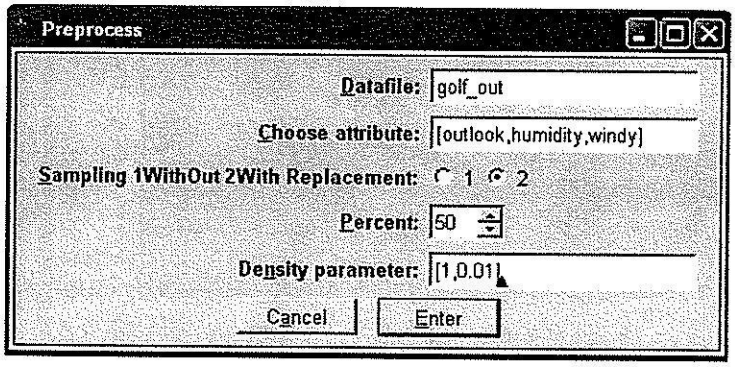
รูปที่ 3.7 ผลลัพธ์ของ Feature selection โดยระบุแอททริบิวต์ [outlook, humidity, windy]

ในขั้นตอนของการเตรียมข้อมูลและการคัดเลือกข้อมูล โปรแกรมที่พัฒนาขึ้นได้สร้างการกระจายข้อมูลในลักษณะของฮิสโตแกรม แสดงไว้เป็น comment ที่ส่วนต้นของไฟล์ golf_out

```
/*---Histogram---
Column outlook
     sunny          5  XXXXX
     overcast       4  XXXX
     rain           5  XXXXX
Column temperature
     hot            3  XXX
     ?              1  X
     mild           6  XXXXXX
     cool           4  XXXX
Column humidity
     high           6  XXXXXX
     ?              1  X
     normal         7  XXXXXXX
Column windy
     false          8  XXXXXXXX
     true           6  XXXXXX
*/
```

การทดสอบ Sampling with replacement

จากข้อมูลเริ่มต้นในรูปแบบ Horn clauses จำนวน 14 รายการที่มีการแทนค่าที่สูญหายในรายการที่สองและสี่ด้วยข้อความ 'missing' เมื่อนำมาทดสอบโปรแกรมสุ่มข้อมูลในรูปแบบการสุ่มแบบใส่ค่ากลับคืนด้วยเปอร์เซนต์การสุ่ม 50% ได้ผลลัพธ์เป็นข้อมูลที่ลดจำนวนลงครึ่งหนึ่ง (นั่นคือจะเหลือข้อมูลเพียง 7 รายการ) ดังรูปที่ 3.8 และเนื่องจากการสุ่มเป็นแบบที่มีการใส่ค่ากลับคืนในผลลัพธ์จึงปรากฏข้อมูลซ้ำคือข้อมูลในรายการที่ 1 และรายการที่ 5



File Edit Format View Help

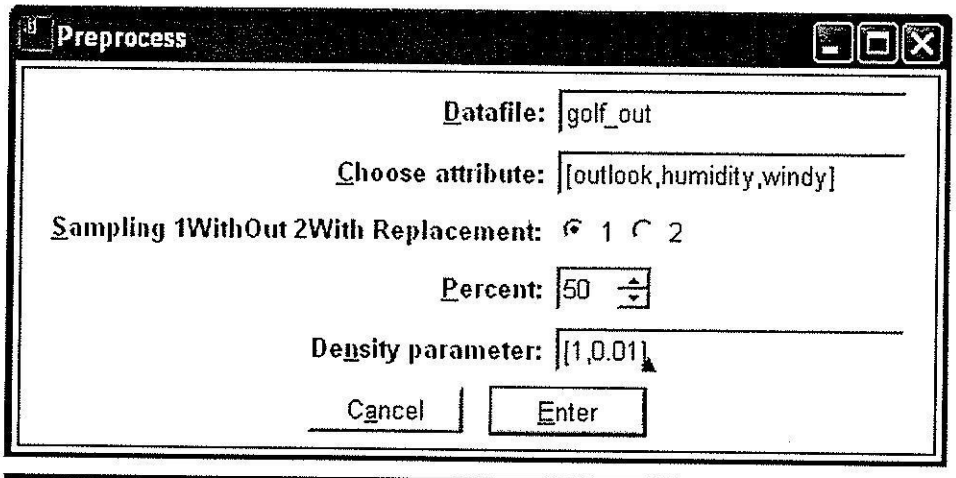
```
%Density Parameter=[1,0.01]  Sampling[Percent,Type]=[50,2]

%Want 7 records, but has 14 records
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
attribute(class, [yes, no]).
instance(7,  class=yes,  [outlook=overcast,  humidity=normal,  windy=true]).
instance(5,  class=yes,  [outlook=rain,      humidity=normal,  windy=false]).
instance(11, class=yes,  [outlook=sunny,     humidity=normal,  windy=true]).
instance(3,  class=yes,  [outlook=overcast,  humidity=high,    windy=false]).
instance(1,  class=no,   [outlook=sunny,     humidity=high,    windy=false]).
instance(1,  class=no,   [outlook=sunny,     humidity=high,    windy=false]).
instance(5,  class=yes,  [outlook=rain,      humidity=normal,  windy=false]).
```

รูปที่ 3.8 ผลลัพธ์ของการสุ่มข้อมูลแบบมีการใส่ค่ากลับคืน

การทดสอบ Sampling without replacement

การลดขนาดของข้อมูลด้วยการสุ่มแบบไม่ใส่ค่ากลับคืน ทดสอบกับข้อมูล golf_out และระบุขนาดของข้อมูลที่ต้องการสุ่มเป็น 50% แสดงผลทดสอบความถูกต้องของโปรแกรมได้ดังรูปที่ 3.9 จากรูปจะเห็นว่าข้อมูลที่ปรากฏในผลลัพธ์มีจำนวน 7 รายการซึ่งคิดเป็น 50% ของข้อมูลตั้งต้น และรายการข้อมูลที่ปรากฏจะไม่ซ้ำกัน



File Edit Format View Help

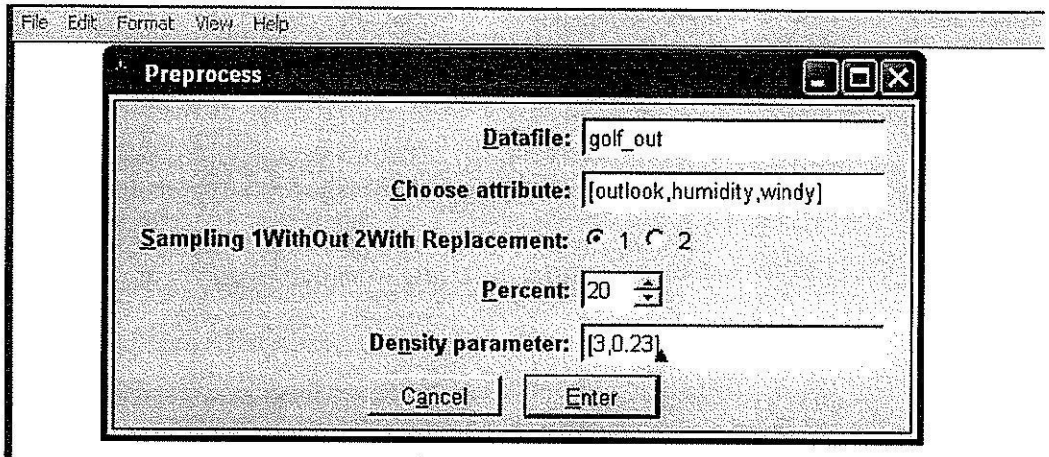
```
%Density Parameter=[1,0.01]  Sampling[Percent,Type]=[50,1]

%Want 7 records, but has 14 records
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
attribute(class, [yes, no]).
instance(3,   class=yes,  [outlook=overcast,  humidity=high,    windy=false]).
instance(2,   class=no,   [outlook=sunny,     humidity=high,    windy=true]).
instance(5,   class=yes,  [outlook=rain,      humidity=normal,  windy=false]).
instance(11,  class=yes,  [outlook=sunny,     humidity=normal,  windy=true]).
instance(1,   class=no,   [outlook=sunny,     humidity=high,    windy=false]).
instance(12,  class=yes,  [outlook=overcast,  humidity=high,    windy=true]).
instance(8,   class=no,   [outlook=sunny,     humidity=high,    windy=false]).
```

รูปที่ 3.9 การสุ่มข้อมูลที่ขนาด 50% และไม่มีการใส่ค่ากลับคืน

การทดสอบ Density-biased sampling

การสุ่มตามความหนาแน่นเป็นวิธีการสุ่มที่จะต้องมีเกณฑ์กำหนดสองอย่างคือ จำนวนแอททริบิวต์ที่จะใช้เปรียบเทียบข้อมูลแต่ละรายการกับข้อมูลอื่นๆ เพื่อคำนวณค่าความหนาแน่นของข้อมูลใกล้เคียง เกณฑ์ที่สองคือสัดส่วนขั้นต่ำของจำนวนข้อมูลที่อยู่ใกล้เคียง ค่าสัดส่วนนี้จะอยู่ระหว่าง [0.0-1.0] เช่นถ้าข้อมูลทั้งหมดมีจำนวน 10 รายการ สัดส่วนขั้นต่ำที่ 0.2 จะหมายถึงต้องมีจำนวนข้อมูลใกล้เคียงปรากฏอยู่อย่างน้อยสองรายการ การทดสอบความถูกต้องของโปรแกรมสุ่มข้อมูลตามความหนาแน่นตามที่ปรากฏในรูปที่ 3.10 เป็นการสุ่มข้อมูลด้วยปริมาณ 20% นั่นคือจะต้องสุ่มข้อมูลมา 3 รายการ มีการกำหนดเกณฑ์ความหนาแน่นเป็น [3, 0.23] หมายถึงพิจารณาสุ่มข้อมูลที่มีความหนาแน่นหรือมีข้อมูลอยู่ใกล้เคียงอยู่ในสัดส่วนขั้นต่ำ 0.23 หรือคิดเป็นข้อมูล 3 รายการจากทั้งหมด 14 รายการ เกณฑ์ความใกล้เคียงพิจารณาจากค่าที่เหมือนกันอย่างน้อย 3 แอททริบิวต์ จากเกณฑ์ความหนาแน่นที่กำหนดทำให้มีข้อมูลที่สุ่มได้เพียงหนึ่งรายการ จากที่ต้องการทั้งหมดสามรายการ ทั้งนี้เนื่องจากระบุเกณฑ์ความหนาแน่นสูงเกินไป

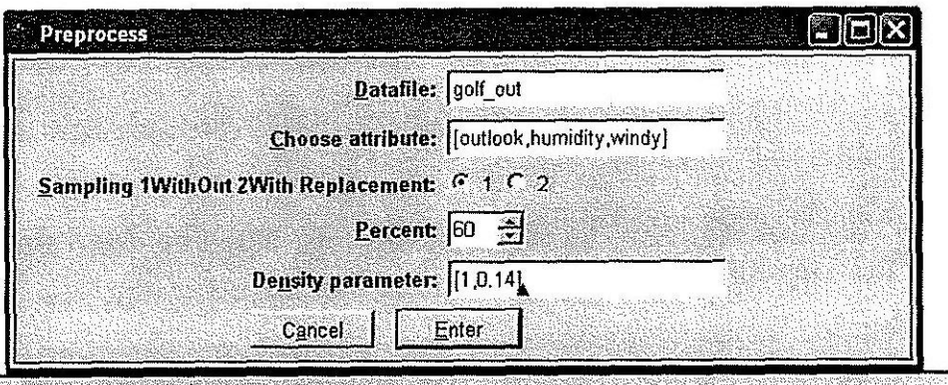



```
%Density Parameter=[3,0.23]  Sampling[Percent,Type]=[20,1]

%Want 3 records, but has 1 records
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
attribute(class, [yes, no]).
instance(5,  class=yes,  [outlook=rain,  humidity=normal,  windy=false]).
```

รูปที่ 3.10 การสุ่มตามความหนาแน่นด้วยเกณฑ์ขั้นต่ำ [3, 0.23]

จากข้อมูล golf_out เมื่อเปลี่ยนลดเกณฑ์ค่าความหนาแน่นในการสุ่มข้อมูล โดยกำหนดให้พิจารณาค่าที่เหมือนกันจากจำนวนแอททริบิต์อย่างน้อยหนึ่งแอททริบิวต์ ความหนาแน่นขั้นต่ำของข้อมูลคือ 0.14 หรือคิดเป็นจำนวนข้อมูลอย่างน้อยสองรายการ ขนาดของข้อมูลสุ่มที่ต้องการคือ 60% จะได้ผลลัพธ์ดังรูปที่ 3.11



File Edit Format View Help

%Density Parameter=[1,0.14] Sampling[Percent,Type]=[60,1]

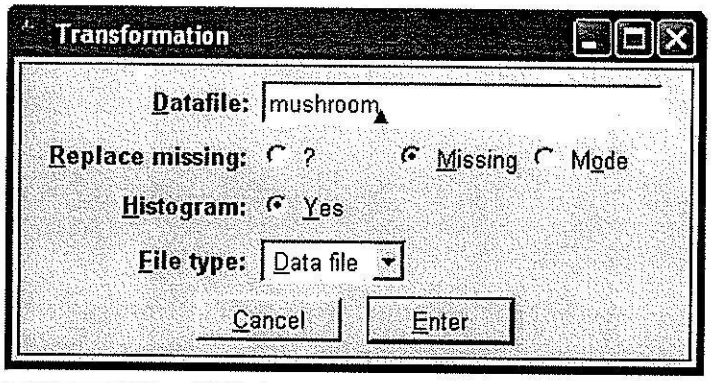
%Want 8 records, but has 14 records
attribute(outlook, [missing, sunny, overcast, rain]).
attribute(humidity, [missing, high, normal]).
attribute(windy, [missing, true, false]).
attribute(class, [yes, no]).

instance(4,	class=yes,	[outlook=rain,	humidity=missing,	windy=false]).
instance(5,	class=yes,	[outlook=rain,	humidity=normal,	windy=false]).
instance(11,	class=yes,	[outlook=sunny,	humidity=normal,	windy=true]).
instance(10,	class=yes,	[outlook=rain,	humidity=normal,	windy=false]).
instance(7,	class=yes,	[outlook=overcast,	humidity=normal,	windy=true]).
instance(13,	class=yes,	[outlook=overcast,	humidity=normal,	windy=false]).
instance(12,	class=yes,	[outlook=overcast,	humidity=high,	windy=true]).
instance(1,	class=no,	[outlook=sunny,	humidity=high,	windy=false]).

รูปที่ 3.11 ผลลัพธ์ของการสุ่มตามความหนาแน่นด้วยเกณฑ์ขั้นต่ำ [1, 0.14]

การทดสอบในขั้นตอนสุดท้ายของกระบวนการเตรียมข้อมูลก่อนการทำเหมืองข้อมูล คือการทดสอบการใช้งานได้ของข้อมูลที่ผ่านกระบวนการเตรียม โดยการทดสอบในขั้นตอนนี้จะใช้ข้อมูลขนาดใหญ่ ได้แก่ข้อมูล mushroom ที่บันทึกรายละเอียดของเห็ดชนิดต่างๆ เพื่อจำแนกว่าเห็ดนั้นๆ เป็นพิษ หรือสามารถรับประทานได้ ข้อมูล mushroom นี้มีจำนวน 22 แอททริบิวต์ และมีจำนวนรายการข้อมูล 5416 instances ข้อมูล mushroom ในฐานข้อมูล UCI จะประกอบด้วยสองไฟล์ คือ mushroom.names และ mushroom.data

เมื่อข้อมูลทั้งสองไฟล์ถูกส่งเป็นอินพุทเข้ามายังโปรแกรมเตรียมข้อมูล จะได้ผลลัพธ์เป็นข้อมูลไฟล์เดียวคือ mushroom_out ที่อยู่ในรูปแบบของ Horn clauses ที่สามารถประมวลผลได้ด้วยระบบ SUT-Miner ในการทดสอบการใช้งานได้ของข้อมูล จะใช้โปรแกรมการทำเหมืองข้อมูลแบบจำแนกเพื่อให้ได้โมเดลของข้อมูลว่า เห็ดที่รับประทานได้และเห็ดที่เป็นพิษ มีลักษณะอย่างไร ผลการทดสอบข้อมูลสามารถถูกประมวลผลได้เป็นผลสำเร็จ และได้โมเดลข้อมูลดังแสดงในรูปที่ 3.12




```
File  Edit  Settings  Run  Debug  Help
% mushroom_data compiled 1.12 sec, 0 bytes

   odor=missing => [ (class=edible)/0]
   odor=almond => [ (class=edible)/274]
   odor=anise => [ (class=edible)/257]
   odor=creosote => [ (class=poisonous)/135]
   odor=fishy => [ (class=poisonous)/368]
   odor=foul => [ (class=poisonous)/1411]
   odor=musty => [ (class=poisonous)/28]
   odor=none
     spore-print-color=missing => [ (class=edible)/0]
     spore-print-color=black => [ (class=edible)/875]
     spore-print-color=brown => [ (class=edible)/929]
     spore-print-color=buff => [ (class=edible)/35]
     spore-print-color=chocolate => [ (class=edible)/32]
     spore-print-color=green => [ (class=poisonous)/41]
     spore-print-color=orange => [ (class=edible)/27]
     spore-print-color=purple => [ (class=edible)/0]
     spore-print-color=white
       habitat=missing => [ (class=edible)/0]
       habitat=grasses => [ (class=edible)/177]
       habitat=leaves
         population=missing => [ (class=edible)/0]
         population=abundant => [ (class=edible)/0]
         population=clustered => [ (class=poisonous)/9]
         population=numerous => [ (class=edible)/0]
         population=scattered => [ (class=edible)/0]
         population=several => [ (class=edible)/33]
         population=solitary => [ (class=edible)/0]
       habitat=meadows => [ (class=edible)/0]
       habitat=paths => [ (class=edible)/33]
       habitat=urban => [ (class=edible)/0]
       habitat=waste => [ (class=edible)/118]
       habitat=woods
         population=missing => [ (class=edible)/0]
         population=abundant => [ (class=edible)/0]
         population=clustered => [ (class=edible)/0]
         population=numerous => [ (class=edible)/0]
         population=scattered => [ (class=edible)/0]
         population=several => [ (class=poisonous)/20]
         population=solitary => [ (class=edible)/5]
     spore-print-color=yellow => [ (class=edible)/30]
   odor=pungent => [ (class=poisonous)/180]
   odor=spicy => [ (class=poisonous)/399]

Size of tree: 42 internal nodes and 38 leaf nodes.

ROBUST-TREE:: robust level 0,  Model building time = 21.801  sec.
```

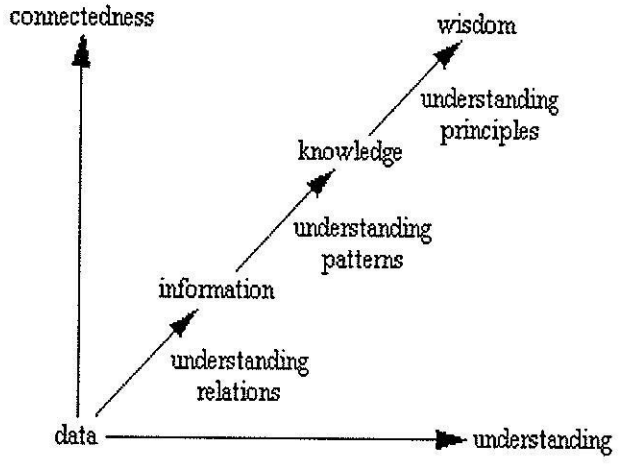
รูปที่ 3.12 ผลลัพธ์ของการเตรียมข้อมูลและประมวลผลด้วยโปรแกรมทำเหมืองข้อมูลแบบจำแนก

บทที่ 4
บทสรุป

สรุปผลการวิจัย

การทำเหมืองข้อมูลเป็นวิวัฒนาการของเทคโนโลยีฐานข้อมูล ที่ต้องการใช้ประโยชน์จากข้อมูลให้มากที่สุด การประมวลผลข้อมูลทำให้ทราบสารสนเทศที่ช่วยสร้างความเข้าใจเกี่ยวกับรายละเอียดข้อมูลได้มากขึ้น การทำเหมืองข้อมูลเป็นพัฒนาการของการประมวลผลอีกขั้นหนึ่งที่ช่วยให้ทราบรูปแบบหรือแพทเทิร์นของข้อมูลและสารสนเทศ การทราบรูปแบบจะช่วยให้เราคาดหมายหรือทำนายแนวโน้มการเปลี่ยนแปลงของข้อมูลที่จะเกิดขึ้นในอนาคตได้ วิวัฒนาการจากข้อมูลไปเป็นสารสนเทศและจากสารสนเทศไปเป็นความรู้แสดงได้ดังรูปที่ 4.1 (http://www.outsights.com/systems/dikw/dikw.htm)

สารสนเทศ (information) เกิดจากการรวบรวมและประมวลข้อมูลเพื่อให้ข้อมูลที่อาจจะถูกเก็บกระจัดกระจาย แต่ข้อมูลเหล่านั้นมีความเกี่ยวข้องเชื่อมโยงกันจึงถูกนำมารวมเป็นกลุ่มเดียวกันและแสดงให้อยู่ในรูปแบบที่ง่ายต่อการทำความเข้าใจ ส่วนความรู้ (knowledge) เกิดจากนำข้อมูลที่มีการเชื่อมโยงนั้นมาวิเคราะห์หารูปแบบของการเชื่อมโยง หรือหาปัจจัยหลักที่ก่อให้เกิดการเชื่อมโยง เมื่อวิเคราะห์ทราบเหตุและปัจจัยแล้วประโยชน์สูงสุดของการประมวลข้อมูลคือช่วยให้เรามีความฉลาด (wisdom) เหตุแห่งความฉลาดคือการวิเคราะห์จากข้อมูล จนกระทั่งสามารถสร้างความเข้าใจในเชิงหลักการเกี่ยวกับข้อมูลนั้นได้



รูปที่ 4.1 พัฒนาการของข้อมูลที่ถูกเปลี่ยนเป็นสารสนเทศและความรู้

กิจกรรมวิเคราะห์ข้อมูลในลักษณะของการทำเหมืองข้อมูล จัดเป็นกระบวนการใหญ่ที่ประกอบด้วยขั้นตอนย่อยๆ หลายขั้นตอน โดยทั่วไปขั้นตอนเหล่านี้มักจะเริ่มต้นด้วยการรวบรวม

ข้อมูลที่อาจจะกระจายอยู่ในหลายแหล่งหรือบันทึกไว้ในหลายฐานข้อมูล ข้อมูลที่รวบรวมมานี้อาจจะบันทึกไว้ในคลังข้อมูล เพื่อความสะดวกในการเรียกใช้โดยโปรแกรมทำเหมืองข้อมูล หรืออาจจะบันทึกไว้ในลักษณะของไฟล์ข้อมูลไฟล์เดียวก็ได้ โดยปกติแล้วข้อมูลเหล่านี้ยังไม่สามารถใช้ในการทำเหมืองข้อมูลได้โดยตรงเนื่องจากรูปแบบ คุณภาพ และขนาดของข้อมูลยังไม่เหมาะสม จึงต้องมีกระบวนการตรียมข้อมูลก่อนการทำเหมืองข้อมูล

การเตรียมข้อมูลเป็นขั้นตอนที่สำคัญและมักจะใช้เวลามาก เนื่องจากต้องมีการเปลี่ยนรูปแบบแฟ้มข้อมูล (data transformation) มีการตรวจสอบรายการข้อมูลว่ามีความครบถ้วนสมบูรณ์ (data cleaning) เพราะถ้าข้อมูลไม่ครบถ้วน โปรแกรมทำเหมืองข้อมูลจะไม่สามารถประมวลผลได้ นอกจากนี้ยังต้องมีการคัดเลือกข้อมูล (data selection) โดยการเลือกเฉพาะแอททริบิวต์ หรือ feature ที่แสดงลักษณะเด่นของข้อมูลได้ค่อนข้างดี การคัดเลือกแอททริบิวต์ (feature selection) อย่างเหมาะสมจะช่วยให้โปรแกรมทำเหมืองข้อมูลสามารถแสดงโมเดลที่ดีมีความถูกต้องสูง และมีขนาดไม่ใหญ่จนเกินไป การคัดเลือกข้อมูลยังรวมถึงการคัดเลือกรายการข้อมูล (data instances) ที่เป็นตัวแทนที่ดีของข้อมูลกลุ่มใหญ่ การมีข้อมูลตัวแทนที่ดีจะช่วยให้การทำเหมืองข้อมูลแสดงผลลัพธ์เป็นโมเดลที่ถูกต้องมีโครงสร้างไม่ซับซ้อนและได้ผลลัพธ์ในเวลาที่รวดเร็ว การคัดเลือกรายการข้อมูลนี้จะมีความสำคัญมากในกรณีที่ข้อมูลเป็นข้อมูลสตรีม เนื่องจากข้อมูลประเภทนี้จะเกิดขึ้นอย่างต่อเนื่องและมีปริมาณข้อมูลมาก การคัดเลือกเฉพาะข้อมูลตัวแทนมาวิเคราะห์หาโมเดลจึงมีความสำคัญต่อประสิทธิภาพการประมวลผลของโปรแกรมทำเหมืองข้อมูล

การทำเหมืองข้อมูลเป็นกระบวนการที่อาจไม่เสร็จสิ้นในคราวเดียว ถ้าหากผลลัพธ์ที่ได้เป็นโมเดลที่ยังมีความถูกต้องต่ำเกินไป กระบวนการอาจจะต้องวนซ้ำ โดยย้อนกลับมาเริ่มต้นรวบรวมข้อมูล และคัดเลือกข้อมูลใหม่อีกครั้ง จากความสำคัญดังกล่าวของข้อมูลและกระบวนการเตรียมข้อมูล งานวิจัยนี้จึงมีวัตถุประสงค์ที่จะพัฒนาเทคนิคและโปรแกรมที่จะเป็นเครื่องมือช่วยงานให้กับนักวิเคราะห์ข้อมูลที่ต้องการใช้เทคโนโลยีการทำเหมืองข้อมูล

เครื่องมือช่วยเตรียมข้อมูลที่พัฒนาขึ้นนี้ประกอบด้วย

- *โปรแกรมแปลงรูปแบบข้อมูล* ทำหน้าที่รวมข้อมูลและเปลี่ยนรูปแบบข้อมูล ในงานวิจัยนี้ใช้ข้อมูลจาก UCI repository ที่เป็นแหล่งข้อมูลมาตรฐานสำหรับวิเคราะห์และทดสอบในงาน machine learning และ data mining ปัจจุบัน (ปี 2009) แหล่งข้อมูลนี้รวบรวมข้อมูลไว้ราว 187 datasets ส่วนใหญ่เป็นข้อมูลที่ใช้ในงานทำเหมืองข้อมูลประเภทการจำแนก (classification) ข้อมูลแต่ละ dataset จะประกอบด้วยสองไฟล์คือ data file (ไฟล์ที่มีส่วนขยายเป็น .data) เป็นรายละเอียดข้อมูล และ names file (ไฟล์ที่มีส่วนขยายเป็น .names) เป็นคำอธิบายแอททริบิวต์และคลาสของข้อมูล ข้อมูลในทั้งสองไฟล์จะปรากฏในรูปแบบข้อความ โปรแกรมแปลงรูปแบบข้อมูลที่พัฒนาขึ้น จะ

รวมทั้งสองไฟล์และแปลงข้อความให้อยู่ในรูปแบบ Horn clauses ที่โปรแกรมโปรล็อกสามารถประมวลผลได้

- *โปรแกรมปรับปรุงข้อมูล* ทำหน้าที่ตรวจสอบความครบถ้วนของค่าที่ปรากฏในแต่ละรายการข้อมูล ถ้ามีค่าใดสูญหายจะแสดงเมนูให้ผู้ใช้เลือกว่าจะเติมแทนค่าที่สูญหายนั้นด้วยค่าใด ผู้ใช้สามารถเลือกค่าคงที่เช่น '?' หรือ 'missing' เติมลงในตำแหน่งที่มีค่าสูญหาย หรือเลือกให้เติมค่าที่เป็นค่าส่วนใหญ่ของข้อมูลในแอททริบิวต์นั้น ในกรณีที่มีค่าสูญหายเป็นปริมาณมากผู้ใช้อาจเลือกที่จะตัดทิ้งแอททริบิวต์นั้นได้
- *โปรแกรมคัดเลือกข้อมูล* การคัดเลือกข้อมูลในงานวิจัยนี้เป็นการคัดเลือกแอททริบิวต์หรือ feature selection โปรแกรมจะมีฟังก์ชันแสดงการกระจายของข้อมูลในแต่ละแอททริบิวต์ในลักษณะฮิสโตแกรม เพื่อให้ผู้ใช้พิจารณาว่าแอททริบิวต์ใดสมควรถูกคัดเลือกไว้ และแอททริบิวต์ใดสมควรถูกตัดทิ้ง ผลลัพธ์ที่ได้จากโปรแกรมนี้คือไฟล์ข้อมูลที่ผ่านการคัดเลือกแอททริบิวต์แล้ว ไฟล์นี้จะถูกบันทึกเพื่อนำไปใช้งานต่อในขั้นการลดขนาดข้อมูล หรืออาจนำไปใช้ในการทำเหมืองข้อมูลได้ทันที
- *โปรแกรมลดขนาดข้อมูล* ในกรณีไฟล์ข้อมูลมีขนาดใหญ่มากโปรแกรมนี้จะช่วยลดปริมาณข้อมูลด้วยเทคนิคการสุ่ม วิธีการสุ่มจะมีให้ผู้ใช้เลือก 3 วิธีคือ การสุ่มแบบปกติและไม่ใส่ค่าคืนกลับ (random sampling without replacement), การสุ่มแบบปกติและใส่ค่าคืนกลับ (random sampling with replacement), และการสุ่มตามความหนาแน่น (density-biased sampling) นอกจากเลือกวิธีการสุ่มแล้วผู้ใช้จะเลือกขนาดของข้อมูลสุ่มว่าต้องการสุ่มเลือกข้อมูลเป็นปริมาณกี่เปอร์เซนต์ ในกรณีของการสุ่มตามความหนาแน่นผู้ใช้จะสามารถเลือกขนาดของความหนาแน่นที่ต้องการ ข้อมูลที่มีความหนาแน่นไม่ถึงเกณฑ์จะไม่ถูกสุ่มเลือก ข้อมูลที่ได้จากการสุ่มจะบันทึกไว้ในไฟล์ให้โปรแกรมทำเหมืองข้อมูลสามารถประมวลผลต่อได้

การเลือกพารามิเตอร์สำหรับการสุ่มตามความหนาแน่น ผู้ใช้สามารถกำหนดจำนวนแอททริบิวต์ที่จะใช้คำนวณค่าความหนาแน่นเป็นเท่าใดก็ได้ ระหว่าง 1 ถึงจำนวนแอททริบิวต์ทั้งหมดที่ปรากฏในข้อมูล ยิ่งกำหนดจำนวนแอททริบิวต์มากข้อมูลที่จะผ่านเกณฑ์ความหนาแน่นจะมีจำนวนลดลงตามลำดับ ดังนั้นจึงไม่ควรกำหนดเกณฑ์จำนวนแอททริบิวต์มากกว่าครึ่งหนึ่งของแอททริบิวต์ทั้งหมด ในส่วนของการกำหนดค่าความหนาแน่นขั้นต่ำ ผู้ใช้สามารถกำหนดค่าอยู่ระหว่าง [0.0-1.0] ถ้าต้องการคัดเลือกข้อมูลที่หนาแน่นสูง ควรกำหนดค่านี้ให้ค่อนข้างสูง เช่น 0.1 ถึง 0.3

ข้อเสนอแนะ

งานวิจัยนี้จำกัดขอบเขตของการพัฒนาโปรแกรมเพื่อการเตรียมข้อมูลก่อนการทำเหมืองข้อมูล ให้รับข้อมูลในรูปแบบ UCI repository จากนั้นแปลงรูปแบบให้เป็น Horn clauses ส่งต่อให้กับส่วนที่ทำหน้าที่ปรับปรุงข้อมูล คัดเลือกข้อมูล และลดขนาดของข้อมูล การทดสอบโปรแกรมให้ผลที่ตรงตามวัตถุประสงค์ แต่โปรแกรมนี้ยังสามารถพัฒนาให้มีความสามารถสูงขึ้นในด้านต่างๆ ดังนี้

(1) การกำหนดรูปแบบข้อมูล สามารถขยายขอบเขตให้รวบรวมข้อมูลจากฐานข้อมูลที่อาจจะอยู่ในหลายฐานข้อมูลให้เป็นไฟล์เดียว หรืออาจจะพัฒนาต่อไปให้สามารถรวบรวมข้อมูลจากคลังข้อมูลทั้งจาก fact table และ dimension table

(2) การปรับปรุงข้อมูลใช้การพิจารณาค่าที่สูญหาย หรือ missing values โดยยังไม่พิจารณากรณีข้อมูลผิดพลาด (random error, noise) ดังนั้นถ้าต้องการเพิ่มความสามารถของโปรแกรม อาจพิจารณาเพิ่มฟังก์ชันการตรวจสอบข้อมูลผิดพลาดและพัฒนาแนวทางการจัดการกับข้อมูลผิดพลาด นอกจากนี้ในส่วนของการจัดการกับข้อมูลสูญหายอาจเพิ่มเทคนิคการจัดการแบบอื่นๆ เช่นใช้การเติมค่าโดยพิจารณาจากค่าใกล้เคียง หรือทำนายค่าจากแอททริบิวต์อื่น

(3) การคัดเลือกแอททริบิวต์ข้อมูลในงานวิจัยนี้ ใช้วิธีแสดงการกระจายค่าของข้อมูลในลักษณะของฮิสโตแกรม จากนั้นให้ผู้ใช้กำหนดว่าจะคัดเลือกแอททริบิวต์ใดบ้าง ในส่วนนี้สามารถพัฒนาให้ดีขึ้นได้โดยการเพิ่มเทคนิคการพิจารณาความเหมาะสมของแอททริบิวต์ (เช่น คำนวณ gain value) จากนั้นแสดงคำแนะนำให้ผู้ใช้ตัดสินใจว่าควรจะเลือกแอททริบิวต์ใด

(4) การลดขนาดข้อมูลในงานวิจัยนี้ใช้วิธีการสุ่มข้อมูล โดยสร้างวิธีการสุ่มไว้ให้ผู้ใช้เลือกสามวิธี ฟังก์ชันในการสุ่มหรือการคัดเลือกรายการข้อมูลอาจเพิ่มเติมมากขึ้นกว่านี้เช่น พิจารณาวิธีการสุ่มแบบ stratified sampling หรืออาจใช้การทำ clustering ก่อนที่จะสุ่มข้อมูลจากแต่ละคลัสเตอร์

บรรณานุกรม

D.W. Aha. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies, 36*(1), 267-287.

D.P. Ballou & G.K. Tayi. (1999). Enhancing data quality in data warehouse environments. *Communications of ACM, 42*, 73-78.

D. Barbara, W. DuMouchel, C. Faloutsos, P.J. Haas, J.H. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng., V. Poosala, K.A. Ross, & K.C. Servcik. (1997). The New Jersey data reduction report. *Bulletin of the Technical Committee on Data Engineering, 20*, 3-45.

L. Breiman, J. Friedman, R. Olshen, & C. Stone. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth International Group.

M. Dash & H. Liu. (1997). Feature selection methods for classification. *Intelligent Data Analysis: An International Journal,* 1.

M. Dash, H. Liu, & J. Yao. (1997). Dimensionality reduction of unsupervised data. In *Proceedings of 9th IEEE International Conference on Tools with AI (ICTAI)*, 532-539.

J. Devore & R. Peck. (1997). *Statistics: The Exploration and Analysis of Data*. New York: Duxbury Press.

J.H. Friedman. (1977). A recursive partitioning decision rule for nonparametric classifiers. *IEEE Transactions on Computing, 26*, 404-408.

J. Han & M. Kamber. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

G.H. John & P. Langley. (1996). Static versus dynamic sampling for data mining. In *Proceedings 1996 International Conference on Knowledge Discovery and Data Mining (KDD'96)*, 367-370.

K. Josien, G. Wang, T.W. Liao, E. Triantaphyllou, & M.C. Liu. (2001). An evaluation of sampling methods for data mining with fuzzy c-means. In *Data Mining for Design and Manufacturing*, Chapter 15, 351-365. Kluwer Academic Publishers.

R.L. Kennedy, Y. Lee, B. Van Roy, C.D. Reed, & R.P. Lippman. (1998). *Solving Data Mining Problems Through Pattern Recognition*. Upper Saddle River, NJ: Prentice Hall.

J. Kivinen & H. Mannila. (1994). The power of sampling in knowledge discovery. In *Proceedings 13th ACM Symposium on Principles of Database Systems*, 77-85.

R. Kohavi & G.H. John. (1997). Wrappers for feature subset selection. *Artificial Intelligence,* 97, 273-324.

H. Liu & H. Motoda, editors. (1998). *Feature Extraction, Construction, and Selection: A Data Mining Perspective.* Boston: Kluwer Academis Publishers.

J. Neter, M.H. Kutner, C.J. Nachtsheim, & L. Wasserman. (1996). *Applied Linear Statistical Models*, 4th ed. Chicago:Irwin.

G. Piatetsky-Shapiro & W.J. Frawley. (1991). *Knowledge Discovery in Databases.* Cambridge, MA:AAAI/MIT Press.

D. Pyle. (1999). *Data Preparation for Data Mining.* San Francisco: Morgan Kaufmann.

J.R. Quinlan. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

J.R. Quinlan. (1989). Unknown attribute values in induction. In *Proc. 6th Int. Workshop on Machine Learning*, 164-168, Ithaca, NY.

J.R Quinlan. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

T. Redman. (1992). *Data Quality: management and Technology.* New York: Bantam Books.

Y. Wand & R. Wang. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of ACM, 39*, 86-95.

R. Wang, V. Storey, & C. Firth. (1995). A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering, 7*, 623-640.

S.M. Weiss & N. Indurkhya. (1998). *Predictive Data Mining.* San Francisco: Morgan Kaufmann.

ภาคผนวก

ภาคผนวก ก

รหัสต้นฉบับ (Source code) ของโปรแกรม

```
/* ============  Pre-Data Mining ========== */
%
% To run the program, call this procedure:
%                                          transmenu.
%                      Then call:
%                                  premenu.
%------------------------------

:-dynamic amountItem/1,instanceR/3,attributeR/2,columnN/1.

:-dynamic instance/3,instanceM/3,attribute/2,col/1,missingT/1,modeM/2.

/* input has 3 files
      1.'dna_d'-data file
      2.'dna_n'--names file
    (or 3.'dna_t'-test)

   output 1 file: 'dna_out'

   must transform dna_out -> dna_da (final result)
   call run. dna. file_name('dna').
   'dna'=name of file
   file_type('_t'). % _t = test, _d = data
*/

trans(Fn,T,M,His):-
          retractall(modeM(_,_)),retractall(instanceM(_,_,_)),
          retractall(missingT(_)),retractall(columnN(_)),
          clearAll,
          atom_concat(Fn,'.names',Names),
          term_to_atom(MissT,M),
          assert(missingT(MissT)),
          (T='data_file'->atom_concat(Fn,'.data',Data);
                    atom_concat(Fn,'.test',Data)),
          open(Names,read,S1),
                 % open names file(attr file) :read line by line
          readAllNameRec(S1),
          close(S1),
          open(Data,read,S), % open file :read line by line
          readAllDataRec(S),
          close(S),
                 %write to out file
          atom_concat(Fn,'_out',Out),
          (tell(Out),
          format('% file ~w_out
          ~nname(~a).~nmissingT(~a).',[Fn,Fn,MissT]),
          ( His ='yes'-> ((format('~n/*---Histogram---
          '),showHistogram,format('~n*/'));
                 true)),
          ( attribute(A,B),format('~n~w.',[attribute(A,B)]),fail;true),
          ( MissT=mode->
          ( modeSelect,

          findall(_,(instance(A1,B1,L),maplist(change,L,Lnew),assert(in
          stanceM(A1,B1,Lnew)),
            format('~ninstance(~w,~w,~w).',[A1,B1,Lnew]) ),_))
```

```
            ;(instance(I1,I2,I3),format('~ninstance(~w,~w,~w).',[I1,I2,I3
            ]),fail;true)
          )
         ,told) .

change(C=missing,C=M):-modeM(C,M),!.
change(C=V,C=V).

readAllNameRec(S):-    % N is the running number
            mem(0),retractall(cA(_)),
                    % output file
            repeat,
            read_line_to_codes(S,X),
            (X = end_of_file,c(N),assert(cA(N)), !
                        % record the number of attribute
               ;  %not eof
             write(X),
            (member(124,X) *->  append(X2,[124|_],X); X2=X),
                        % delete line comment |

            split_string(X2,L),
            maplist(codes_atom,L,Res),
            write(Res),nl,nl,
            Res \==[] ->
            (   c(N),N1 is N+1,mem(N1),
              add_attr(N1,Res)
              ),
           fail
            ).

add_attr(N,[H|T]):-
       (N==1   *-> (assert(attribute(class,[H|T])) );
                    % ****** first row is class
         (N1 is N-1, atom_codes(N1,Codes),atom_codes(CN,[99|Codes]),
                    %ascii 99 is c
        nl, write(assertCN=CN), missingT(M1),
        ( (M1=missing;M1=mode) ->M=missing;M='?'),
                    %assert(attribute(CN,[M|T]))
        assert(attribute(H,[M|T])) %add missing value
        , assert(columnN(H))   % to store column name

        )% ******* other rows
            ).

% find mode
% % 11 ?- findall(F,(instance(NO,CLASS,LI),member(c2=F,LI)),X),length(X,Len).

find(C,(C,Res)):-
            findall(F,(instance(NO,CLASS,LI),member(C=F,LI)),Res).

% show Histogram
showHistogram:- findMode(R),maplist(writeHis,R).
writeHis((C,[])):-!.
writeHis((C,FreqL)):- format('~nColumn ~w~n',C),
            maplist(writehis,FreqL).
writehis(F-A):-format('~w~t~w~10|',[A,F]),writeBar(F).
```

```
writeBar(0):-nl,!.
writeBar(F):-write('X'),F1 is F-1,writeBar(F1).

 % modeSelect,
 % findall(_,(instance(A1,B1,L),maplist(change,L,Lnew),assert(instanceM(A1,B1,Lnew))),_)
modeSelect:-  findMode(R),maplist(maxmode,R). % assert(mode).

%?- findMode(R),maplist(maxmode,R).
findMode(Result):-findall(C,(attribute(C,L)),AttrList),
          maplist(find,AttrList,Res),
          maplist(mycount,Res,Result).

% (c1,[92-f,3-m])
% mus
maxmode((_,[])):-!.
maxmode((C,L)):-
          findall(F,member(F-_,L),Res1),
          max_list(Res1,FMax),member(FMax-V,L),
          assert(modeM(C,V)),!.

count1([],X,0-X):-!.
count1([X|T],X,Res-X):- count1(T,X,Res1-X),
          Res is Res1+1,!.
count1([_|T],X,Res-X):- count1(T,X,Res-X).

%   +[a,a,a,b,a],-[a-4,b-1]
count(Y,Res):-list_to_set(Y,X),maplist(count1(Y),X,Res).

mycount((C,L),(C,Lnew)):-count(L,Lnew) .

readAllDataRec(S) :-   % N is the running number , now-writing to out file
  mem(0),
  repeat,
  read_line_to_codes(S,X),
  (X = end_of_file, !
   ;
   write(X),
  (member(124,X) *->  append(X2,[124|_],X); X2=X)
      , % delete line comment |
   split_string(X2,L),maplist(codes_atom,L,Res1),
   missingT(M),

   (  (M=missing;M=mode) -> maplist(miss,Res1,Res);Res=Res1),    % check
        if missingT= missing

   write(Res),nl,nl,
   Res \==[] ->
      (     c(N),N1 is N+1,mem(N1),add_inst(N1,Res)),

   fail
  ).


% replace ? with missing
miss('?','missing'):-!.
miss(X,X).
```

```
clearAll:-retractall(instance(_,_,_)),
          retractall(attribute(_,_)),mem(0).

%=============== tokenizer ====================
split_string(S, L) :- phrase(split_str(L), S).

% scan a list of words separated by spaces
%
split_str([H|T]) --> blanks, inwords(H), blanks, !, split_str(T).
split_str([], _, _).

% a word is a sequence of (at least one!) not blanks
%
inwords([C|Cs]) --> [C], { ok(C) }, inwords(Cs).
inwords([C]) --> [C], { ok(C) }.

% skip blanks (test and lose...)
%
blanks --> [C], { ko(C) }, blanks.
blanks --> [].

ok(C) :- \+ ko(C).
  %ko(C) :- code_type(C,period);C==44;C==58.
ko(C) :- code_type(C, space);C==46;C==44;C==58.
  % skip space and dot and comma and colon

codes_atom(C,A):- atom_codes(A,C).   % for maplist-second order

mem(N):-retractall(c(_)),assert(c(N)).

add_inst(N,L):-   last(L,H,X),%  col(C1),    %columnN(ColName),
          findall(Name,columnN(Name),C1),    % <<<< EDIT HAER
          mmerge(C1,H,H1),missingT(Mode),
          assert(instance(N,class=X,H1)) .


last([X],[],X):-!.   % find First and Last
last([H|T],[H|T1],X):- last(T,T1,X).

% max column = 70

col([c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,
      c19,c20,c21,c22,c23,c24,c25,c26,c27,c28,c29,c30,c31,c32,c33,c34,
      c35,c36,c37,c38,c39,c40,c41,c42,c43,c44,c45,c46,c47,c48,c49,c50 ,
      c51 ,c52 ,c53 ,c54 ,c55 ,c56 ,c57 ,c58 ,c59,c60,c61,c62,c63,c64,
      c65,c66,c67,c68,c69,c70 ]).

mmerge(_,[],[]):-!. % merge c1=red
mmerge([],_,[]):-!.
mmerge([H1|T1],[H2|T2],[H1=H2|T3]):-mmerge(T1,T2,T3).

%------------------------- end of transformWeka3.pl--------
```

```
%--------------------Find density ----------
:- dynamic dens_list/1,dens_rec/1,no_rec/2.

% +1,+5,-[1.,2,3,4,5]
runlist(F,F,[F]):-!.
runlist(S,F,[S|T]):- S1 is S+1,runlist(S1,F,T).

comp(E1,E2,V):-(E1==E2->V=1;V=0).
mycompare(L1,L2,CL):-maplist(comp,L1,L2,CL).

%test 2 instances, M=number of match
%+1,+1,+4,1
similar(I1,I2,M,V):- instance(I1,_,L1),instance(I2,_,L2),
          mycompare(L1,L2,VL), %compare 2 instances
          sumlist(VL,SumV),
          (SumV>=M ->V=1 ;V=0   ).


%+[1,2,3,4],+[1,2,3,4],+3,-L).
all_dens(IL1,IL2,M,L):-
          findall((X,Y,V),(member(X,IL1),member(Y,IL2),similar(X,Y,M,V)
          ),L).

% +[(1,1,1),(1,2,0),..],+1, -1-1)
each_dens(L,I,I-Dens):-findall(V,(member((I,_,V),L)),VL),
         sumlist(VL,SumDens),Dens is SumDens/14 .

%+3,-assert(dens_list([1-1,2-1,...]))
%+No_of_att_match,-assert(dens_list)
all_inst(M):-runlist(1,14,L1),
          all_dens(L1,L1,M,L),
          maplist(each_dens(L),L1,DL),
          retractall(dens_list(_)),
        assert(dens_list(DL)),!.

% +3
maindens(M):-all_inst(M),listing(dens_list).


all_rec_dens(M,D):-maindens(M),dens_list(L),
          findall(X,(member(X-D1,L),D1>=D),AllRec),
          retractall(dens_rec(_)),
          assert(dens_rec(AllRec)),!.

%---------------------PREPROCESS---------------
% pre.pl
%

%create_attr(+ListofAttr)
create_attr([]):- H=class,attribute(H,R1),assert(attributeR(H,R1)),!.
create_attr([H|T]):-attribute(H,R1),
          assert(attributeR(H,R1)),!,
          create_attr(T).

choose_sampling(Per,C,A):-
          write('Per,Type,AttrList'+[Per,C,A]),
          sampling(C,Per,A),writeln(samplingA+A).
```

```
init:-      retractall(amountItem(_)),retractall(instanceR(_,_,_)),
            retractall(attributeR(_,_)),retractall(no_rec(_,_)),
            assert(amountItem(14)),!,true.

%random without replacement- L1 for temp List
%rand1(+100,+30,+[],-Res).
rand(1,_,Nsel,L1,[]):-length(L1,Len),Len is Nsel,!.
rand(1,Nall,Nsel,L1,L2):-dens_rec(DensRec),
            H is random(Nall-1)+1,%shift to 1...Nall
            (    (memberchk(H,L1);not(memberchk(H,DensRec)))
                -> rand(1,Nall,Nsel,L1,L2);
                   ( L2=[H|T],L=[H|L1],rand(1,Nall,Nsel,L,T))
            ).

%random with replacement -L1 for temp List
%rand2(+100,+30,+[],-Res).
rand(2,_,Nsel,L1,[]):-length(L1,Len),Len is Nsel,!.
rand(2,Nall,Nsel,L1,[H|T]):-dens_rec(DensRec),
            H1 is random(Nall-1)+1,
            (memberchk(H1,DensRec)->(H=H1,rand(2,Nall,Nsel,[H|L1],T));
                                    rand(2,Nall,Nsel,L1,[H|T]) ).

%sampling(+Type,+Per,+Attribute)
%quit if NoSampling < len of Density List
%sampling(+Type,+Per,+Attribute)
sampling(C,Per,A):-amountItem(N),NoSel is round( (Per/100)*N),
            dens_rec(Rec),length(Rec,DensLen),
            assert(no_rec(NoSel,DensLen)),!,
            ( NoSel>DensLen ->
                  (NumSel=DensLen,
                   format('~n--densRec size<sampling size->choose all
            density list=~a records:~n',
                    [DensLen])) ;
              NumSel=NoSel),
            rand(C,N,NumSel,[],LS),
            create_rec(0,LS,A).

%create sampling rec
%(+0,+ListofRandkey,+Attribute)
create_rec(_,[],_):-true.
create_rec(N,[H|T],A):-instance(H,R1,R2),
            include(filter(A),R2,R22),
            N1 is N+1,
            assert(instanceR(N1-H,R1,R22)),!,
            create_rec(N1,T,A).

%filter(+AttrList,+Element)
% true or false -- filter for selected attri
filter([],_):-false.
filter([H|_],(H=_)):-true,!.
filter([H|T],(M=V)):-M\==H,filter(T,(M=V)).

%TLL=[[outlook+sunny+3, outlook+overcast+1, outlook+rainy+3],...]
%tally(-TLL)
tally(TLL):-findall(A+VL,attributeR(A,VL),L),
            maplist(map,L,LL),tallyAtt(LL,TLL).
```

```
tallyAtt(LL,TLL):-maplist(tallyEach,LL,TLL).
tallyEach(L,TL):-maplist(finda,L,TL).
finda(A+V,(A+V+N)):-findall(A+V,

          (instanceR(_,class=C,L),(member(A=V,L);(A=class,V=C))),Res),
          length(Res,N).
map(A+VL,EL):-maplist(add(A),VL,EL). add(A,B,A+B).


% mainp('data.pl',[outlook,humidity],+20,+1,[3,0.23])<-- called from MENU GUI.

mainp(D1,A1,Per,S,ParaDens1):-
          reconsult(D1),
          init,
          term_to_atom(AL,A1),
          term_to_atom(ParaDL,ParaDens1),
          [M,D]=ParaDL,
          all_rec_dens(M,D),

          create_attr(AL),

        choose_sampling(Per,S,AL),
          tally(TLL),writeln(TLL),
          writeln(end+main), no_rec(Want,Actual),
           tell('ple.pl'),
          format('~n%Density Parameter=[~a,~a]
          Sampling[Percent,Type]=[~a,~a]~n',[M,D,Per,S]),
          format('~n%Want ~a records, but has ~a
          records~n',[Want,Actual]),
          (attributeR(X,Y),write(attribute(X,Y)),writeln('.'),fail;true
          ),
          (instanceR(N4-
          _,K4,L4),write(instance(N4,K4,L4)),writeln('.'),fail;true),

          !,told.

%---------------------MENU---------------------
premenu:-
  new(Dialog,dialog('Preprocess')),send_list(Dialog, append,
  [ new(D1, text_item(datafile,'outlook_out')),
    new(A1, text_item(choose_attribute,'[outlook,humidity,windy]')),
    new(S, new(S, menu('sampling 1WithOut 2With Replacement'))),
    new(Per, int_item('percent', low := 1, high := 100)),
    new(Dens, text_item(density_parameter,'[3,0.23]')),
    button(cancel, message(Dialog, destroy)),
    button(enter, and(message(@prolog,mainp,

                          D1?selection,
                          A1?selection,
                          Per?selection,
                          S?selection,
                          Dens?selection
                         ),
              message(Dialog, destroy)))  % enter&destroy

  ]),
```

```
  send_list(S, append, [1,2]),
  send(Dialog, default_button, enter),
  send(Dialog, open).

transmenu:-
  new(Dialog,dialog('Transformation')),send_list(Dialog, append,
  [ new(D1, text_item(datafile,'outlook')),
    new(S, new(S, menu(replace_missing))),
    new(SS, new(SS, menu(histogram))),
    new(O, menu(file_type, cycle)),
    button(cancel, message(Dialog, destroy)),
    button(enter, and( message(@prolog,trans,
                          D1?selection,
                          O?selection,
                          S?selection,
                          SS?selection
                             ),
              message(Dialog, destroy)))  % enter&destroy

  ]),
  send_list(S, append, ['?',missing,mode]),
  send_list(SS, append, [yes]),
  send_list(O, append, [data_file, test_file]),
  send(Dialog, default_button, enter),
  send(Dialog, open).


% ========= End of Program ===========
```

ภาคผนวก ข

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

- K. Kerdprasop, N. Kerdprasop, and J. Sun (2005). Density biased reservoir sampling for clustering. *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2005)*, Innsbruck, Austria, February 14-16, pp. 95-100.

- T. Thianniwet, N. Kerdprasop, and K. Kerdprasop (2005). The sampling methods for determining clusters in large data sets. *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 T. Thianniwet (2005). เทคนิคการลดขนาดข้อมูลเพื่องานจัดกลุ่มข้อมูลขนาดใหญ่. *Proceedings of the Research Network Development of Higher Education Alliance in Nakhon Ratchasima*, Suranaree University of Technology, Nakhon Ratchasima, Thailand, June 24, pp.66-67.

Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, 14-16 February 2005.

DENSITY BIAS RESERVOIR SAMPLING FOR CLUSTERING

Nittaya Kerdprasop[1], Kittisak Kerdprasop[1], and Junping Sun[2]

[1]Data Engineering and Knowledge Discovery (DEKD) Research Unit,
School of Computer Engineering, Suranaree University of Technology,
Nakhon Ratchasima, Thailand

[2]Graduate School of Computer and Information Sciences
Nova Southeastern University
Fort Lauderdale, Florida 33314, USA
jps@nsu.nova.edu

ABSTRACT

A reservoir-sampling algorithm is a simple random algorithm for drawing a sample of size n without replacement from a population of size N, $N \geq n$. We adopt the algorithm to get the benefit from its advantage of efficient memory usage and extend it to deal with clustering large data of varying cluster sizes. Our proposed algorithm is a density biased sampling using only a single scan of the data. Thus, good efficiency can be expected. Moreover, our experimental results reveal its effectiveness on the subsequent clustering phase. Noise-tolerance is additional characteristics of our method.

KEYWORDS

Clustering, Density biased sampling, Reservoir sampling

1. INTRODUCTION

Clustering is a task of grouping similar objects into classes or clusters based on their attribute values. It is an unsupervised learning in the sense that the class labels of objects are not predefined. The grouping process searches for groups of objects with maximum similarity within group and minimum similarity between groups. Many clustering algorithms determine clusters on the basis of some distance measures such as Euclidean or Manhattan. Based on such measures, discovered clusters tend to have spherical shape with similar size and density. This is an important limitation on clustering real-world data, especially spatial data with non-uniform density among regions. Density-based clustering methods ([1], [2], [3]) have been developed to discover clusters of arbitrary shape. Generally, the methods connect regions with sufficient high density into clusters. Each cluster contains a maximum set of density-connected objects.

However, the inherent complexity of clustering is high and its application on large data sets is time and resource consuming. Sampling is a powerful data reduction paradigm to improve the efficiency of several algorithms ([4], [5]). The sampling techniques used in these algorithms is uniform random sampling, which assigns every object the same probability of being included in the sample. But many data sets in real life do not follow the uniform distribution scheme. It instead seems to

follow the Zipf's distribution ([6]), for instance, income and population distribution. In these data sets, some areas such as large metropolitan area have much higher population density than the small cities. If all the populations have equal opportunity of being selected as a representative, sparse areas may be missed and not be included in the sample.

Density biased sampling ([7]) is a sampling technique that takes into account the different sizes of the groups. Small groups or sparse regions are assigned higher probability to be included in the sample than the large groups or dense regions. By biasing the sampling process, small clusters will not be missed or overlooked as outliers.

In this paper, we propose a novel approach of adapting reservoir technique ([8], [9]) to perform a density biased sampling on large data sets. Our algorithm can obtain a desired sample through a single data set scan. The proposed method is simpler and requires less resource than the hash-based method ([7]). The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 explains reservoir sampling. The proposed algorithm for the density biased reservoir sampling is presented in Section 4. Section 5 shows experimental results on sampling quality and efficiency. Section 6 concludes the paper.

2. Related work

On scalable popular and successful clustering methods such as k-means and k-medoids to work against large data sets, many algorithms like BIRCH ([10]) and CLARANS ([11]) employ the sampling technique to minimize data sets. In BIRCH, a CF-tree structure is built after an initial random sampling step. The CF-tree is used as a summarized data structure with statistical representations of space regions stored on leaf nodes. After the phase of CF-tree building, any clustering algorithm can be applied to the leaf nodes. CLARANS is an extension of k-medoids algorithm. It uses uniform sampling to derive initial representative objects or medoids for the clusters.

Recent advancement on clustering very large data sets in which summarized data structure is even to big to fit into main memory, sampling is independently applied to the data set prior to the subsequent clustering phase. Palmer and Faloutsos ([7]) develop a non-uniform sampling method for clusters that differ very much in size and density. Their method is a generalization of uniform random sampling in that every group of data sets can be assigned different probability of being drawn. When sampling is biased by group density, smaller groups are oversampling, whereas larger groups are undersampling. Since clusters are not known a priori, Palmer and Faloutsos combine the phase of density information extraction with the biased sampling phase using the hash-based approach. They argue that the inherent collision problem of any hash-based approach will not dramatically degrade the sample.

Nevertheless, their method is significantly affected by noise due to the tendency of oversampling noisy area. Our approach adopts the reservoir technique to eliminate the collision problem of hash-based approach and it is independent on the assumption regarding cluster distribution to avoid the impact of noise.

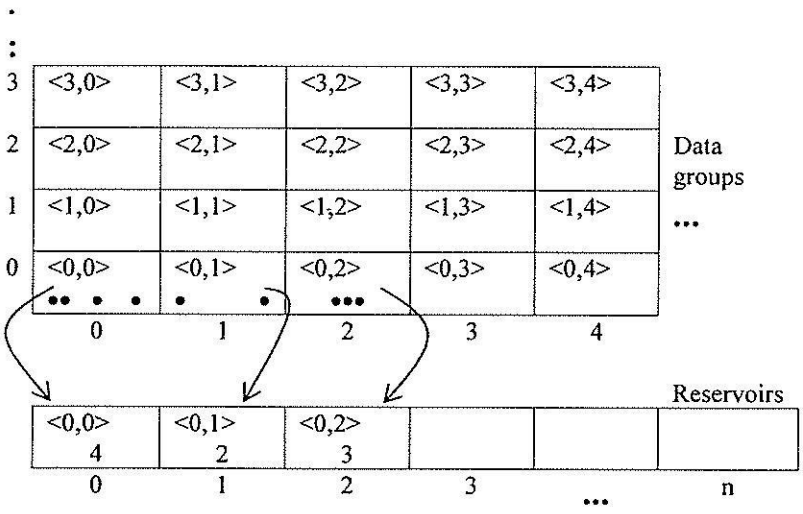
3. Reservoir sampling

A reservoir-sampling algorithm ([8],[9]) is a simple random sampling algorithm for drawing a sample of size n without replacement from a population of size N ($N \geq n$). Vitter ([8]) has developed a one-pass reservoir-sampling algorithm when the population size (N) is unknown and cannot be determined efficiently. The term "reservoir" defines a storage area j ($j \geq n$, but mostly $j = n$) to store the potential candidates of the sample. The j reservoirs is initialized to store the first j records of the file, that is, all areas of the reservoir pool are initially filled up. Then the algorithm starts scanning the remaining part of the file with a randomly skipping steps. The random picked record is evaluated whether to replace the existing one in the reservoir pool. If it passes the test, the position in the reservoir is also randomly selected. The process stops when the end of file has been reached and the records in the reservoir form a simple random sample of the population. The general procedure of reservoir-sampling algorithm ([9],[12]) is given in Figure 1.

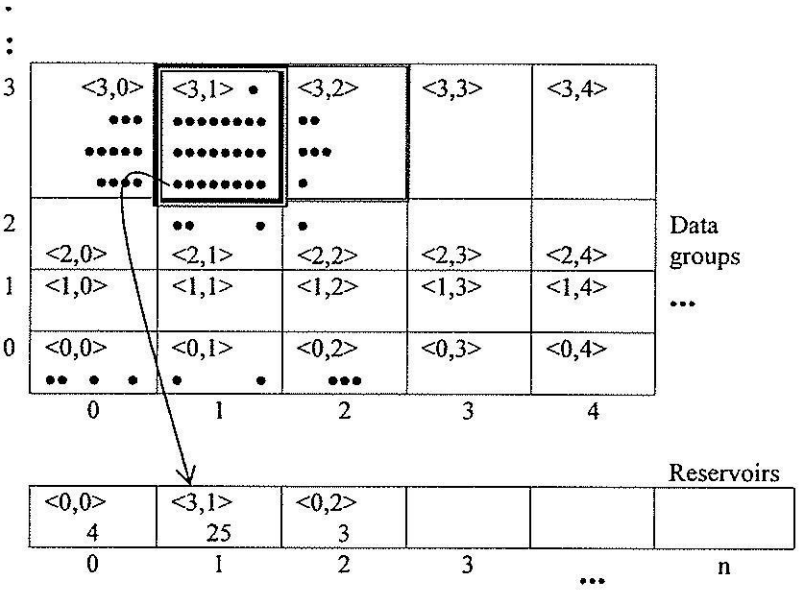
The time complexity of the algorithm is shown ([8],[9]) to be $O(n\ (1+ \log(N/n))$. In the reservoir sampling algorithm, each record of the file is assigned uniform (0,1) random number. When the reservoir is needed to be updated, each record in the reservoir has the same chance to be replaced by the new record.

Algorithm Reservoir sampling
Input: a sequential file of N population
Output: a random sample of size n ($n \leq N$)
Steps:
1) Initialize the reservoir $X_1, \ldots, X_n$ to be the first n records of the file
2) $W \leftarrow \exp(\log(\text{random}()) / n)$ // initialize W to be the largest value in a sample of // size n from the uniform distribution on the // interval (0, 1).
3) While not eof do
4) $S \leftarrow \lfloor \log(\text{random}()) / \log(1-W) \rfloor$ // generate the random variate S to denote // the number of records to be skipped over // before a new record can enter the reservoir
5) If (not eof) Then $Y \leftarrow (S+1)^{\text{th}}$ record // search for the next potential record // to be in the reservoir
6) Else return $X_1, \ldots, X_n$
8) $X_{\lfloor 1+n*\text{random}() \rfloor} \leftarrow Y$ // update X
9) $W \leftarrow W * \exp(\log(\text{random}()) / n)$ // update W
10) End While

Figure 1. Reservoir-sampling algorithm



(a) initialize the reservoir



(b) update reservoir randomly

Figure 2. Density biasing in a reservoir scheme

4. Density biased reservoir sampling

Our sampling algorithm generalizes the reservoir scheme for the case of data with different density distribution. In our proposed method, the initial step of partitioning data into groups resembles that of Palmer and Faloutsos ([7]). But our subsequent steps are not based on hashing scheme in order to avoid the effect of noise and collision problems.

After the initial step of dividing the data space into equally sized bins, the information of the first n groups are put into the n reservoirs residing in main memory (see Figure 2a). The collected information includes the number of points in each group and the id of the group.

The algorithm performs a single scan on a data set in a random manner controlled by a random variate S with the distribution W. The density biasing (step 7 in Figure 3) is achieved through the consideration of two consecutive data groups. If the density difference of the two data groups is above some threshold δ (i.e., detecting cluster edge) or the sum of density on both groups is above the threshold value ε (i.e., avoiding noisy cases), then the denser group is a candidate to be included in a sample. This new candidate is put in a reservoir pool at a random position (the reservoir update is pictorially shown in Figure 2b). The density-biased sampling proceeds until the skipping variate S reaches the end of the data groups.

Algorithm Density-biased reservoir sampling

Input: a data set of N objects
Output: a density-biased sample of size n ($n \leq N$)
Steps:

1) Partition data into g groups (with group-id $1,2,\ldots, g$), $g \geq n$
2) Initialize the reservoir $X_1, \ldots, X_n$ to be the first n <group-id, density>-pairs of the data groups
3) Set $W \leftarrow \exp(\log(\text{random}()) / n)$ // initialize W that will be used in the // generation step of random variate S
4) Set $S \leftarrow \lfloor \log(\text{random}()) / \log(1-W) \rfloor$
5) While $S < g$ do
6) Read data group g_S and g_{S+1} // read two consecutive data groups
7) If ($|\text{density}(g_S) - \text{density}(g_{S+1})| > \delta$) OR ($(\text{density}(g_S) + \text{density}(g_{S+1})) > \varepsilon$)
// δ and ε are predefined density threshold values
Then // randomize the reservoir area to be updated
$X_{1+\lfloor n * \text{random}() \rfloor} \leftarrow$ <group-id, density> of maximum density $\{ g_S, g_{S+1} \}$
8) $W \leftarrow W * \exp(\log(\text{random}()) / n)$ // update W for the skipping process
9) $S \leftarrow \lfloor \log(\text{random}()) / \log(1-W) \rfloor$
// generate the random variate S to denote
// the number of groups to be skipped over
10) End While
11) Return $X_1, \ldots, X_n$

Figure 3. Density-biased reservoir sampling algorithm

5. Experiments and results

We evaluate the performance of the proposed reservoir-based density bias sampling method against the hash-based sampling method ([7]). The efficiency regarding memory usage of our reservoir-based sampling method is obviously better than the hash-based method. In the hashing scheme, some amount of memory is needed to store the hashing table in addition to the memory required for storing the drawn sample. Thus, it requires twice the amount of memory comparative to those required by our method.

Effectiveness of the proposed sampling method is examined by measuring the quality of a sample with respect to the number of correctly found clusters. We run clustering using BIRCH ([10]) algorithm. We use a synthetic data generator to generate d-dimensional data sets having k clusters and N data points. We vary d from 2 to 5, k from 2 to 10, and N from 50,000 to 100,000.

The measurement *Number of Clusters found* (NC) is the metric defined in ([7]). NC is calculated by comparing the distances of the cluster centers found by the clustering algorithm with the true cluster centers. We say that the cluster is found if the calculated distance is less than a predefined threshold (e.g., 0.001).

The results in Figure 4 show the NC when run clustering on various sample sizes with the presence of noise. The reported results are observed from the experiments using 3-dimensional data set having 7 clusters. One cluster contains 50,000 points and the other six clusters contain 500 points. The results obtained from other experiments on data sets with different dimensions, various number of clusters, and varied number of data points are confirmed with the one presented in Figure 4, so we omit them for brevity. The experimental results reveal the efficiency of the biased reservoir method especially in the presence of noise.

6. Conclusions

We propose a density biased sampling technique based on the reservoir method. The inherent advantage of efficient memory usage in the reservoir scheme is adopted and extended with the additional capability of dealing with data that are much different in density distribution. The proposed technique is designed to lessen the effect of noise as it is the case in the hash-based approach. The experimental results have shown that the proposed method is as good as the hash-based method in discovering correct number of clusters. Our method, moreover, is less sensitive to noisy data even when the percentage of noise is greater than 20. The evaluation of the proposed method on real large databases and the consideration of outliers are our future work.

7. 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.

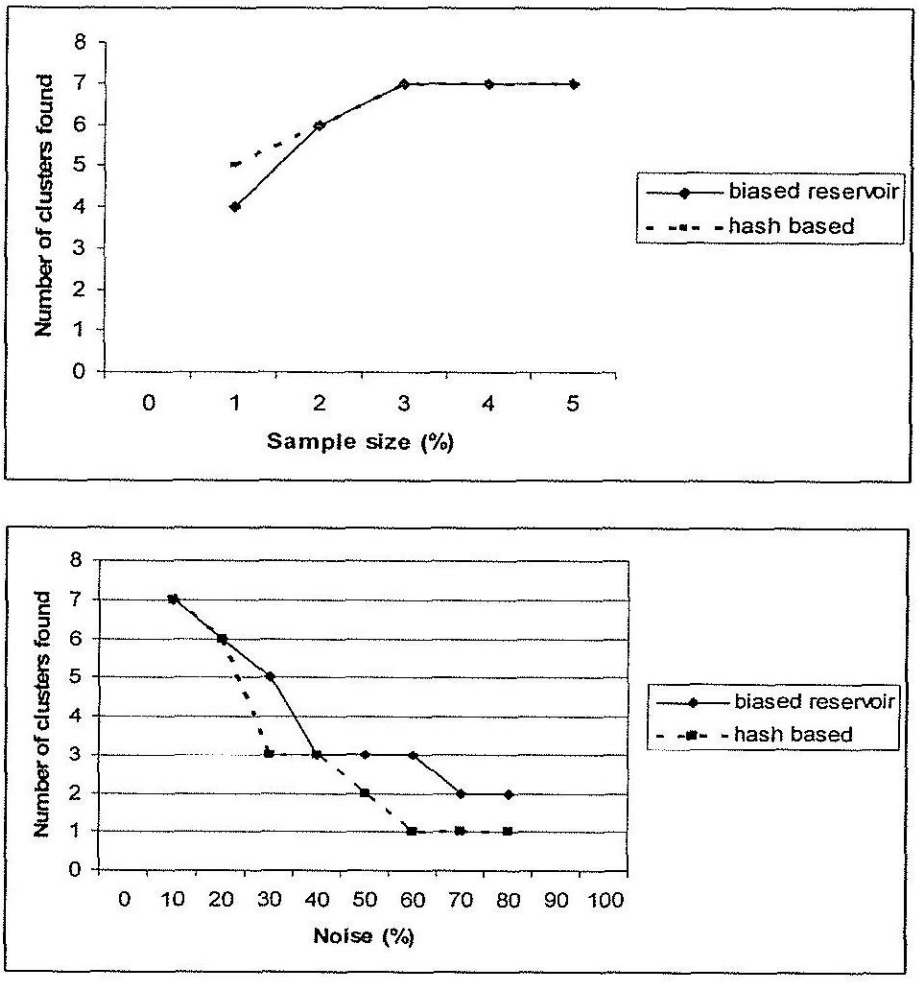


Figure 4. Finding clusters of 3-dimensional data on various sample sizes and in the presence of noise

References:

[1] M. Ester, H.-P. Kriegel, J. Sander, & X. Xu, A density-based algorithm for discovering clusters in large spatial databases, *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, 1996, 226-231.

[2] M. Ankerst, M. Breunig, H.-P. Kriegel, & J. Sander, OPTICS: Ordering points to identify the clustering structure, *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, 1999, 49-60.

[3] A. Hinneburg, & D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, 1998, 58-65.

[4] S. Zhou, A. Zhou, J. Cao, J. Wen, Y. Fan, & Y. Hu, Combining sampling technique with DBSCAN algorithm for clustering large spatial databases, *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2000, 169-172.

[5] A. Nanopoulos, Y. Theodoridis, & Y. Manolopoulos, C^2P: Clustering based on closest pairs, *Proc. Int. Conf. on Very Large Data Bases*, 2001, 331-340.

[6] G.K. Zipf, *Human behavior and principle of least effort: An introduction to human ecology* (Cambridge, MA: Addison Wesley, 1949).

[7] C. Palmer, & C. Faloutsos, Density biased sampling: An improved method for data mining and clustering, *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, 2000, 82-92.

[8] J.S. Vitter, Random sampling with a reservoir, *ACM Transactions on Mathematical Software, 11*(1), 1985, 37-57.

[9] K.-H. Li, Reservoir-sampling algorithms of time complexity O(n(1 + log(N/n))), *ACM Transactions on Mathematical Software, 20*(4), 1994, 481-493.

[10] T. Zhang, R. Ramakrishnan, & M. Livny, BIRCH: An efficient data clustering method for very large databases, *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, 1996, 103-114.

[11] R.T. Ng, & J. Han, Efficient and effective clustering methods for spatial data mining, *Proc. Int. Conf. on Very Large Data Bases*, 1994, 144-155.

[12] L. Devroye, *Non-uniform random variate generation* (New York: Springer-Verlag, 1986).

เทคนิคการสุ่มข้อมูลเพื่องานการจัดกลุ่มข้อมูลขนาดใหญ่

THE SAMPLING METHODS FOR DETERMINING CLUSTERS IN LARGE DATA SETS

ธรรมศักดิ์ เธียรนิเวศน์, กิตติศักดิ์ เกิดประสพ และ นิตยา เกิดประสพ

Thammasak Thianniwet, Kittisak Kerdprasop and Nittaya Kerdprasop

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

บทคัดย่อ: กระบวนการจัดกลุ่มข้อมูลอัตโนมัติบนชุดข้อมูลที่มีขนาดใหญ่มากๆ เป็นกระบวนการที่ต้องใช้เวลาและสิ้นเปลืองหน่วยความจำเป็นจำนวนมาก การลดขนาดข้อมูลเป็นแนวทางหนึ่งที่จะช่วยแก้ปัญหานี้ได้ สำหรับงานวิจัยนี้จะศึกษาและเปรียบเทียบวิธีการลดขนาดข้อมูลด้วยเทคนิคการสุ่มที่เหมาะสมสำหรับงานการจัดกลุ่มข้อมูล เพื่อค้นหารูปแบบการสุ่มที่มีประสิทธิภาพและสามารถสร้างชุดข้อมูลสุ่มที่สามารถเป็นตัวแทนชุดข้อมูลต้นฉบับได้อย่างสมบูรณ์ที่สุด จากการศึกษาพบว่า การสุ่มข้อมูลด้วยเทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นเพียง 2% สามารถให้ผลลัพธ์ของกลุ่มข้อมูลได้เทียบเท่ากับข้อมูลทั้งหมด อีกทั้งยังสามารถลดเวลาในการจัดกลุ่มข้อมูลได้มากกว่า 95% และจากผลการทดลองยังพบว่าเทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นแบบสะสม สามารถสร้างชุดข้อมูลสุ่มที่มีคุณสมบัติของการทนทานต่อข้อมูลรบกวนบนชุดข้อมูลตั้งต้นได้อีกด้วย

Abstract: Determining clusters in large data sets takes a very long time and consumes many resources. Data reduction is an important step to increase the efficiency of determining clusters in large data sets. Our work is intended to examine the appropriate sampling techniques as a data reduction scheme for clustering which require only a single data set scan. A good sample of the data set shall be a good substitute for the original data set while also keeping as much important cluster information as possible. Our experiments show that 2% of density-biased sampling (DBS) of the original data set can group clusters as well as clustering on the whole original data set and also help reduce time to cluster by over 95%. And the sampled data sets with density-biased reservoir sampling (DBRVS) technique report a noise tolerance property while the original data set is surrounded by many noises.

Introduction: Clustering in data mining is the process of discovering the clusters in a set of data, by maximizing the intra-cluster similarity and minimizing the inter-cluster similarity between clusters ([1], [3]). An efficient clustering method needs many difficult and complicated techniques to find the correct clusters from a very large data set. Clustering on large data sets is time and resource consuming ([3]). Many researchers have proposed sampling techniques to efficiently reduce the size of the data set, while keeping all important cluster information as much as possible. In this paper, we study three sampling techniques which can draw the samples by only a single scan over the data.

Random Sampling with a Reservoir (called RVS) ([1]) is the sampling technique which selects a random sample of size n from a data set of size N in a single data set pass within $O(n(1+\log N/n))$ expected time, where the size of the data set is unknown prior to sampling. The process is started by initializing first n objects to the reservoir output buffer of size n. And then, randomly select a number of objects, k, to be skipped in the main data set to select a new sample object. When the object is selected, it becomes a candidate and randomly replaces one object in the reservoir buffer. This step is repeated until the end of file has been reached. Finally, all objects in the reservoir buffer of size n become a final sampled data set.

Density Biased Sampling (called DBS) ([2]) is the sampling technique that is proposed to take into account the sampling over a data set which follows the Zipf's distribution. The sampling process is started by partitioning data into groups using a hashing function. And the biasing process is to draw an object by considering the reverse density of its group. It probabilistically over-samples sparse regions and under-samples dense regions. With this biased sampling, small clusters will not be missed.

Density Biased Reservoir Sampling (called DBRVS) ([3]) is the adapted sampling technique which combines the density biased scheme together with the reservoir scheme. The sampling process is started by partitioning the data space into a finite number of equiwidth bins in the quantized space. Then apply random sampling with a reservoir scheme to the series of binning groups. The biased reservoir sampling draws a group by the consideration of two consecutive binning groups. The denser group is a candidate to be included in the sample if the density difference of two groups is above some threshold δ or the sum of the density on both groups is above the threshold ε.

Methodology: In our experiments, we studied both efficiency and effectiveness of each sampling technique with some synthetic data sets. We monitored sampling time and memory usage on various sizes of samples. Then we determined the clusters for the sampled data sets using k-means (the partitioning clustering algorithm) and monitor time to discover clusters. After finishing the clustering process, we compare all found clusters with the original clusters and calculate the value of *Number of Clusters found* (NC), which is defined in [2].

Results, Discussion and Conclusion: From the experimental results, we found that DBS is the most accurate sampling technique. It shows that a 2% DBS sample of the original data set can produce the same result as the whole original data set as show in Figure 3, and also help reduce time to find the clusters by over 95%. On the other hand, it requires more memory and takes longer time of sampling process than the others (Figures 1 and 2). RVS is the most resource saving technique. It consumes a small amount of memory and runs faster than the others, but its NC tends to decrease after the sample size has reached some value (Figure 3).

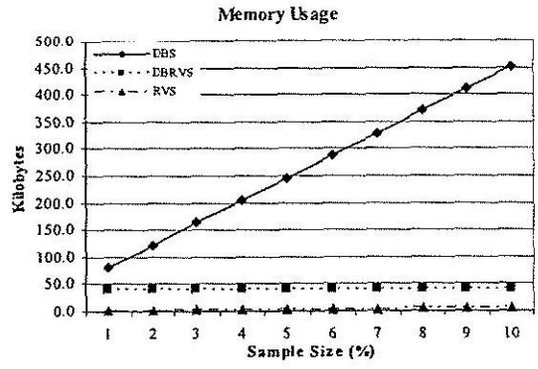


Figure 1: Memory usage (kb) for each sampling technique.

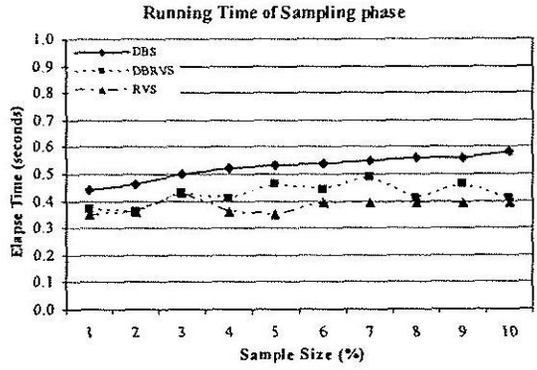


Figure 2: Running time of sampling phase.

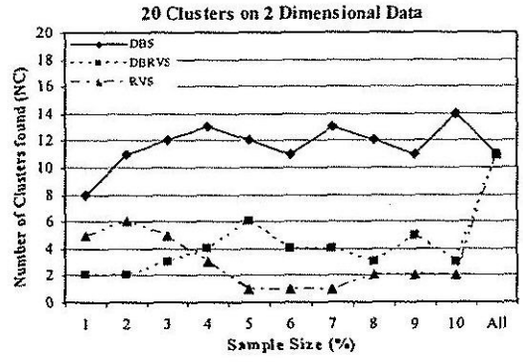


Figure 3: Number of clusters found on sampled data sets.

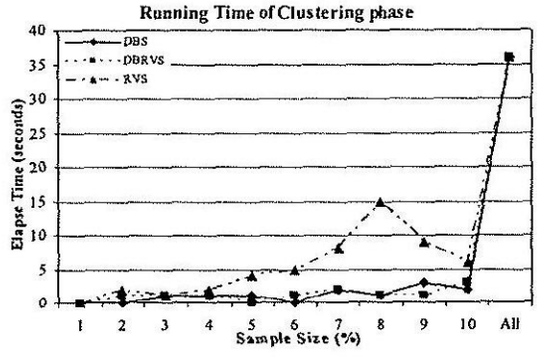


Figure 4: Running time of clustering phase.

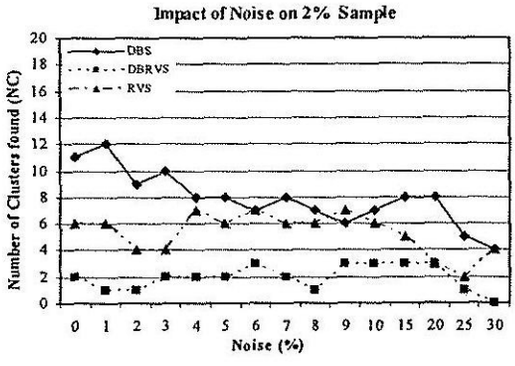


Figure 5: The Impact of noise on 2% samples.

Figure 5 show that DBRVS has a noise tolerance property. It has the least impact when many noises occurred although its NC is less than satisfactory, while DBS is very sensitive to noises. Our future research is to extend our study and to design such a sampling technique that computes densities accurately and efficiently and also is less sensitive to noise.

Acknowledgement: We would like to thank C. Palmer for putting his density biased sampling source code on the Web. And thank J. Handl for sharing his Cluster Generator and some synthetic data sets. This research has been supported by the Data Engineering and Knowledge Discovery Research Unit (DEKD), which is fully supported by the Suranaree University of Technology.

References: [1] J. S. Vitter, "Random Sampling with a Reservoir," *ACM Transactions on Mathematical Software*, vol. 11, pp. 37-57, 1985.

[2] C. R. Palmer and C. Faloutsos, "Density Biased Sampling: An Improved Method for Data Mining and Clustering," *ACM-SIGMOD Int. Conf. on Management of Data*, pp. 82-92, 2000.

[3] K. Kerdprasop, N. Kerdprasop, and J. Sun, "Density Biased Reservoir Sampling for Clustering," *Proc. of IASTED Int. Conf. on Artificial Intelligence and Applications*, pp. 95-100, 2005.

Keywords: Data mining, Data reduction, Sampling, Clustering, Density biased, Reservoir

Research Network Development of Higher Education Alliance in Nakhon Ratchasima,
Suranaree University of Technology, 24 June 2005, pages 66-67

ชื่องานวิจัย : **เทคนิคการลดขนาดข้อมูลเพื่องานจัดกลุ่มข้อมูลขนาดใหญ่**

คณะผู้วิจัย : ผู้ช่วยศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ,

ผู้ช่วยศาสตราจารย์ ดร.นิตยา เกิดประสพ และ นายธรรมศักดิ์ เธียรนิเวศน์

ผู้นำเสนอผลงานวิจัย : นายธรรมศักดิ์ เธียรนิเวศน์

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>

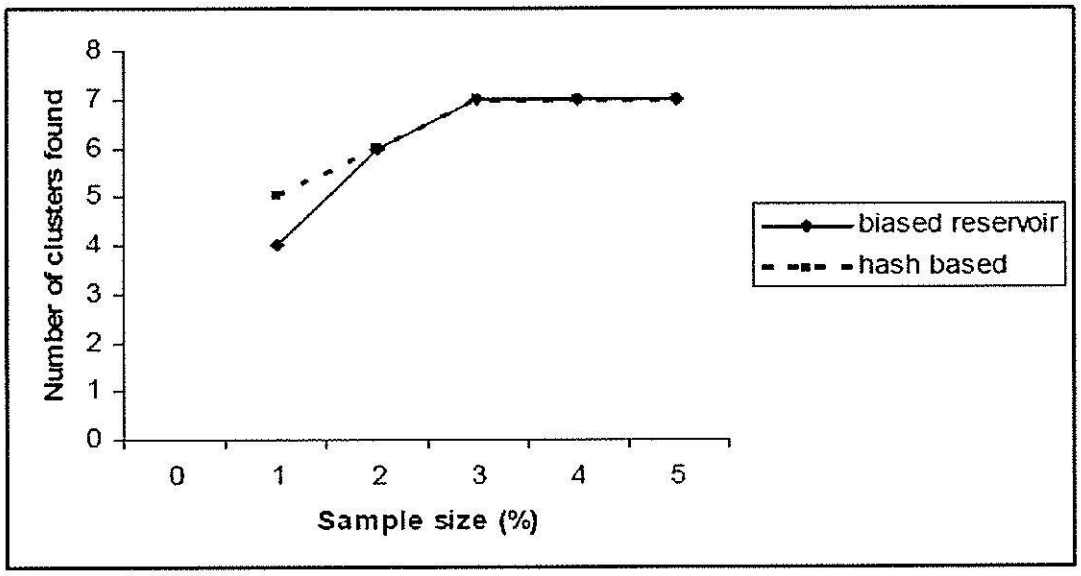
การจัดกลุ่มข้อมูลโดยอัตโนมัติใช้เทคนิคการวัดความคล้ายคลึงกันของข้อมูลแต่ละคู่ เพื่อรวมกลุ่มข้อมูลที่คล้ายกันเข้าด้วยกัน อัลกอริทึมในการจัดกลุ่มข้อมูลโดยอัตโนมัติมีหลากหลายอัลกอริทึม เวลาที่ใช้ของแต่ละอัลกอริทึมจะแปรผันไปตั้งแต่ $O(knl)$ ไปจนถึง $O(n^2 \log n)$ เมื่อ n คือจำนวนข้อมูล, k คือจำนวนกลุ่มของข้อมูล และ l คือ จำนวนรอบของการทำงานซ้ำ เนื้อที่หน่วยความจำที่ใช้จะแปรผันไปตามแต่ละอัลกอริทึม โดยทั่วไปจะเป็นตั้งแต่ $O(k)$ ไปจนถึง $O(n^2)$ ดังนั้นยิ่งจำนวนข้อมูลมีมากการจัดกลุ่มข้อมูลโดยอัตโนมัติจะยิ่งเสียเวลาและเสียเนื้อที่หน่วยความจำมาก เมื่อปริมาณข้อมูลมากถึงระดับเนื้อที่หน่วยความจำไม่เพียงพอต่อการใช้งาน วิธีการแก้ปัญหาที่ได้ผลคือลดขนาดของข้อมูลก่อนที่จะจัดกลุ่มข้อมูล งานวิจัยนี้จึงมีวัตถุประสงค์ที่จะศึกษาเปรียบเทียบเทคนิคต่างๆ ที่ใช้ในการลดขนาดข้อมูลได้อย่างมีประสิทธิภาพและเหมาะสมกับงานจัดกลุ่มข้อมูล

<u>วิธีการ :</u>

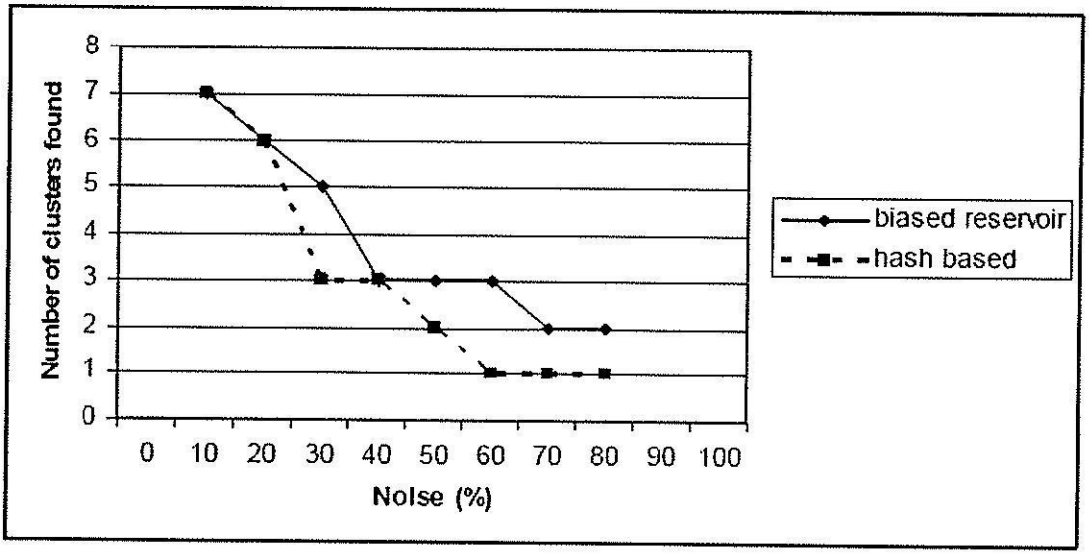
เทคนิคการลดขนาดข้อมูลที่ใช้ในการศึกษาทดลองนี้เน้นที่เทคนิคการสุ่ม โดยใช้การสุ่มในหลายลักษณะ ได้แก่ random sampling, biased sampling, reservoir sampling ข้อมูลสุ่มที่ได้จะถูกนำไปทดสอบการจัดกลุ่มด้วยโปรแกรม k-means และ EM (Expectation-Maximization) เพื่อทดสอบเปรียบเทียบประสิทธิภาพ ผลการทดสอบจะนำไปสู่การออกแบบเทคนิคที่เหมาะสมในการลดขนาดข้อมูลสำหรับงานจัดกลุ่มข้อมูลอัตโนมัติ

<u>ผลที่ได้ :</u>

ผลการทดสอบเปรียบเทียบประสิทธิภาพการลดขนาดข้อมูลด้วยเทคนิคการสุ่มแบบ reservoir ที่มีการ bias ตามความหนาแน่นของข้อมูลและเทคนิคการลดขนาดข้อมูลด้วยการใช้ตารางแฮช แสดงได้ดังรูปที่ 1 และ 2



รูปที่ 1 ผลการเปรียบเทียบประสิทธิภาพการจัดกลุ่มข้อมูลกับชุดข้อมูลที่ลดขนาดด้วยวิธีการสุ่มแบบ reservoir และวิธีการแบบแฮช



รูปที่ 2 เปรียบเทียบประสิทธิภาพการจัดกลุ่มข้อมูลกับชุดข้อมูลที่มีข้อมูลรบกวน

<u>สรุปผลการทดลอง :</u>

ปัจจุบันมีผู้เสนอเทคนิคการลดขนาดข้อมูลด้วยหลากหลายเทคนิค งานวิจัยนี้มีวัตถุประสงค์ที่จะค้นหาเทคนิคที่ทำงานได้ดีกับข้อมูลที่มีการกระจายหลายลักษณะเพื่อเพิ่มความถูกต้องและความเร็วในการทำงานของโปรแกรมจัดกลุ่มข้อมูลอัตโนมัติ ผลการศึกษานี้จะเป็นแนวทางในการพัฒนาโปรแกรมจัดกลุ่มข้อมูลขนาดใหญ่ได้ในอนาคต

ประวัติผู้วิจัย

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา ประเทศสหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของทบวงมหาวิทยาลัย (หรือสำนักงานคณะกรรมการการอุดมศึกษาในปัจจุบัน) โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "Active database rule set reduction by knowledge discovery" หลังสำเร็จการศึกษาได้ปฏิบัติงานในตำแหน่งอาจารย์ ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ปัจจุบันดำรงตำแหน่งหัวหน้าหน่วยปฏิบัติการวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ (Data Engineering and Knowledge Discovery Research Unit – DEKD) สำนักวิชาวิศวกรรมศาสตร์ ดำเนินการวิจัยประยุกต์เกี่ยวกับการออกแบบและพัฒนาระบบเหมืองข้อมูลประสิทธิภาพสูงที่สามารถทนต่อข้อมูลรบกวน และการวิจัยพื้นฐานเกี่ยวกับเทคนิคการจัดกลุ่มข้อมูล และเทคนิคการวิเคราะห์ข้อมูลขนาดใหญ่ด้วยฮิวริสติก โดยมีผลงานวิจัยตีพิมพ์ในวารสารวิชาการและเอกสารการประชุมวิชาการ จำนวนมากกว่า 30 เรื่อง ในสาขาระบบฐานความรู้ ฐานข้อมูลแอคทีฟ ฐานข้อมูลนิรนัย การทำเหมืองข้อมูลและการค้นหาความรู้