การพัฒนาเครื่องมือวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย โดยนำวิธีการของเจเนติกมาใช้ร่วมกับดิสชั่นทรี

Developing Analysis Tool for Cost-Sensitive Classification by Using Hybrid Genetic Decision Tree

โดย

นางสาวแววดาว อดุลย์พิจิตร

รหัส 42067008

อาจารย์ที่ปรึกษา

รศ. ดร. วิเชียร เปรมชัยสวัสดิ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 2 ปีการศึกษา 2543
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	**การพัฒนาเครื่องมือวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่ายโดยนำวิธีการของเจเนติกมาใช้ร่วมกับดิสิชั่นทรี**
นักศึกษา	**นางสาวแววดาว อดุลย์พิจิตร**
อาจารย์ที่ปรึกษา	**รศ. ดร. วิเชียร เปรมชัยสวัสดิ์**
ระดับการศึกษา	**วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ**
แขนงวิชา	**วิทยาการสารสนเทศ**
ปีการศึกษา	2543

บทคัดย่อ

โครงการนี้ได้ทำการศึกษาและพัฒนาโปรแกรมคอมพิวเตอร์ที่ช่วยในการวิเคราะห์จัดประเภทข้อมูล โดยการนำเอาวิธีการทางคาต้าไมนิ่งมาใช้ในการวิเคราะห์ข้อมูล อัลกอริทึมที่ใช้ในการพัฒนานี้คืออัลกอริทึม ICET ซึ่งเป็นอัลกอริทึมในการจัดประเภทข้อมูลที่คำนึงถึงทั้งค่าใช้จ่ายจากการทดสอบ และ ค่าใช้จ่ายจากการจัดประเภทที่ผิดพลาด โดยอัลกอริทึมนี้เป็นการนำวิธีการทางคาต้าไมนิ่งสองวิธีมาประยุกต์ใช้ร่วมกัน ได้แก่ ดิสิชั่นทรีและเจเนติกอัลกอริทึม โดยดิสิชั่นทรีนั้นใช้ในการจัดประเภทของข้อมูล และเจเนติกนั้นช่วยในการหาจุดที่ดีหรือเหมาะสมที่สุด

อัลกอริทึม ICET นำเจเนติกมาใช้เพื่อช่วยปรับปรุงการทำงานของดิสิชั่นทรีเพื่อให้ได้ผลที่ความถูกต้อง และมีค่าใช้จ่ายน้อยที่สุด และในโครงงานนี้ก็ได้นำอัลกอริทึมดังกล่าวมาพัฒนาเป็นโปรแกรมคอมพิวเตอร์ที่ช่วยในการวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย ซึ่งจะเป็นประโยชน์สำหรับการวิเคราะห์ข้อมูลในชีวิตประจำวันได้

Title	Developing Analysis Tool for Cost-Sensitive Classification by Using Hybrid Genetic Decision Tree
Student	Miss Waewdao Adulpichit
Advisor	Assoc. Prof. Dr. Wichian Premchaisawasdi
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

ABSTRACT

In this development project, we have studied on the development of an analysis tool for data classification, which use data mining to analyze the data. Algorithm used in this project is ICET , which is a cost-sensitive classification algorithm. This algorithm applies two algorithms of data mining; that is decision trees and genetic algorithm. Decision tree is used for classification and genetic is used for finding the optimum point.

ICET algorithm use genetic algorithm to improve classification of decision trees for the more accurate results and lower costs. This project develops ICET algorithm into an analysis tool for cost-sensitive data classification. This tool will be useful for analyzing data in real life.

กิตติกรรมประกาศ

โครงการการพัฒนาเครื่องมือวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่ายโดยนำวิธีการของ
จเนติกมาใช้ร่วมกับดิสิชั่นทรีฉบับนี้ ผู้เขียนขอขอบพระคุณ รศ. ดร. วิเชียร เปรมชัยสวัสดิ์ อาจารย์
ที่ปรึกษาที่ได้กรุณาให้คำปรึกษาและแนะนำ และขอขอบคุณผู้ที่เกี่ยวข้องทุก ๆ ท่านที่ได้กรุณาให้
คำปรึกษาและแนะนำวิธีการแก้ปัญหาต่าง ๆ ให้สามารถแก้ไขปัญหาให้ผ่านพ้นลุล่วงไปได้ ทำให้
โครงการพัฒนาระบบงานนี้สำเร็จลง

แววดาว อดุลย์พิจิตร

กุมภาพันธ์ 2544

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

หน้า

ตารางที่

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

บทที่ 1

บทนำ

1.1 ความเป็นมา

การทำดาต้าไมนิ่ง เป็นเรื่องหนึ่งที่กำลังได้รับความสนใจในปัจจุบันนี้ เพราะช่วยให้เราได้ใช้ประโยชน์จากข้อมูลที่เรามีอยู่ได้อย่างคุ้มค่า ซึ่งทั้งเจเนติกและดิสิชั่นทรีก็เป็นหนึ่งในหลาย ๆ วิธีที่ใช้ในการทำดาต้าไมนิ่ง โดยดิสิชั่นทรีนั้นใช้ในการจัดประเภทของข้อมูล และเจเนติกนั้นช่วยในการหาจุดที่ดีหรือเหมาะสมที่สุด

ในการทำ decision tree เพื่อทำการจัดประเภทข้อมูลนั้น เราควรคำนึงถึง ทั้งค่าใช้จ่ายจากการทดสอบ (test costs) และค่าใช้จ่ายจากการจัดประเภทที่ผิดถ้าไม่มีการทดสอบ (classification costs) เนื่องจากเราอาจไม่สามารถตัดสินอย่างมีเหตุผลได้ว่าควรจะทำการทดสอบหรือไม่ ถ้าเราไม่รู้ค่าใช้จ่ายของการจัดประเภทที่ถูกต้องและที่ผิดพลาด เราจึงต้องหาจุดสมดุลระหว่างค่าใช้จ่ายของการทดสอบ กับ ประโยชน์ของการทดสอบที่จะมีส่วนช่วยให้การจัดประเภทมีความถูกต้อง นอกจากนั้นเราจะต้องพิจารณาว่า เมื่อไรที่การทำการทดสอบต่อไปนั้นจะไม่เหมาะสมในแง่เศรษฐกิจ เนื่องจากเป็นไปได้ที่ประโยชน์ที่ได้จากการทดสอบ อาจไม่คุ้มค่ากับค่าใช้จ่ายที่ต้องเสียไปจากการทดสอบนั้น

อัลกอริทึมที่จะนำมาใช้ในการพัฒนาโปรแกรมนี้เป็นอัลกอริทึมเพื่อจัดประเภทที่คำนึงถึงค่าใช้จ่าย (cost-sensitive classification) มีชื่อว่า ICET (Inexpensive Classification with Expensive Test) ซึ่งมีคุณสมบัติหลัก ๆ คือ คำนึงถึงค่าใช้จ่ายของการทดสอบ และคำนึงถึงค่าใช้จ่ายของการแบ่งประเภทที่ผิดพลาด โดยใช้การค้นหาแบบ greedy heuristic ร่วมกับวิธีการของเจเนติก ซึ่งอัลกอริทึมนี้สามารถจัดการกับค่าใช้จ่ายที่เป็นเงื่อนไข (เมื่อค่าใช้จ่ายของการทดสอบหนึ่งขึ้นกับว่ามีการเลือกทำอีกการทดสอบหนึ่งไปหรือยัง) และมีการแยกการทดสอบออกเป็นสองแบบ คือ แบบที่ให้ผลทันที (immediate) และแบบที่ให้ผลล่าช้า (delayed)

โปรแกรมนี้ที่พัฒนาขึ้นในโครงงานนี้จะพัฒนาโดยอาศัยการเขียนโปรแกรมด้วยวิชวลเบสิก ความรู้ทางด้านวิธีการของดาต้าไมนิ่ง ทั้งความรู้ทางการจัดประเภทข้อมูลโดยใช้ดิสิชั่นทรี และความรู้ในการหาจุดที่เหมาะสมที่สุดโดยใช้วิธีการของเจเนติก และนำทั้งสองวิธีการมาประยุกต์ใช้ร่วมกันในการพัฒนาโปรแกรม

times_pregnant:	1.00$
glucose_tol:	17.61$
diastolic_pb:	1.00$
triceps:	1.00$
insulin:	22.78$
mass_index:	1.00$
pedigree:	1.00$
age:	1.00$

รูปที่ 1.2 ตัวอย่าง cost file

1.5 ขั้นตอนการดำเนินงาน

1. ทำการศึกษาถึงวิธีการของดิสิชั่นทรีและเจเนติก ที่จะนำมาใช้ในการพัฒนาโปรแกรม
2. ทำการศึกษาอัลกอริทึมที่นำทั้งสองวิธีการมาประยุกต์ใช้ร่วมกัน
3. นำอัลกอริทึมดังกล่าวมาพัฒนาเป็นโปรแกรมเครื่องมือวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย
4. ทำการตรวจสอบและปรับปรุงส่วนต่าง ๆ ของโปรแกรมให้มีความถูกต้องและง่ายต่อการใช้งานยิ่งขึ้น
5. จัดทำเอกสารประกอบโครงงาน

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. เพิ่มประสบการณ์ความรู้ความเข้าใจในการประยุกต์ใช้วิธีการทางคาต้าไมนิ่ง และการนำแต่ละวิธีการมาใช้งานร่วมกัน
2. เป็นแนวทางในการออกแบบและพัฒนาโปรแกรมวิเคราะห์ข้อมูลโดยใช้วิธีการอื่น ๆ ต่อไป

1.2 เป้าหมายของโครงงาน

1. เพื่อพัฒนาโปรแกรมคอมพิวเตอร์ที่นำเอาวิธีการทางคาต้าไมนิ่งมาใช้ในการวิเคราะห์ข้อมูล
2. เพื่อเพิ่มความเข้าใจในการประยุกต์ใช้วิธีการเจเนติกร่วมกับวิธีการดิสิชั่นทรี
3. เพื่อเป็นแนวทางในการออกแบบและพัฒนาโปรแกรมวิเคราะห์ข้อมูลโดยใช้วิธีการอื่น ๆ ต่อไป

1.3 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษารูปแบบและวิธีการนำเอาเจเนติกและดิสิชั่นทรีมาประยุกต์ใช้ร่วมกันเพื่อสร้างเป็นโปรแกรมเครื่องมือวิเคราะห์จัดประเภทข้อมูล
2. เพื่อศึกษาข้อมูลที่ได้จาก
3. เพื่อออกแบบโปรแกรมช่วยวิเคราะห์ข้อมูลที่เกี่ยวข้องกับการจัดประเภท และค่าใช้จ่าย

1.4 ขอบเขตของการศึกษา

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมช่วยวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย ซึ่งต้องทำการศึกษาความรู้เกี่ยวกับวิธีการทางคาต้าไมนิ่งได้แก่ วิธีการของดิสิชั่นทรีที่ใช้ในการจัดประเภทข้อมูล และวิธีการของเจเนติกที่ใช้ในการหาจุดที่เหมาะสม ซึ่งในที่นี้คือการหาจุดที่ค่าใช้จ่ายน้อยที่สุด และทำให้การจัดประเภทมีความถูกต้องด้วย โดยข้อมูลที่จะนำมาใช้ในการวิเคราะห์นี้ที่จำเป็นประกอบด้วยตัว data file ที่ประกอบด้วย attribute ต่าง ๆ และ ประเภทของข้อมูล (class) และข้อมูลแสดงค่าใช้จ่ายในการทดสอบแต่ละ attribute (cost file)

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31.0,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
0,118,84,47,230,45.8,0.551,31,1
```

รูปที่ 1.1 ตัวอย่าง data file

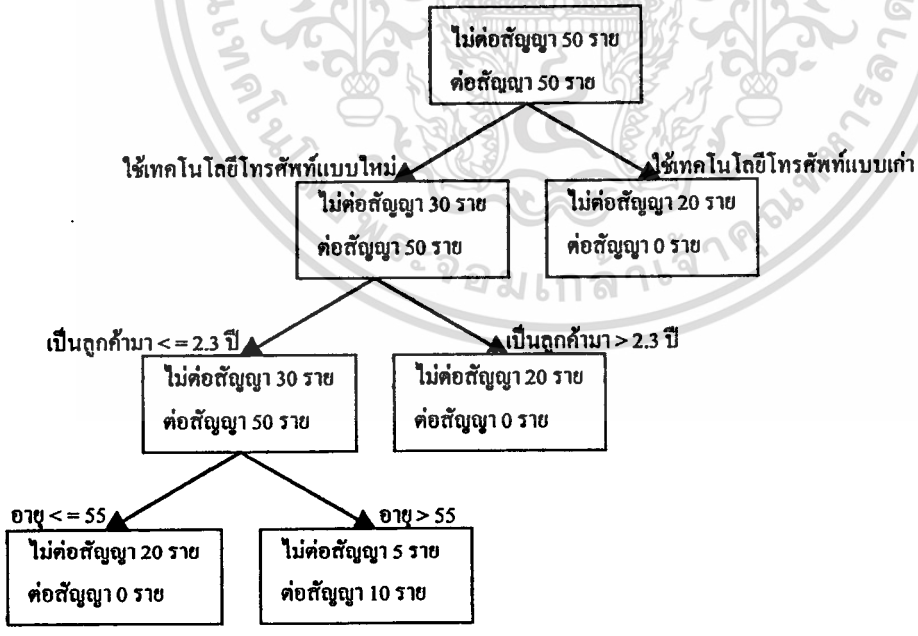
บทที่ 2

ความรู้เบื้องต้นเกี่ยวกับวิธีการทางดาต้าไมนิ่ง

วิธีการทางดาต้าไมนิ่งมีอยู่หลายวิธี ซึ่งจะนำวิธีใดมาใช้ก็ขึ้นอยู่กับจุดประสงค์ของงานแต่ละงาน ในที่นี้จะกล่าวถึงวิธีการทางดาต้าไมนิ่งที่ได้นำมาใช้ในงานนี้สองวิธีคือ วิธีของดิสิชั่นทรีและเจเนติก

2.1 วิธีการดิสิชั่นทรี (Decision Trees)

ดิสิชั่นทรีเป็นโครงสร้างที่สามารถมองได้ในรูปแผนภูมิต้นไม้ โดยแต่ละกิ่งของต้นไม้ก็คือการทดสอบเพื่อการจัดประเภท (classification) และที่ปลาย (leaves) ของต้นไม้ก็คือกลุ่มของข้อมูลที่ถูกแบ่งตามประเภทของมัน ตัวอย่างเช่น ถ้าต้องการจัดประเภทของลูกค้าที่จะไม่ต่อสัญญาในธุรกิจโทรศัพท์เคลื่อนที่ ก็อาจได้ดิสิชั่นทรีที่มีลักษณะดังรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างโครงสร้างของดิสิชั่นทรี

อัลกอริทึมที่ใช้ในการสร้างดิสิชั่นทรีมีอยู่หลายอัลกอริทึมด้วยกัน อาทิเช่น ID3, C4.5 และ CART เป็นต้น สำหรับอัลกอริทึมในการสร้างดิสิชั่นทรีที่ใช้ในโครงการนี้มีพื้นฐานตามอัลกอริทึม C4.5 ของ J. Ross Quinlan ซึ่งมีวิธีการทำงานดังนี้

2.1.1 หลักการ Divide and conquer

ถ้ากำหนดให้ประเภทของข้อมูล(class) เขียนแทนด้วย $\{C_1, C_2, ..., C_k\}$ และให้ T คือ กลุ่มของข้อมูลสำหรับการฝึกสอน (training case) แล้ว จะมีความเป็นไปได้ 3 ทาง คือ

- T มี ≥ 1 case และทุก case อยู่ใน class C_j

  ดิสิชั่นทรีของ T คือ leaf node ที่ระบุว่าเป็น class C_j

- T ไม่มี case ใดอยู่เลย

  ดิสิชั่นทรีของ T คือ leaf node แต่ class ที่สัมพันธ์กับ leaf node นี้ ต้องใช้ข้อมูลอื่นนอกเหนือจาก T ในการระบุประเภท เช่น ใช้ class หลักของกลุ่มข้อมูลนั้น (class ที่กลุ่มข้อมูลส่วนใหญ่ถูกจัดอยู่)

- T มี case ที่ตกอยู่ในหลาย ๆ class

  ต้องแบ่ง T ออกเป็น subset ของ case ที่แต่ละกลุ่มอยู่ (หรือมีแนวโน้มจะอยู่) ใน class เดียวกัน เช่น ถ้า *testT10* ให้ผลเป็น mutually exclusive $\{O_1, O_2, ..., O_n\}$ แล้ว T จะถูกแบ่งเป็น subset T_1, T_2, ..., T_n โดย T_i จะบรรจุทุก case ใน T ที่ให้ผลเป็น O_i ดังนั้น ดิสิชั่นทรีของ T จะประกอบด้วย decision node ที่แต่ละ node ระบุ test และแต่ละกิ่งจะแทนผลที่เป็นไปได้ กลไกการสร้าง tree จะดำเนินไปแบบ recursive เช่นนี้กับแต่ละกลุ่มย่อย (subset) ของ training case โดยกิ่งที่ i จะต่อไปที่ decision tree ที่สร้างจาก subset T_i ของ training case

2.1.2 การวัดค่าของแอททริบิวต์หนึ่ง ๆ (Gain Criterion)

ให้ T เป็นกลุ่มของข้อมูล และให้ $freq(C_i, T)$ เป็นจำนวน case ที่อยู่ในกลุ่ม T และจัดอยู่ในประเภท C_i และ $|T|$ เป็นจำนวนของ case ที่อยู่ในกลุ่ม T

ทฤษฎีเกี่ยวกับ information กล่าวไว้ว่า: information ที่ message หนึ่งมีสามารถวัดได้เป็น bit เท่ากับ ลบลอกการิทึมฐาน 2 ของความน่าจะเป็นของ message นั้น เช่น ถ้ามี 8 messages ที่มีสัดส่วนเท่ากัน ค่า information ของแต่ละ message จะเท่ากับ $-\log_2(1/8)$ เท่ากับ 3 bits

ดังนั้น ถ้ามี case หนึ่งในกลุ่มข้อมูลสำหรับการฝึกสอน T จัดอยู่ในประเภท C_j ดังนั้นความน่าจะเป็นของ message นี้เท่ากับ

$$\frac{freq(C_j, T)}{|T|}$$

และ information ของ message นี้เท่ากับ

$$-\log_2 \left(\frac{freq(C_j, T)}{|T|} \right) \text{ bits}$$

ดังนั้นค่า information ของกลุ่มข้อมูล T คือ

$$info(T) = - \sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} \times \log_2 \left(\frac{freq(C_j, T)}{|T|} \right) \text{bits}$$

เมื่อกลุ่มข้อมูล T ถูกแบ่งตามแอททริบิวต์ X ที่มีค่า n ค่า

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

ดังนั้น information ที่จะได้จากการแบ่งส่วน T ออกตามค่าของแอททริบิวต์ X จะเท่ากับ

$$Gain(X) = info(T) - info_x(T)$$

ซึ่งในการเลือกว่าจะนำแอททริบิวต์ใดมาใช้ในการแตกกิ่งของทรีลงไป ก็จะเลือกแอททริบิวต์ที่ให้ค่า gain สูงสุด

ตัวอย่างเพื่อให้เห็นวิธีการทำงาน เช่น ให้ข้อมูลการจัดประเภทว่าควรเล่นเทนนิสหรือไม่ตามข้อมูลที่มี 4 แอททริบิวต์ ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่างข้อมูลที่ใช้ในการสร้างดิสิชั่นทรี

สภาพอากาศ	อุณหภูมิ($^{\circ}$F)	ความชื้น (%)	ลมแรง?	ประเภท
มีแดด	75	75	จริง	เล่นได้
มีแดด	80	90	จริง	ไม่ควรเล่น
มีแดด	85	85	เท็จ	ไม่ควรเล่น
มีแดด	72	95	เท็จ	ไม่ควรเล่น
มีแดด	69	70	เท็จ	เล่นได้
มีเมฆมาก	72	90	จริง	เล่นได้
มีเมฆมาก	83	78	เท็จ	เล่นได้
มีเมฆมาก	64	65	จริง	เล่นได้
มีเมฆมาก	81	75	เท็จ	เล่นได้
ฝนตก	71	80	จริง	ไม่ควรเล่น
ฝนตก	65	70	จริง	ไม่ควรเล่น

ตารางที่ 2.1(ต่อ) ตัวอย่างข้อมูลที่ใช้ในการสร้างดิสชั่นทรี

สภาพอากาศ	อุณหภูมิ(°F)	ความชื้น (%)	ลมแรง?	ประเภท
ฝนตก	75	80	เท็จ	เล่นได้
ฝนตก	68	80	เท็จ	เล่นได้
ฝนตก	70	96	เท็จ	เล่นได้

จากตัวอย่างข้อมูลเพื่อการตัดสินใจ เล่น หรือ ไม่เล่น เทนนิส ข้างต้น

$$info(T) = -9/14 \times \log_2(9/14) - 5/14 \times \log_2(5/14) = 0.940 \text{ bits}$$

และถ้าใช้แอททริบิวต์ *สภาพอากาศ* ในการแบ่งข้อมูล T ออกเป็น 3 ส่วนตามค่าของแอททริบิวต์

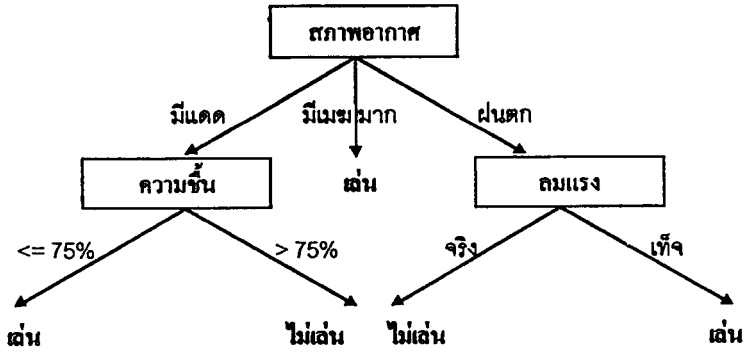
$$info_X(T) = 5/14 \times (-2/5 \times \log_2(2/5) - 3/5 \times \log_2(3/5))$$
$$+ 4/14 \times (-4/4 \times \log_2(4/4) - 0/4 \times \log_2(0/4))$$
$$+ 5/14 \times (-3/5 \times \log_2(3/5) - 2/5 \times \log_2(2/5))$$
$$= 0.694 \text{ bits}$$

ดังนั้นค่า information gain ของการทดสอบนี้เท่ากับ 0.940 - 0.694 = 0.246

แต่ถ้าเราเลือกแอททริบิวต์ *ลมแรง* มาแทน

$$info_X(T) = 6/14 \times (-3/6 \times \log_2(3/6) - 3/6 \times \log_2(3/6))$$
$$+ 814 \times (-6/8 \times \log_2(6/8) - 2/8 \times \log_2(2/8))$$
$$= 0.892 \text{ bits}$$

ดังนั้นจะมีค่า gain เท่ากับ 0.940 - 0.892 = 0.048 ซึ่งน้อยกว่าค่า gain ที่ได้จากการใช้แอททริบิวต์ *สภาพอากาศ* ดังนั้นจึงควรเลือกแอททริบิวต์ *สภาพอากาศ* มาใช้ในการแตกกิ่งของทรี มากกว่าใช้แอททริบิวต์ *ลมแรง* ซึ่งจากหลักการดังกล่าวสามารถสร้างดิสชั่นทรีจากข้อมูลข้างต้นได้ ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ดิสชั่นทรีที่ได้จากข้อมูลในตารางที่ 2.1

2.1.3 การทดสอบแอททริบิวต์ที่มีค่าเป็นเลขต่อเนื่อง (continuous)

สำหรับแอททริบิวต์ที่มีค่าเป็นเลขต่อเนื่องนั้น จะต้องมีการหาค่า threshold ว่าจะใช้ตัวเลขใดเป็นตัวแบ่งในการทดสอบแอททริบิวต์นั้น ถ้าให้ T เป็นกลุ่มข้อมูลสำหรับการฝึกสอน และ A เป็นแอททริบิวต์ที่มีค่าเป็นเลขต่อเนื่องที่ต้องการจะหาค่า threshold ประการแรกจะต้องนำกลุ่มข้อมูล T มาเรียงลำดับตามค่าของแอททริบิวต์ A ก่อน โดยค่าของแอททริบิวต์ A จะถูกเรียงอยู่ในช่วง $\{V_1, V_2, ..., V_m\}$ ดังนั้นค่า threshold ที่ต้องการหาจะอยู่ระหว่าง V_i ถึง V_{i+1} ซึ่งจะแบ่งข้อมูลออกเป็นกลุ่มข้อมูลที่ค่าของแอททริบิวต์ A อยู่ในช่วง $\{V_1, V_2, ..., V_i\}$ และกลุ่มข้อมูลที่แอททริบิวต์ A มีค่าอยู่ในช่วง $\{V_{i+1}, V_{i+2}, ..., V_m\}$ ค่า threshold ที่เป็นไปได้มี m-1 ค่า แล้วจึงนำทั้ง m-1 ค่านี้มาคำนวณหาค่า Gain ตามสมการที่ได้กล่าวไปแล้ว ตัวเลขใดให้ค่า Gain มากที่สุดก็ใช้เป็นค่า threshold ต่อไป

แต่จากหลักการหาค่า threshold ของแอททริบิวต์ที่มีค่าเป็นเลขต่อเนื่องนี้ ทำให้มันมีโอกาสที่จะถูกเลือกไปใช้ในการแตกกิ่งของทรีมากกว่าแอททริบิวต์ที่มีค่าเฉพาะ (discrete attribute) โดยเฉพาะถ้าแอททริบิวต์แบบต่อเนื่องนั้นมีค่าที่แตกต่างกันหลายค่า ก็จะยิ่งมีโอกาสมากกว่าได้ทดลองหาตัวที่จะแบ่งข้อมูลได้ดีที่สุด ทำให้ค่า gain ที่ได้มาอาจไม่เป็นธรรมนัก จึงมีการปรับปรุงการหาค่า gain ของแอททริบิวต์ที่เป็นแบบต่อเนื่องเป็น

$$\text{Gain}(X) = \text{Gain}(X) - \frac{\log_2(N-1)}{|D|}$$

โดยที่ N คือจำนวนค่าที่แตกต่างกันของแอททริบิวต์ X และ |D| คือจำนวนของข้อมูล

2.1.4 การ Prune ดิสิชั่นทรี

การสร้างทรีโดยอาศัยหลักการทำงานแบบ Divide and Conquer จะสิ้นสุดการทำงานเมื่อกลุ่มข้อมูลย่อยแต่ละกลุ่มตกอยู่ใน class ใด class หนึ่งทั้งกลุ่ม แต่ดิสิชั่นทรีที่ได้มานี้ก็อาจจะขึ้นกับกลุ่มข้อมูลที่ใช้ในการฝึกสอนมากเกินไปได้ ดังนั้นจึงต้องมีการ Prune ทรี

การ Prune ก็เปรียบเสมือนกับการตัดแต่งกิ่งต้นไม้ไม่ให้มีกิ่งก้านมากเกินไป สำหรับการทำงานของดิสิชั่นทรีแล้ว การ Prune คือการตัดบางกิ่งของทรีออกแล้วแทนกิ่งนั้นด้วย leaf node โดยหลักการในการตัดก็จะดูที่ค่าความผิดพลาด (error) ที่จะเกิดขึ้นกับการตัดส่วนนั้นออก ถ้าไม่มากเกินจุดที่กำหนดก็ตัดออกได้ รูปที่ 2.3 แสดงตัวอย่างการ prune ทรีที่ได้จากข้อมูลการออกเสียงในการประชุมสภาของสหรัฐอเมริกา

ดิสชั่นทรีก่อนทำการ Prune

ไม่เก็บค่ารักษาพยาบาล = ไม่เห็นด้วย
| รับการแก้ไขงบประมาณ = เห็นด้วย : พรรคเดโมแครต (151)
| รับการแก้ไขงบประมาณ = งดออกเสียง : พรรคเดโมแครต (1)
| รับการแก้ไขงบประมาณ = ไม่เห็นด้วย
| | ค่าใช้จ่ายในการศึกษา = ไม่เห็นด้วย : พรรคเดโมแครต (6)
| | ค่าใช้จ่ายในการศึกษา = เห็นด้วย : พรรคเดโมแครต (9)
| | ค่าใช้จ่ายในการศึกษา = งดออกเสียง : พรรครีพับลิกัน (1)

ไม่เก็บค่ารักษาพยาบาล = เห็นด้วย
| เลิกสัญญากับบริษัทเชื้อเพลิง = ไม่เห็นด้วย : พรรครีพับลิกัน (97/3)
| เลิกสัญญากับบริษัทเชื้อเพลิง = งดออกเสียง : พรรครีพับลิกัน (1)
| เลิกสัญญากับบริษัทเชื้อเพลิง = เห็นด้วย
| | ไม่เก็บภาษีสินค้าส่งออก = เห็นด้วย : พรรคเดโมแครต (2)
| | ไม่เก็บภาษีสินค้าส่งออก = งดออกเสียง : พรรครีพับลิกัน (1)
| | ไม่เก็บภาษีสินค้าส่งออก = เห็นด้วย
| | | ค่าใช้จ่ายในการศึกษา = ไม่เห็นด้วย : พรรคเดโมแครต (5/2)
| | | ค่าใช้จ่ายในการศึกษา = เห็นด้วย : พรรครีพับลิกัน (13/2)
| | | ค่าใช้จ่ายในการศึกษา = งดออกเสียง : พรรคเดโมแครต (1)

ไม่เก็บค่ารักษาพยาบาล = งดออกเสียง
| ร่วมออกค่าใช้จ่ายในโครงการน้ำ = ไม่เห็นด้วย : พรรคเดโมแครต (0)
| ร่วมออกค่าใช้จ่ายในโครงการน้ำ = เห็นด้วย : พรรคเดโมแครต (4)
| ร่วมออกค่าใช้จ่ายในโครงการน้ำ = งดออกเสียง
| | โครงการจรวดมิสไซล์ mx = ไม่เห็นด้วย : พรรครีพับลิกัน (0)
| | โครงการจรวดมิสไซล์ mx = เห็นด้วย : พรรคเดโมแครต (3/1)
| | โครงการจรวดมิสไซล์ mx = งดออกเสียง : พรรครีพับลิกัน (2)

หลังทำการ Prune

ไม่เก็บค่ารักษาพยาบาล = ไม่เห็นด้วย : พรรครีพับลิกัน (168/2.6)
ไม่เก็บค่ารักษาพยาบาล = เห็นด้วย : พรรคเดโมแครต (123/13.9)
ไม่เก็บค่ารักษาพยาบาล = งดออกเสียง
| โครงการจรวดมิสไซล์ mx = ไม่เห็นด้วย : พรรคเดโมแครต (3/1.1)
| โครงการจรวดมิสไซล์ mx = เห็นด้วย : พรรคเดโมแครต (4/2.2)
| โครงการจรวดมิสไซล์ mx = งดออกเสียง : พรรครีพับลิกัน (2/1)

รูปที่ 2.3 ตัวอย่างดิสชั่นทรีก่อนและหลังการ Prune

เทคนิคในการ Prune มีหลายวิธี แต่แนวทางที่อัลกอริทึม C 4.5 ใช้คือแนวทางแบบ Cross-validation โดยข้อมูลที่มีจะถูกแบ่งออกเป็นส่วนขนาดเท่า ๆ กัน และสำหรับแต่ละส่วนนั้น ทรีจะถูกสร้างขึ้นโดยใช้ข้อมูลจากส่วนอื่น ๆ ทั้งหมด และใช้ข้อมูลส่วนนั้นสำหรับการทดสอบ

ถ้าให้ leaf node มีข้อมูลตกอยู่ทั้งหมด N cases และในจำนวนนั้นมี E cases ที่จัดประเภทผิด อัตราส่วนความผิดพลาด (error rate) ของ leaf node นี้จะเท่ากับ E/N ถ้ามองในเชิงสถิติ โดยมองข้อมูล N cases เป็นกลุ่มตัวอย่าง แล้ว อัตราส่วน E/N ก็มองได้ว่าเป็น ความน่าจะเป็นที่จะเกิดเหตุการณ์ความผิดพลาด E ขึ้นในประชากร cases ทั้งหมดที่ตกอยู่ใน leaf node นี้

ซึ่งความน่าจะเป็นนี้ไม่สามารถกำหนดแน่นอนได้ แต่ถ้ากำหนดค่าระดับความเชื่อมั่น CF ก็จะสามารถเขียนขอบเขตบนของความน่าจะเป็นนี้ได้ในรูป $U_{CF}(E, N)$ และ C4.5 ก็ใช้ค่าขอบเขตบนนี้ในการคาดการณ์ค่าความผิดพลาดของ leaf node โดย leaf node ที่มีข้อมูลตกอยู่ N cases และมีอัตราส่วนความผิดพลาดที่คาดการณ์ไว้เท่ากับ $U_{CF}(E, N)$ คาดว่าจะมีความผิดพลาดได้เท่ากับ $N \times U_{CF}(E, N)$

เพื่อให้เห็นภาพชัดเจนขึ้นจะยกตัวอย่างจากรูปที่ 2.3 ในส่วนทรีย่อย

ค่าใช้จ่ายในการศึกษา = ไม่เห็นด้วย : พรรคเดโมแครต (6)
ค่าใช้จ่ายในการศึกษา = เห็นด้วย : พรรคเดโมแครต (9)
ค่าใช้จ่ายในการศึกษา = งดออกเสียง : พรรครีพับลิกัน (1)

ใน leaf node แรกนั้น N = 6 และ E = 0 ถ้าให้ CF = 25% (0.25) แล้ว $U_{25\%}(0,6) = 0.206$ ดังนั้นจำนวนของความผิดพลาดที่อาจเกิดขึ้นถ้า leaf node นี้ต้องใช้ในการจัดประเภทข้อมูลที่ไม่เคยพบ 6 cases เท่ากับ 6×0.206 ส่วนใน leaf node อื่น ๆ $U_{25\%}(0,9) = 0.143$ และ $U_{25\%}(0,1) = 0.750$ ดังนั้นจำนวนของความผิดพลาดที่อาจเกิดขึ้นของทรีย่อยนี้เท่ากับ

$$6 \times 0.206 + 9 \times 0.143 + 1 \times 0.750 = 3.273$$

ถ้าทำการ Prune ทรีย่อยนี้โดยแทนที่ทรีย่อยนี้ด้วย leaf node "พรรคเดโมแครต" ซึ่งเป็นประเภทที่มีจำนวน cases ตกอยู่มากที่สุดใน leaf node นี้ แล้วจำนวนของความผิดพลาดที่อาจเกิดขึ้นจะเท่ากับ

$$16 \times U_{25\%}(0,6) = 16 \times 0.157 = 2.512$$

จะเห็นว่าจำนวนความผิดพลาดที่จะเกิดขึ้นจากการ Prune มีน้อยกว่าจำนวนความผิดพลาดของทรีย่อย ดังนั้นจึงทำการ Prune ทรีย่อยนี้

2.2 วิธีการเจเนติก (Genetic Algorithms)

Genetic Algorithm เป็นกระบวนการทำงานที่เลียนแบบของธรรมชาติ โดยมีแนวคิดในเรื่องความทนทาน (robustness) ของสิ่งมีชีวิตที่ได้จากการวิวัฒนาการตามธรรมชาติ จึงเกิดความคิดที่จะนำหลักการทางชีววิทยา มาใช้ในการพัฒนาระบบงานทางคอมพิวเตอร์

genetic algorithm มีกระบวนการทำงานหลัก ๆ ดังนี้

1. สุ่มสร้างกลุ่มประชากรเริ่มต้น
2. แต่ละสมาชิกของกลุ่มประชากรนี้จะถูกนำมาประเมินโดยใช้หลักตามแต่ละปัญหา เพื่อหาค่าความสมบูรณ์ (fitness)ให้กับแต่ละสมาชิก
3. สมาชิกที่มีค่า fitness สูง ๆ จะมีแนวโน้มที่จะเป็นผู้ให้กำเนิดสมาชิกใหม่ (parent) ในกระบวนการ reproduction
4. ประชากรกลุ่มใหม่ก็จะมาแทนที่ประชากรกลุ่มเดิม ถือเป็น 1 generation จากนั้นจึงกลับไปเริ่มทำตามข้อ 2 ใหม่

เพื่อให้เข้าใจการทำงานของ genetic ดีขึ้นจะขอยกตัวอย่างปัญหาหนึ่ง คือ ถ้าเรามีกล่องดำอยู่หนึ่งกล่อง ซึ่งเมื่อใส่ตัวเลขเข้าไปแล้วจะให้ค่าจำนวนเต็มออกมา 1 ค่า เราต้องการหาว่าต้องใส่ตัวเลขใดเข้าไปในกล่องดำนี้ จึงจะให้ค่าออกมาเป็น 32 โดยมีเงื่อนไขว่า

- ข้อมูลเข้าเป็นเลขจำนวนเต็มคิดเครื่องหมาย 32-bit
- ข้อมูลที่ได้ออกมาเป็นเลขจำนวนเต็มระหว่าง 0 - 32
- ข้อมูลเข้าหลาย ๆ ตัวอาจให้ค่าออกมเป็นเลขเดียวกันได้ ยกเว้นข้อมูลเข้าที่จะให้ค่าเป็น 32
- ไม่มีความสัมพันธ์ที่เด่นชัดระหว่างข้อมูลเข้าและผลลัพธ์ที่ได้ออกมา

2.2.1 กลุ่มประชากร (Population)

แต่ละสมาชิกของประชากรจะเป็น 32-bit string ซึ่งแทนข้อมูลเข้า และมี array ที่มีขนาดเท่ากับจำนวนประชากรเพื่อเก็บค่าผลลัพธ์ที่ได้จากการใส่สมาชิกแต่ละตัวเข้าไปในกล่องดำ

2.2.2 การเลือกโดยใช้วงล้อรูเล็ต (Roulette Wheel Selection)

วงล้อรูเล็ตเป็นการละเล่นอย่างหนึ่งซึ่งจะมีวงล้อหมุนที่ถูกแบ่งออกเป็น 37-38 ส่วนเท่า ๆ กัน เล่นโดยการหมุนวงล้อและโยนลูกหินเข้าไปในวงล้อในทิศตรงข้ามกับที่วงล้อหมุน เมื่อวงล้อหยุดลง ลูกหินก็จะตกอยู่ในช่อง ๆ หนึ่งภายในวงล้อนั้น

ในเจเนติกจะใช้หลักการของวงล้อรูเล็ตในการเลือกสมาชิกจากกลุ่มประชากรเดิม ไปเป็นกลุ่มประชากรใหม่ในการ reproduction ซึ่งวงล้อก็คือ array ของค่า fitness นั่นเอง และลูกหินก็คือ

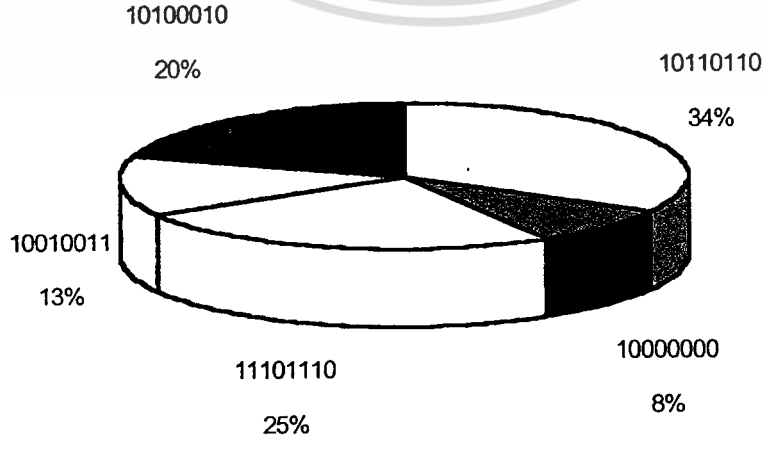
เลขจำนวนเต็มที่สุ่มมาที่น้อยกว่าผลรวมค่า fitness ทั้งหมดในกลุ่มประชากรนั้น ในการหา bit string ที่สัมพันธ์กับจุดที่ลูกหินหยุด จะทำโดยกระบวนการซ้ำ ๆ ที่ว่า:

ถ้าค่าของลูกหินน้อยกว่าค่า fitness ที่เก็บอยู่ใน array ตำแหน่งปัจจุบัน แล้ว
bit string ที่สัมพันธ์กับ fitness array ตำแหน่งนั้นก็จะกลายเป็น parent
ถ้าไม่เช่นนั้น จะลบค่าของลูกหินด้วยค่า fitness ณ ตำแหน่ง array นั้น
แล้วจึงพิจารณา fitness array ในตำแหน่งถัดไป

ดังนั้นค่า fitness ที่มากที่สุดจึงมีแนวโน้มที่จะได้แทนที่ค่าในลูกหิน เพราะมันเหมือนกับมีพื้นที่มากที่สุดในวงล้อเสมือนนี้ เช่น ถ้ามีประชากรและค่า fitness ดังแสดงในตารางที่ 2.2 ซึ่งผลรวมค่า fitness ทั้งหมดของกลุ่มประชากรนี้เท่ากับ 60 แล้วรูปที่ 2.4 จะแสดงแผนภูมิวงกลมที่แทนขนาดของช่องในวงล้อที่แต่ละ bit string จะได้รับ

ตารางที่ 2.2 ตัวอย่างค่า fitness ของแต่ละ bit string

bit string	Fitness
10110110	20
10000000	5
11101110	15
10010011	8
10100010	12



รูปที่ 2.4 แผนภูมิวงกลมที่แทนขนาดของช่องในวงล้อที่แต่ละ bit string จะได้รับ

จากรูปที่ 2.4 แสดงว่า 10110110 มีโอกาส 34% ที่จะถูกเลือกเป็น parent ขณะที่ 10000000 มีโอกาสเพียง 8% เท่านั้น

ถ้าต้องการเลือก parent 5 ตัว ก็ต้องมีการสร้างตัวเลขสุ่ม 5 ครั้ง ถ้าตัวเลขที่สุ่มมาเป็น 44, 5, 49, 18, และ 22 สมาชิกที่ได้รับเลือกเป็น parent จะแสดงในตารางที่ 2.3 ซึ่งจะเห็นว่า 10110110 ที่มีค่า fitness สูงที่สุด ถูกเลือกมาถึง 2 ครั้ง แม้แต่ 10000000 ที่มีค่า fitness น้อยที่สุดก็ถูกเลือกมา 1 ครั้งเช่นกัน แต่ 11101110 ที่มีค่า fitness สูงเป็นอันดับ 2 กลับไม่ถูกเลือกเลย

ตารางที่ 2.3 ผลการเลือก parent

ตัวเลขที่สุ่มเลือกมา	สมาชิกที่ได้รับเลือกเป็น parent
44	10010011
5	10110110
49	10100010
18	10110110
22	10000000

2.2.3 การถ่ายทอดพันธุกรรม (Crossover)

Crossover เป็นตัวปฏิบัติการสำคัญของเจเนติกในการสร้างสมาชิกใหม่ขึ้นมาจาก parent 2 ตัว ตามหลักพันธุกรรม โดยการสุ่มเลือกจุดหนึ่งจากคู่ของ parent เพื่อทำการสับเปลี่ยน bit string กัน ตัวอย่างการทำ crossover ดังแสดงในตารางที่ 4

ตารางที่ 2.4 ตัวอย่างการทำ crossover

Parent1	Parent2	Crossover Point	New (Child)
10010011	10110110	3	100 | 100110
10000000	10110110	6	100000 | 10
10110110	11101110	2	10 | 101110
10110110	11101110	5	10110 | 110

2.2.4 การผ่าเหล่า (Mutation)

ขั้นตอนสุดท้ายในการทำ reproduction คือการทำ mutation ซึ่งเป็นการสุ่มเปลี่ยน 1 หรือมากกว่า 1 bit ในแต่ละ bit string ของกลุ่มประชากรใหม่ ซึ่งจุดประสงค์หลักของการทำ mutation คือเพื่อเพิ่มความแปรปรวนให้กับกลุ่มประชากร (mutation จะมีความสำคัญมาก ถ้ากลุ่มประชากรเริ่มต้นเป็นเพียงกลุ่มย่อยขนาดเล็กของค่าที่เป็นไปได้ทั้งหมด) ตัวอย่างเช่น ถ้าในการสุ่มสร้างกลุ่มประชากรเริ่มต้น ปรากฏว่าทุก ๆ สมาชิกในประชากรมีอยู่ bit หนึ่งที่ให้ค่า 0 ทั้งหมด ดังนั้นการ crossover ไม่ว่ากี่ครั้งก็ไม่สามารถเปลี่ยน bit นั้นให้กลายเป็น 1 ได้ จึงต้องมีการทำ mutation

บทที่ 3

อัลกอริทึมที่ใช้ในการจัดประเภทที่คำนึงถึงค่าใช้จ่าย (ICET)

ICET ย่อมาจาก Inexpensive Classification with Expensive Test เป็นอัลกอริทึมที่จัดประเภทโดยคำนึงถึงค่าใช้จ่าย พัฒนาขึ้นโดย Peter D. Turney [1995] มีคุณสมบัติดังนี้:

1. คำนึงถึงค่าใช้จ่ายของการทดสอบ
2. คำนึงถึงค่าใช้จ่ายของการแบ่งประเภทที่ผิดพลาด
3. ใช้การค้นหาแบบ greedy heuristic ร่วมกับวิธีการของเจเนติก
4. สามารถจัดการกับค่าใช้จ่ายที่เป็นเงื่อนไข (เมื่อค่าใช้จ่ายของการทดสอบหนึ่งขึ้นกับว่ามีการเลือกทำอีกการทดสอบหนึ่งไปหรือยัง)
5. แยกการทดสอบออกเป็นสองแบบ คือ แบบที่ให้ผลทันที (immediate) และแบบที่ให้ผลล่าช้า (delayed)

อัลกอริทึม ICET ได้นำวิธีการของเจเนติกมาใช้ร่วมกับดิสิชั่นทรี โดยเจเนติกจะวิวัฒนาการประชากรของ biases ให้อัลกอริทึมของ decision tree ซึ่ง ICET จะใช้อัลกอริทึม GENESIS ในการทำเจเนติก และในการทำดิสิชั่นทรีจะใช้อัลกอริทึม EG2 ที่ปรับมาจาก อัลกอริทึม C4.5

3.1 อัลกอริทึมของดิสิชั่นทรี

อัลกอริทึมในการสร้างดิสิชั่นทรีที่มีการนำค่าใช้จ่ายมาคำนวณด้วยมีอยู่หลายอัลกอริทึมด้วยกัน เช่น EG2 (Nunez, 1991), CS-ID3 (Tan & Schlimmer, 1989, 1990; Tan, 1993), IDX (Norton, 1989) แต่ ICET เลือกนำอัลกอริทึม EG2 มาปรับใช้กับวิธีการทำงานของ อัลกอริทึม C4.5 (Quinlan, 1992)

3.1.1 อัลกอริทึม C4.5

วิธีการของ C4.5 จะสร้างดิสิชั่นทรีโดยใช้วิธีการ TDIDT (Top-Down Induction of Decision Tree) ทำการแบ่งข้อมูลออกเป็นกลุ่มย่อย ๆ ที่เล็กลงเรื่อย ๆ โดยดูตามค่าของข้อมูลที่อยู่ใน attribute นั้น ๆ โดยในแต่ละขั้นของการสร้าง decision tree นั้น C4.5 จะเลือก attribute ที่มีค่า information gain สูงที่สุด

3.1.2 อัลกอริทึม EG2

EG2 (Nunez, 1991) เป็นอัลกอริทึมแบบ TDIDT เช่นเดียวกับ C4.5 แต่ EG2 ใช้ค่า ICF (Information Cost Function) ในการเลือก attribute โดยดูทั้งค่า information gain และค่าใช้จ่าย (cost) ของ attribute นั้น สมการหาค่า ICF ของ attribute ที่ i คือ:

$$ICF_i = \frac{2^{\Delta I_i} - 1}{(c_i + 1)^{\varpi}} \; ; 0 \le \varpi \le 1$$

ในสมการนี้ ΔI_i คือค่า information gain ของ attribute ที่ i และ C_i คือค่าใช้จ่ายของการวัด attribute ที่ i ส่วนพารามิเตอร์ ϖ เป็นการปรับค่าความแข็งแรงของ bias ตามจำนวนค่าใช้จ่ายที่ต่ำลง โดยเมื่อ $\varpi = 0$ ถือว่าไม่ต้องนำค่าใช้จ่ายมาสนใจ และการเลือกของ ICF_i จะเหมือนกับการเลือกโดยใช้ information gain (ΔI_i) ปกติ ถ้า $\varpi = 1$ แสดงว่าค่า ICF_i จะขึ้นกับค่าใช้จ่ายมาก ทำให้การทำงานของอัลกอริทึมไวต่อค่าใช้จ่ายจากการจัดประเภท ซึ่งสมการหาค่า ICF_i เมื่อให้ $\varpi = 1$ คือ:

$$ICF_i = \frac{2^{\Delta I_i} - 1}{(c_i + 1)}$$

3.1.3 อัลกอริทึม CS-ID3

CS-ID3 (Tan & Schlimmer, 1989, 1990; Tan, 1993) ก็มีวิธีการทำงานแบบ TDIDT เช่นกัน โดย CS-ID3 จะเลือกแอททริบิวต์ที่ให้ค่าตามฟังก์ชันด้านล่างสูงที่สุด

$$\frac{(\Delta I_i)^2}{(C_i)}$$

3.1.4 อัลกอริทึม IDX

CS-ID3 (Tan & Schlimmer, 1989, 1990; Tan, 1993) ก็มีวิธีการทำงานแบบ TDIDT เช่นกัน โดย CS-ID3 จะเลือกแอททริบิวต์ที่ให้ค่าตามฟังก์ชันด้านล่างสูงที่สุด

$$\frac{\Delta I_i}{(C_i)}$$

สำหรับวิธีการในการสร้างดิสิชั่นทรีของ ICET นำวิธีการหาค่า ICF ของ EG2 มาปรับใช้กับวิธีการทำงานของ C4.5 คือใช้ค่า ICF ของ EG2 แทนค่า Information Gain ของ C4.5 เนื่องจาก EG2 มีพารามิเตอร์ ϖ ทำให้สามารปรับค่าได้ว่าให้ความสนใจกับค่าใช้จ่ายเพียงใด

3.2 อัลกอริทึมของเจเนติก

วิธีการของเจเนติกที่นำมาใช้ในโครงการพัฒนาระบบงานนี้ คือGENESIS (GENEtic Search Implementation System) ซึ่งเป็นระบบที่พัฒนาขึ้นโดย John J. Grefenstette เพื่อสนับสนุนการศึกษา genetic algorithms โดยกระบวนการทำงานหลัก ๆ ของ GENESIS มีดังนี้:

- Initialization เป็นกระบวนการทำการสร้างกลุ่มประชากรเริ่มต้น
- Selection เป็นกระบวนการในการเลือกโครงสร้าง (bit strings) สำหรับรุ่นถัดไป จากโครงสร้างที่มีอยู่ในรุ่นปัจจุบันโดยยึดหลักที่ว่า โอกาสที่โครงสร้างนั้นจะถูกเลือก เป็นไปตามสัดส่วนของค่า fitness ของโครงสร้างนั้น
- Mutation เป็นกระบวนการในการสุ่มเปลี่ยนค่าในบางตำแหน่งของบางโครงสร้าง
- Crossover เป็นกระบวนการในการแลกเปลี่ยนบางส่วนของ binary representation ของสองโครงสร้าง
- Evaluation เป็นกระบวนการที่ใช้ประเมินเพื่อหาค่า fitness ให้แต่ละโครงสร้างในกลุ่มประชากร โดยกระบวนการนี้จะให้ผู้ใช้เป็นผู้กำหนดขึ้นเองตามแต่ละปัญหา

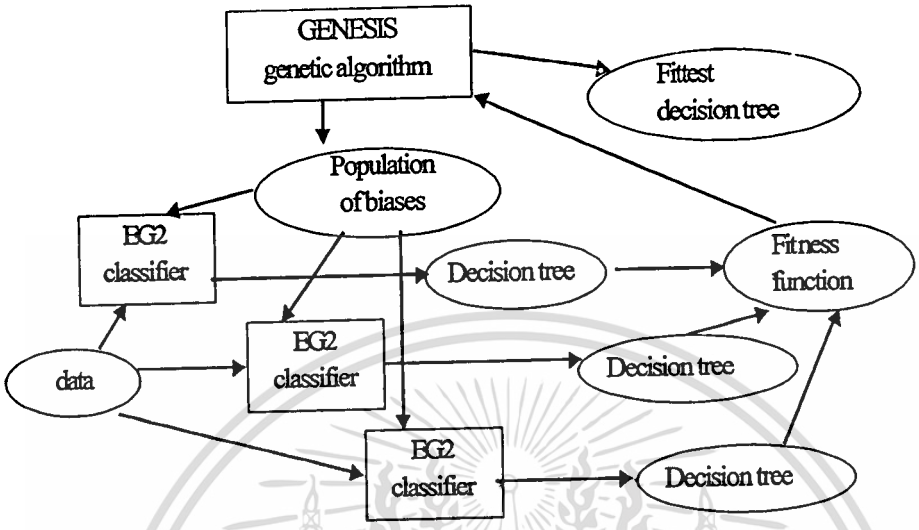
3.3 อัลกอริทึม ICET

ICET เป็นอัลกอริทึมที่ใช้การค้นหาแบบ 2 ชั้น (2-tired) คือในชั้นล่างจะใช้ EG2 ทำการค้นหาแบบ greedy ทั่วพื้นที่ของ decision tree โดยใช้มาตรฐาน TDIDT ส่วนในชั้นบนจะใช้ GENESIS ทำการค้นหาแบบ genetic ทั่วพื้นที่ของ biases และนำ biases มาปรับพฤติกรรมของ EG2

ICET ไม่ได้ใช้ EG2 ตามที่มันถูกออกแบบมา โดยค่า C_i ไม่ได้ใช้เป็นค่าใช้จ่าย (cost) แต่ใช้เป็นพารามิเตอร์ bias ของ ICET แทน ดังนั้น ICET จึงทำการควบคุม bias ของ EG2 โดยการปรับค่าพารามิเตอร์ C_i โดยใน ICET นั้น ค่าของพารามิเตอร์ C_i ไม่มีความเกี่ยวข้องโดยตรงกับค่าใช้จ่ายที่แท้จริงของการทดสอบ

วิธีการของเจเนติกนั้นคล้ายกับการวิวัฒนาการทางชีวภาพ โดยแต่ละตัวที่ GENESIS วิวัฒนาการนั้นจะเป็นแถวของบิต (bit strings) ซึ่ง GENESIS จะเริ่มต้นด้วยประชากรของ bit strings ที่สุ่มสร้างขึ้นมา จากนั้นมันจะวัดค่าความสมบูรณ์ (fitness) ของแต่ละ bit string ซึ่งใน ICET นั้น bit string ก็คือ bias สำหรับ EG2 นั่นเอง ส่วนค่า fitness ของแต่ละ bit string ก็คือค่าใช้จ่ายเฉลี่ยของการจัดประเภทของ decision tree ที่สร้างโดย EG2 จากนั้นในรุ่น (generation) ถัดไปประชากรของ biases ก็จะถูกแทนที่ด้วย bit string ใหม่ที่สร้างขึ้นมาจากรุ่นที่ผ่านมา โดยใช้การ mutation และ crossover ดังนั้น bit string ที่สมบูรณ์ที่สุด (fittest) จากรุ่นที่ 1 ก็จะมีลูกหลานมากที่

สุดในรุ่นที่ 2 และหลังจากทำงานไปจนถึงจำนวนรุ่นที่กำหนด ICET ก็จะหยุดการทำงานและได้ผลลัพธ์ออกมาเป็น decision tree ที่กำหนดโดย bit string ที่สมบูรณ์ที่สุด



รูปที่ 3.1 โครงสร้างของอัลกอริทึม ICET

3.4 การทำงานของ ICET

ในการใช้งาน GENESIS ในอัลกอริทึม ICET นั้น ต้องมีการกำหนดค่าพารามิเตอร์ให้กับ GENESIS โดยตัวอย่างการกำหนดพารามิเตอร์ดังแสดงในตารางที่ 3.1

จากตารางที่ 3.1 ขนาดของประชากรเท่ากับ 50 bit strings และมี 1,000 trials จึงได้รุ่น (generation) ทั้งหมด 20 รุ่น แต่ละ bit string ของประชากรประกอบด้วย string ของ n+2 ตัวเลข โดย n คือจำนวนของ attribute (การทดสอบ) ของกลุ่มข้อมูลที่กำหนด ซึ่ง n+2 ตัวเลขนี้จะอยู่ในรูปแบบ binary โดยใช้ Gray code ซึ่ง binary string นี้ก็จะใช้เป็น bias ของ EG2 โดยตัวเลข n ตัวเลขแรกนั้นจะใช้เป็นค่าใช้จ่าย C_i ในการคำนวณ ICF (สมการที่ 1) โดยมีค่าตั้งแต่ 1 ถึง 10,000 และแต่ละตัวจะแทนด้วยเลข binary 12 หลัก ส่วนตัวเลขอีก 2 ตัวหลังใน string จะใช้ในการกำหนดค่าให้กับ ϖ และ CF โดย ϖ เป็นพารามิเตอร์ที่ใช้ในสมการ ICF ส่วน CF นั้นเป็นพารามิเตอร์ที่ใช้ใน C4.5 เพื่อควบคุมระดับการ pruning ของ decision tree ซึ่งทั้งสองจำนวนนี้ แต่ละตัวจะแทนด้วยเลข binary 8 หลัก โดย ϖ จะมีค่าตั้งแต่ 0 (คือการที่ไม่สนใจค่าใช้จ่าย) ถึง 1 (คำนึงถึงค่าใช้จ่ายมากที่สุด) และ CF จะมีค่าตั้งแต่ 1 (การ pruning สูง) ถึง 100 (การ pruning ต่ำ) ดังนั้นในแต่ละ bit string จะประกอบด้วยเลข 12n+16 บิต

ตารางที่ 3.1 ตัวอย่างการกำหนดค่าพารามิเตอร์ให้กับ GENESIS

พารามิเตอร์	กำหนดค่า
Experiments	1
Total Trials	1000
Population Size	50
Structure Length	12n + 16
Crossover Rate	0.6
Mutation Rate	0.001
Generation Gap	1.0
Scaling Window	5
Report Interval	100
Structures Saved	1
Max Gens without Evaluation	2
Dump Interval	0
Dumps Saved	0
Rank Min	0.75

แต่ละรอบของ bit string หนึ่ง ๆ จะประกอบด้วยการทำงานของ EG2 (โดยใช้ในลักษณะเป็นการปรับเปลี่ยนมาจาก C4.5) กับกลุ่มข้อมูลสำหรับการฝึกสอน (training dataset)โดยใช้ตัวเลขใน bit string กำหนดค่าให้กับ C_i (i = 1,..., n), ϖ และ CF

กลุ่มข้อมูลสำหรับการฝึกสอนนี้จะถูกสุ่มแบ่งออกเป็น 2 กลุ่มย่อยที่มีขนาดเท่ากัน (± 1 สำหรับกลุ่มที่มีขนาดเป็นจำนวนคี่) เป็น sub-training set และ sub-testing set ซึ่งในแต่ละรอบนั้นข้อมูลจะถูกสุ่มแบ่งไม่เหมือนกัน ดังนั้นแม้ว่าจะเป็น bit string ที่มีตัวเลขเหมือนกันก็อาจให้ผลไม่เหมือนกันได้ เราจึงต้องนำทุก ๆ bit string มาพิจารณา แม้ว่าอาจจะมี bit string ที่เหมือนกันในกลุ่มประชากร และมีค่าความสมบูรณ์ (fitness) ที่ต่างกันน้อยมากก็ตาม การวัดค่าความสมบูรณ์ของแต่ละ bit string นั้นคือการวัดค่าใช้จ่ายเฉลี่ยของการจัดประเภทกับ sub-testing set โดยใช้ decision tree ที่สร้างขึ้นจาก sub-training set หลังจากทำงานครบ 1,000 รอบ bit string ที่สมบูรณ์ที่สุด (มีค่าใช้จ่ายน้อยที่สุด) จะใช้เป็น bias สำหรับ EG2 ซึ่งจะทำงานโดยใช้กลุ่มข้อมูลสำหรับการฝึกสอนทั้งชุดเป็นข้อมูลเข้า (input) จากนั้นผลลัพธ์จาก ICET จากกลุ่มข้อมูลสำหรับการฝึกสอนนั้น ก็จะเป็น decision tree ที่ต้องการ

ค่าใช้จ่าย C_i ที่ใช้ในสมการ ICF ไม่ได้มีความสัมพันธ์โดยตรงกับค่าใช้จ่ายที่แท้จริงของ attribute เนื่องจาก 50 bit strings ในการทำงานรุ่นแรกนั้นถูกสร้างขึ้นแบบสุ่ม ดังนั้นค่าเริ่มต้นของ C_i จึงไม่มีความสัมพันธ์กับค่าใช้จ่ายที่แท้จริงเลย แต่หลังจากการทำงานครบ 20 รุ่น ค่าของ C_i อาจมีความสัมพันธ์กับค่าใช้จ่ายที่แท้จริง แต่ไม่ใช่ความสัมพันธ์โดยตรง ดังนั้นค่าต่าง ๆ ของ C_i ควรถูกมองเป็นค่า biases มากกว่าจะเป็นค่าใช้จ่าย

ค่า biases C_i มีค่าตั้งแต่ 1 ถึง 10,000 โดยเมื่อใดที่ C_i มีค่าสูงกว่า 9,000 แล้ว attribute ที่ i จะถูกตัดทิ้งไป ดังนั้น C4.5 จะไม่สามารถใส่ attribute ที่ i นี้ไปใน decision tree ได้ แม้ว่า attribute นี้อาจจะทำให้ ICF มีค่าสูงก็ตาม

การที่เราเลือกใช้ EG2 ใน ICET เนื่องจาก EG2 มีพารา-มิเตอร์ ϖ ซึ่งทำให้ GENESIS สามารถควบคุม bias ของ EG2 ได้ โดย ICF_i นั้นส่วนหนึ่งจะขึ้นอยู่กับข้อมูล (ด้วย information gain ΔI_i) และส่วนหนึ่งขึ้นอยู่กับค่า biases (ค่าใช้จ่ายเทียม C_i)

ถือว่า GENESIS นั้น "หลอก" EG2 ในเรื่องค่าใช้จ่ายของการทดสอบ แต่ก็เพื่อปรับปรุงประสิทธิภาพในการทำงานของ EG2 เนื่องจาก EG2 นั้นมีวิธีการทำงานที่อาจทำให้เกิดการพบจุดที่ดีที่สุดปลอม (local optimum) ได้ การทำงานของ EG2 นั้นเป็นแบบ greedy ที่จะมองเพียงแค่การทดสอบเดียวล่วงหน้าขณะที่ทำการสร้าง decision tree ดังนั้น ขณะที่ EG2 พยายามจะหลีกเลี่ยงการทดสอบที่มีค่าใช้จ่ายสูง โดยเลือกการทดสอบที่มีค่าใช้จ่ายต่ำกว่า มันก็อาจไปพบกับการทดสอบที่มีค่าใช้จ่ายสูงกว่าอีกถัดจากนั้น ซึ่ง GENESIS จะช่วยป้องกันการมองเพียงระยะสั้นของ EG2 ด้วยการ "หลอก" โดย GENESIS อาจจะบอกค่าใช้จ่ายของการทดสอบที่ต่ำกว่าหรือสูงกว่าความเป็นจริง ซึ่งทำให้ EG2 มีประสิทธิภาพในการทำงานดีขึ้น

ในการทำงานของ ICET นั้น การเรียนรู้ (การค้นหาแบบ local ใน EG2) และการวิวัฒนาการ (ใน GENESIS) สามารถทำงานโต้ตอบร่วมกันได้

3.5 การคำนวณค่าใช้จ่ายเฉลี่ย

หลังจาก EG2 สร้างดิสิชั่นทรีขึ้นมาแต่ละต้นแล้ว ต้องมีการคำนวณค่าใช้จ่ายเฉลี่ยของทรีต้นนั้น เพื่อใช้เป็นค่าประสิทธิภาพของ bitstring ที่ใช้ในการสร้าง tree ต้นนั้น เพื่อให้ GENESIS นำไปใช้ในกระบวนการเจเนติก เพื่อสร้าง bitstring ในรุ่นต่อมา

ค่าใช้จ่ายเฉลี่ยของทรี เท่ากับ ผลรวมของค่าใช้จ่ายจากการทดสอบ กับ ค่าใช้จ่ายจากการทำนายที่ผิดพลาด แล้วหารด้วยจำนวนของข้อมูลทั้งหมดที่นำมาใช้คำนวณ

ค่าใช้จ่ายจากการทดสอบจะคำนวณโดยคิดไล่ลงมาตามดิสิชั่นทรี โดยบวกค่าใช้จ่ายเพิ่มเข้าไปตามการทดสอบที่เกิดขึ้นระหว่างทางจากราก (root) ไปยังส่วนปลาย (leaf node) ของทรี โดยถ้ามีการทดสอบเดิมเกิดขึ้นมากกว่าหนึ่งครั้ง จะคิดค่าใช้จ่ายเพียงครั้งแรกเท่านั้น

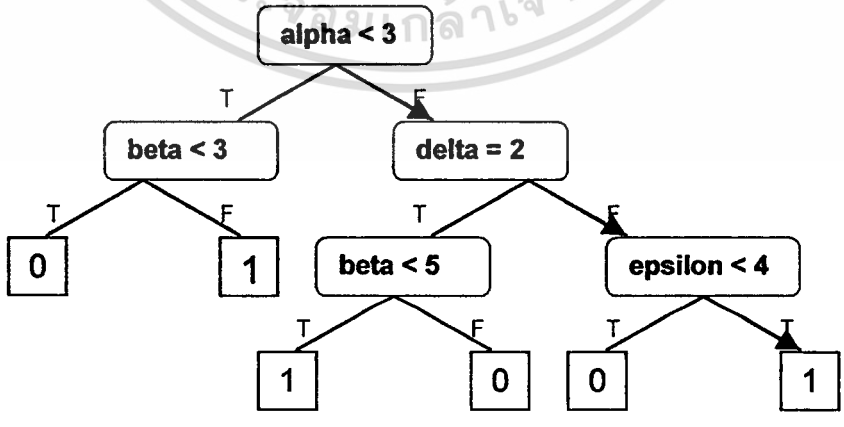
ค่าใช้จ่ายจากการจัดประเภทที่ผิดพลาด คำนวณโดยใช้ classification cost matrix มาวัดค่าใช้จ่ายจากการทำนายของ tree ซึ่ง classification cost matrix คือ เมตริกซ์ขนาด C x C โดย $C_{i,j}$ คือ ค่าใช้จ่ายที่เกิดจากการทำนายว่า case หนึ่งจัดอยู่ในประเภท i แต่ในความเป็นจริง case นั้นจัดอยู่ในประเภท j

เพื่อให้เห็นภาพจะขอยกตัวอย่างการคำนวณค่าใช้จ่าย โดยให้ข้อมูลมี 4 attribute คือ alpha, beta, delta, และ epsilon ซึ่งแต่ละ attribute มีค่าใช้จ่ายดังตารางที่ 3.2

ตารางที่ 3.2 ตัวอย่างค่าใช้จ่ายจากการทดสอบ

	การทดสอบ	ค่าใช้จ่าย
1	Alpha	50 บาท
2	Beta	100 บาท
3	Delta	70 บาท
4	Epsilon	100 บาท

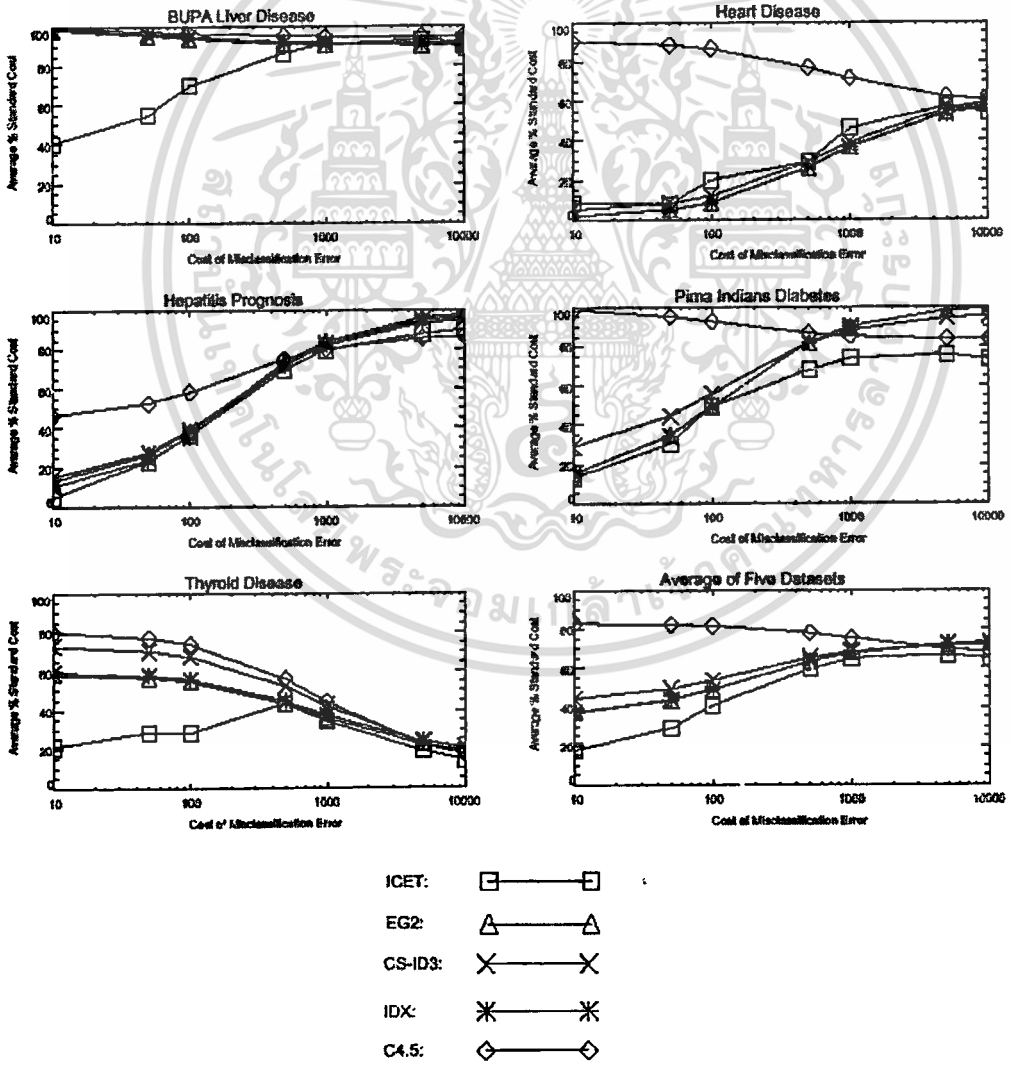
ส่วนค่าใช้จ่ายจากการจัดประเภทที่ผิดพลาดเท่ากับ 500 บาท และโครงสร้างดิสิชั่นทรีที่ได้มาเป็นดังแสดงในรูปที่ 3.2



รูปที่ 3.2 ดิสิชั่นทรีที่สร้างขึ้นมา

ถ้าข้อมูลที่เข้ามามีค่า alpha = 6, beta = 5, delta = 3 และ epsilon = 2 ข้อมูลจะตกตามกิ่งดังแสดงตามลูกศรในรูปที่ 3.2 ดังนั้นค่าใช้จ่ายจากการทดสอบของข้อมูล case นี้เท่ากับ 50 (alpha) + 70 (delta) + 100 (epsilon) = 220 จากทรีจะจัดในข้อมูลนี้ตกอยู่ในประเภท "1" แต่ถ้าประเภทที่แท้จริงของข้อมูลนี้เป็น "0" แล้ว จะมีค่าใช้จ่ายจากการจัดประเภทที่ผิดพลาดอีก 500 บาท รวมค่าใช้จ่ายของข้อมูล case นี้เป็น 720 บาท เป็นต้น

รูปที่ 3.3 แสดงผลการทดลองของ Peter D. Turney ในการจัดประเภทข้อมูลกับกลุ่มข้อมูล 5 กลุ่มคือ BUPA Liver Disease, Hepatitis Prognosis, Thyroid Disease, Heart Disease และ Pima Indians Diabetes โดยเปรียบเทียบผลระหว่างการใช้อัลกอริทึม C4.5, EG2, CS-ID3, IDX และ ICET ในการจัดประเภท



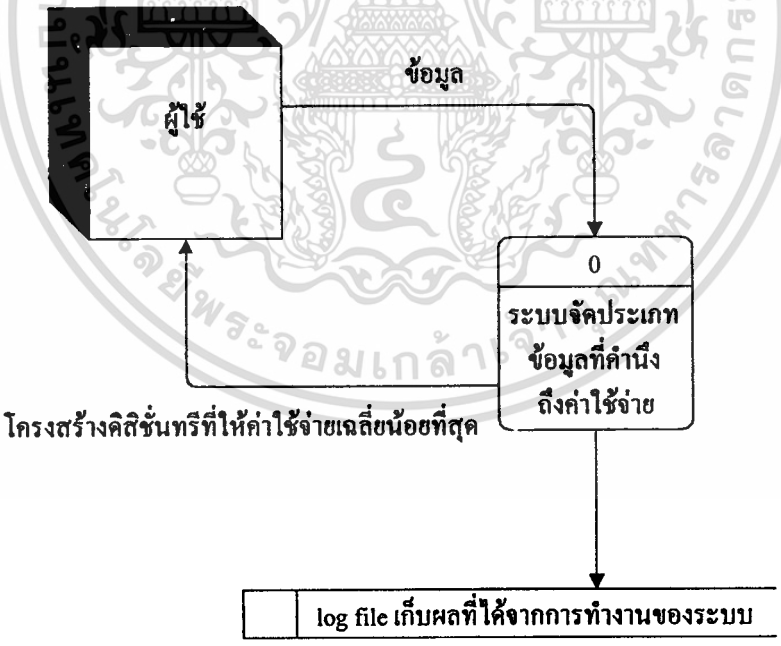
รูปที่ 3.3 เปรียบเทียบค่าใช้จ่ายเฉลี่ยในการจัดประเภทข้อมูลโดยใช้ 5 วิธีการ

บทที่ 4

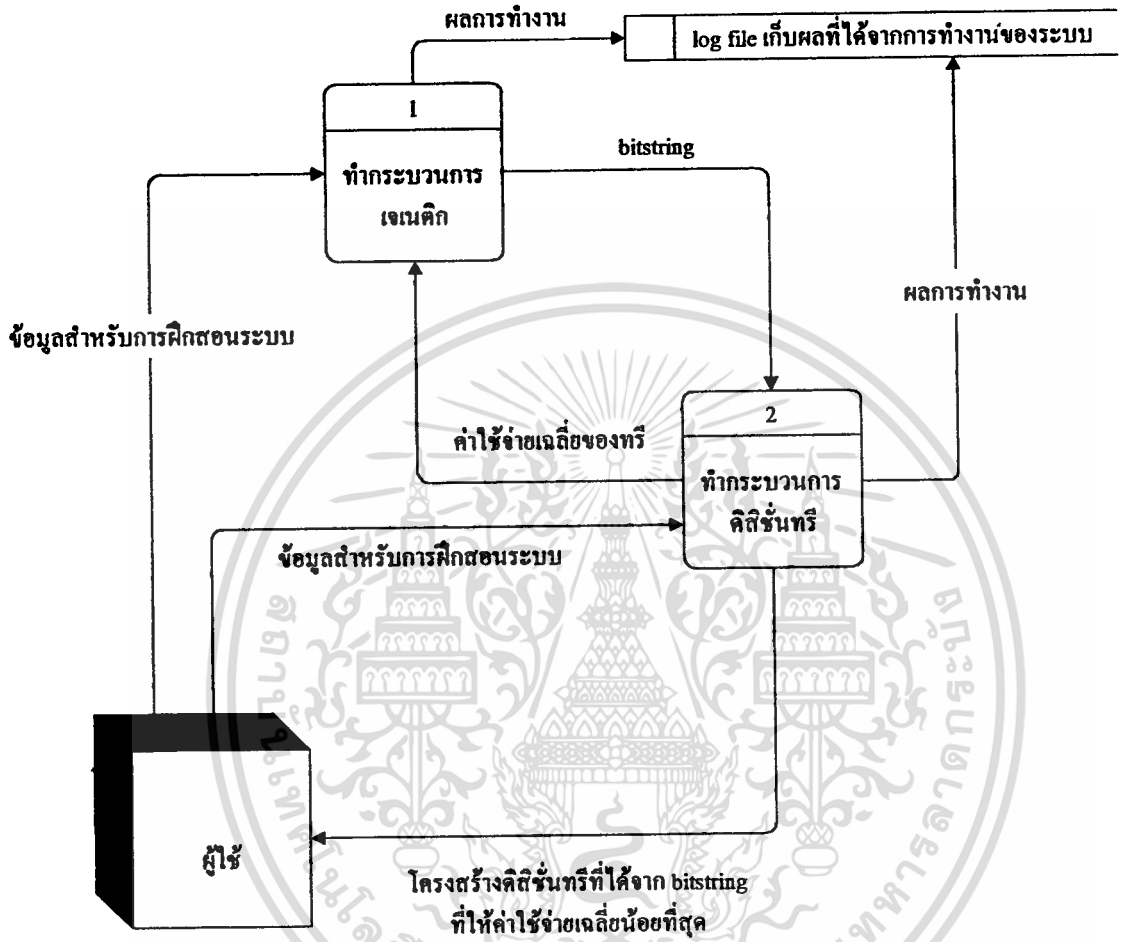
การออกแบบระบบงาน

4.1 การออกแบบโปรแกรม

จากการศึกษาอัลกอริทึมการจัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย (ICET) จึงได้ออกแบบระบบงาน โดยระบบงานสามารถแบ่งออกเป็นส่วนการทำงานย่อยได้ 2 ส่วน คือส่วนกระบวนการทาง เจเนติก ที่ทำการสร้าง bitstring ขึ้นมา และส่วนกระบวนการทางดิสิชั่นทรีที่นำ bitstring ที่ได้มาใช้ในการคำนวณเพื่อสร้างดิสิชั่นทรีขึ้นมา โดยสามารถนำระบบงานมาเขียนเป็น context diagram ได้ดังแสดงในรูปที่ 4.1 และเป็น data flow diagram ได้ดังแสดงในรูปที่ 4.2, 4.3 และรูปที่ 4.4 ตามลำดับ

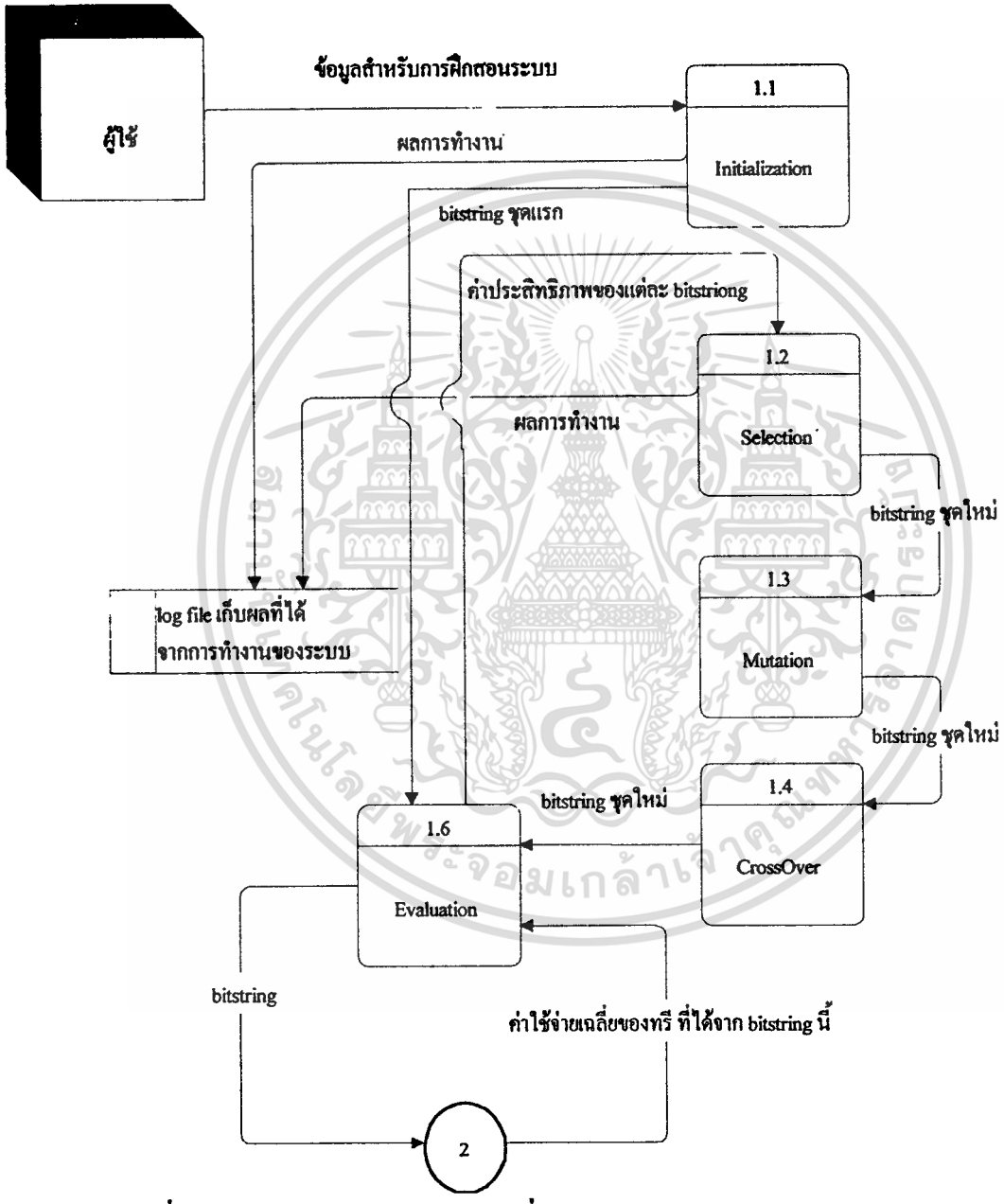


รูปที่ 4.1 Context Diagram ของระบบจัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย

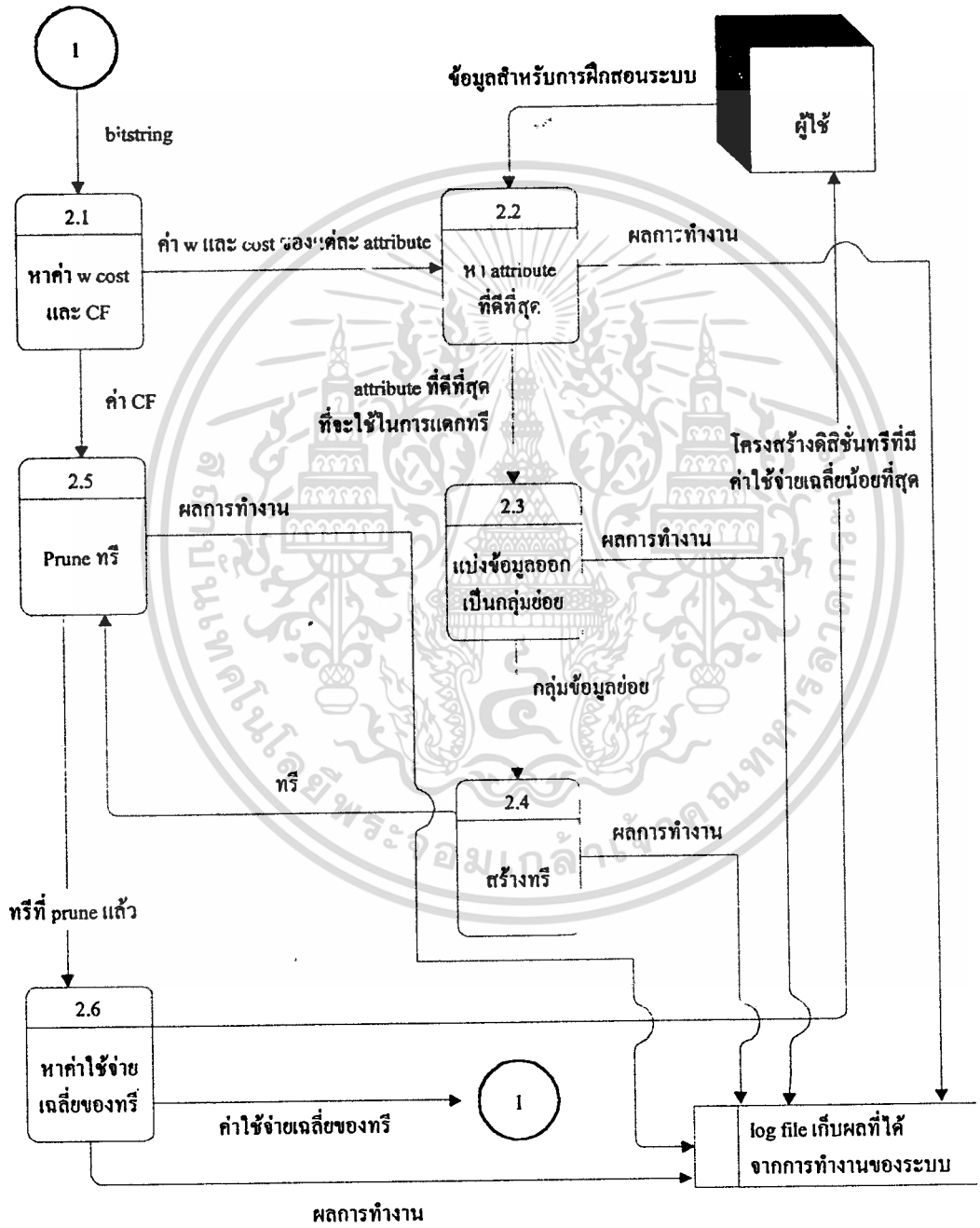


รูปที่ 4.2 Data Flow Diagram ระดับที่1

งานของ Process ที่ 1 ทำกระบวนการเจเนติก สามารถแตกออกเป็นงานย่อยได้ดังแสดงในรูปที่ 4.3 และงานของ Process ที่ 2 ทำกระบวนการดิสิชั่นทรี สามารถแตกออกเป็นงานย่อยได้ดังแสดงในรูปที่ 4.4



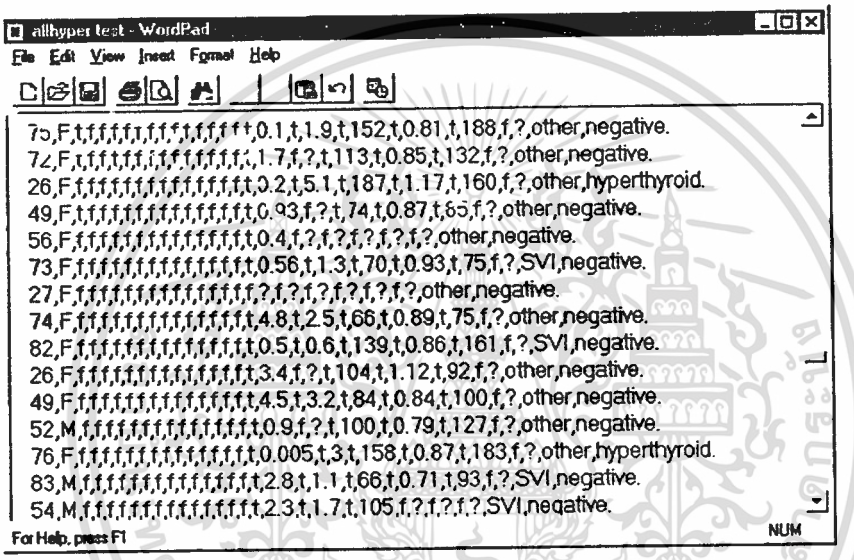
รูปที่ 4.3 Data Flow Diagram ระดับที่ 2 ของ Process ทำกระบวนการเจเนติก



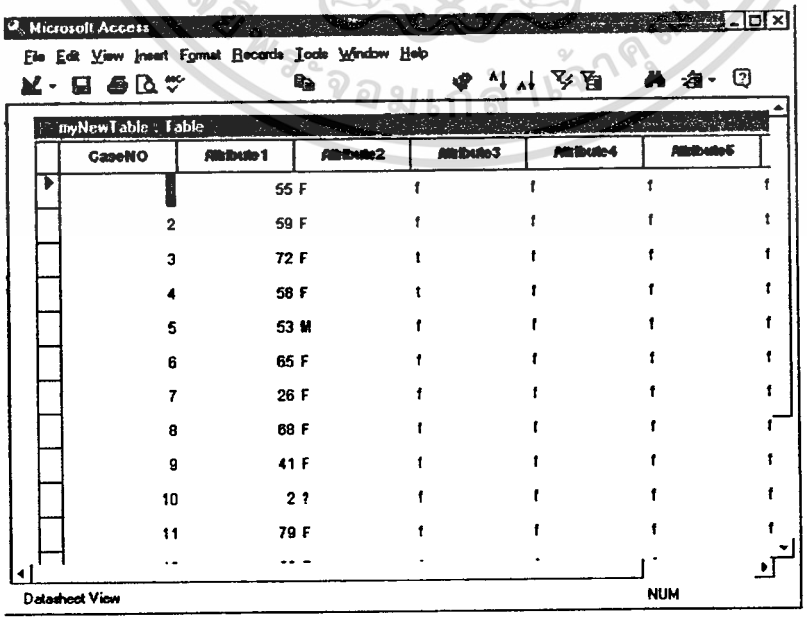
รูปที่ 4.4 Data Flow Diagram ระดับที่ 2 ของ Process ทำกระบวนการดิสิชั่นทรี

4.2 ลักษณะข้อมูลเข้าระบบ

ลักษณะข้อมูลเข้าสำหรับเครื่องมือช่วยวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่ายโดยใช้วิธีการทางเจเนติกร่วมกับดิสิชั่นทรีนี้ ต้องประกอบด้วยแอททริบิวต์ที่มีผลต่อการตัดสินกำหนดประเภทของข้อมูล และแอททริบิวต์ที่เป็นประเภท (class) โดยอาจเป็นเท็กซ์ไฟล์ดังแสดงในรูปที่ 4.5 หรือเป็นตารางในฐานข้อมูลดังในแสดงในรูปที่ 4.6 โดยผู้ใช้จะต้องระบุประเภท (class) ที่ข้อมูลถูกจัดอยู่ไว้ที่แอททริบิวต์สุดท้าย



รูปที่ 4.5 ตัวอย่างข้อมูลเข้าแบบเท็กซ์ไฟล์



รูปที่ 4.6 ตัวอย่างข้อมูลเข้าแบบฐานข้อมูล

นอกจากนั้นยังต้องมีข้อมูลเกี่ยวกับค่าใช้จ่ายที่จะเกิดขึ้นในการทดสอบแต่ละแอททริบิวต์ เช่น ถ้ามีข้อมูลดังแสดงในตารางที่ 4.1

ตารางที่ 4.1 ตัวอย่างข้อมูลเข้า

อายุ	เพศ	กรุ๊ปเลือด	ความดัน	ผลการวินิจฉัย
50	ญ	O	125	เป็น
35	ญ	AB	80	เป็น
42	ช	A	90	ไม่เป็น
...	...	...	...	...

ก็จะต้องมีข้อมูลที่บอกว่าการทดสอบแอททริบิวต์ต่าง ๆ มีค่าใช้จ่ายเท่าใด เช่น การตรวจกรุ๊ปเลือดและการวัดความดันต้องเสียค่าใช้จ่ายเท่าใด ส่วนแอททริบิวต์อายุและเพศ อาจไม่ต้องเสียค่าใช้จ่ายในการวัด เป็นต้น

นอกจากนั้นข้อมูลที่นำเข้าควรจะมีการเตรียมก่อน (Clean Data) สำหรับการจัดการกับข้อมูลที่ไม่ครบ (missing data) นั้น ถ้าแอททริบิวต์นั้นเป็นแบบ Discrete (คือมีลักษณะเป็นตัวหนังสือ เช่น เป็นเพศ ชาย หรือ หญิง เป็นต้น) ก็ไม่จำเป็นต้องจัดการ สามารถนำเข้าได้เลย โดยอาจจะแปลงเป็นคำว่า "ไม่ทราบค่า" แทนก็ได้ แต่ถ้าแอททริบิวต์นั้นเป็นค่าต่อเนื่อง (เป็นตัวเลข) ก็จะต้องมีการจัดการหาตัวที่ใช้แทนค่านั้นก่อน ซึ่งการจัดการกับข้อมูลที่สูญหายนั้นมีหลายวิธี เช่น การแทนด้วยค่ามัธยฐาน, ค่าเฉลี่ย หรือ ค่าฐานนิยม เป็นต้น แต่ก็ต้องพิจารณาให้ดีเพราะถ้าทำการเปลี่ยนแปลงข้อมูลมากเกินไปอาจทำให้รูปแบบของข้อมูลเปลี่ยนไปได้ และถ้าแอททริบิวต์นั้นมีข้อมูลที่ไม่ทราบค่ามากจนถึงระดับที่กำหนด ก็อาจพิจารณาตัดแอททริบิวต์นั้นทิ้ง จะดีกว่าการใส่ค่าอื่นเข้าไป

บทที่ 5

การพัฒนาระบบงาน

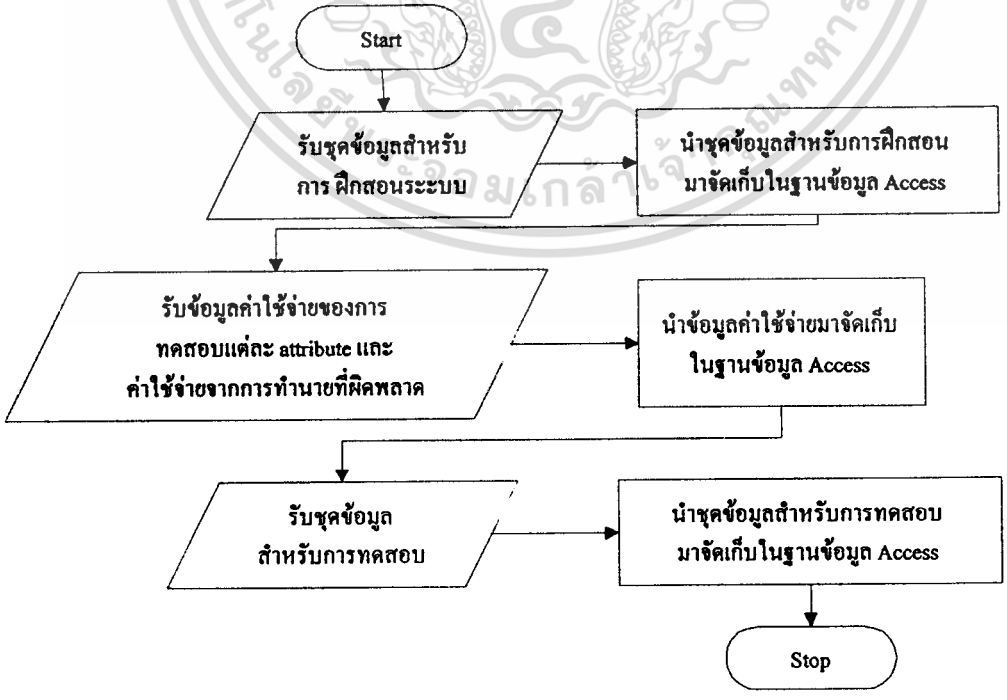
5.1 การทำงานของโปรแกรม

จากการออกแบบระบบที่ผ่านมา จึงพิจารณาแบ่งส่วนการทำงานของโปรแกรมออกเป็น 4 ส่วนหลัก ๆ คือ ส่วนของการนำข้อมูลเข้า ส่วนของกระบวนการเจเนติก ส่วนของกระบวนการสร้างดิสชั่นทรี และส่วนของการนำผลที่ได้มาแสดงต่อผู้ใช้ โดยใช้ Microsoft Visual Basic เวอร์ชัน 6.0 ในการพัฒนาโปรแกรม นอกจากนั้นยังนำ Microsoft Access 97 มาใช้เป็นฐานข้อมูลเพื่อใช้เก็บข้อมูลชั่วคราวระหว่างการทำงานของโปรแกรมด้วย

5.2 การพัฒนาโปรแกรม

5.2.1 โปรแกรมในส่วนของการนำข้อมูลเข้า

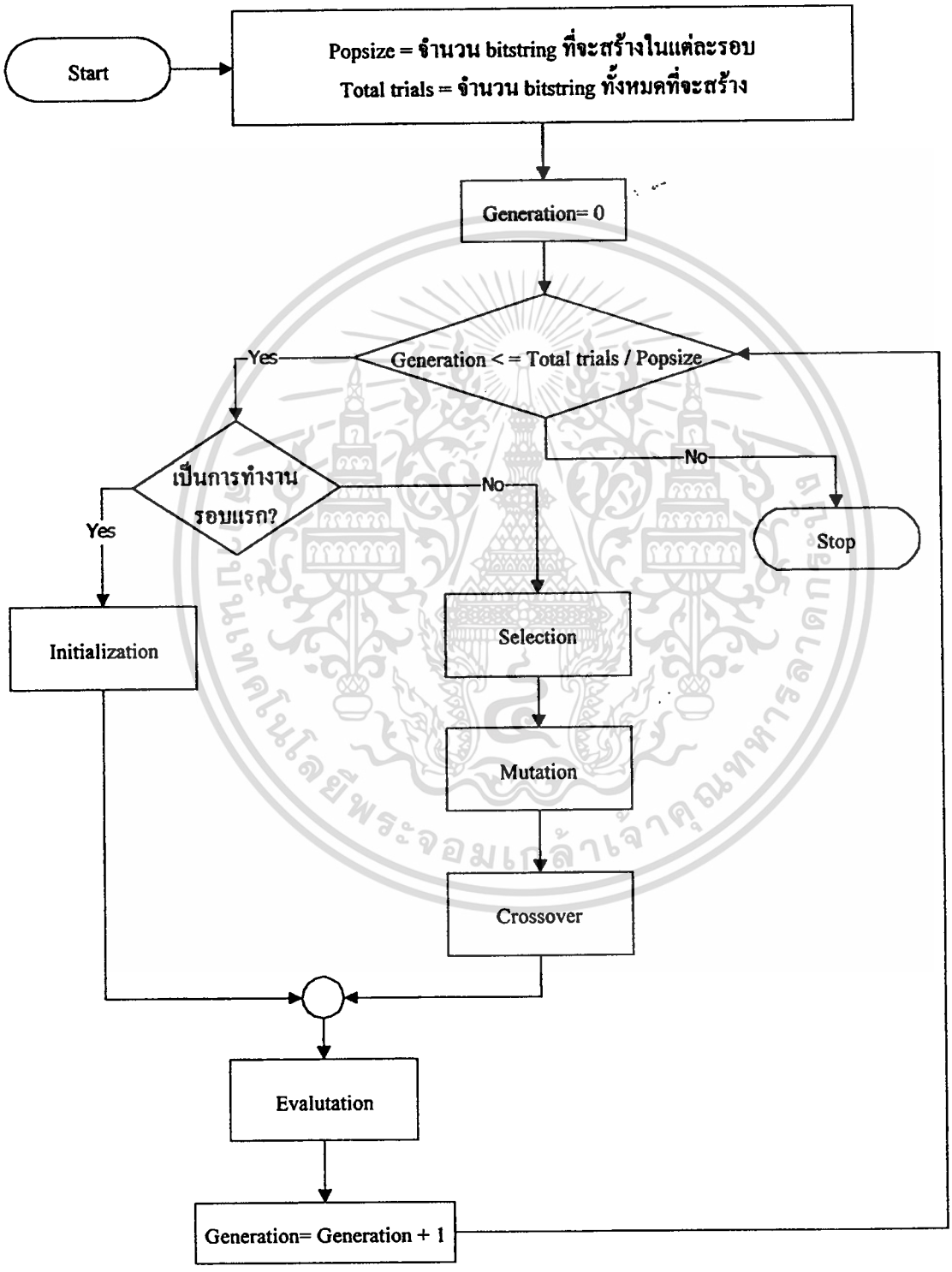
มีการทำงานหลัก ๆ ดังแสดงใน Flowchart รูปที่ 5.1



รูปที่ 5.1 การนำข้อมูลเข้า

5.2.2 โปรแกรมในส่วนของกระบวนการเจเนติก

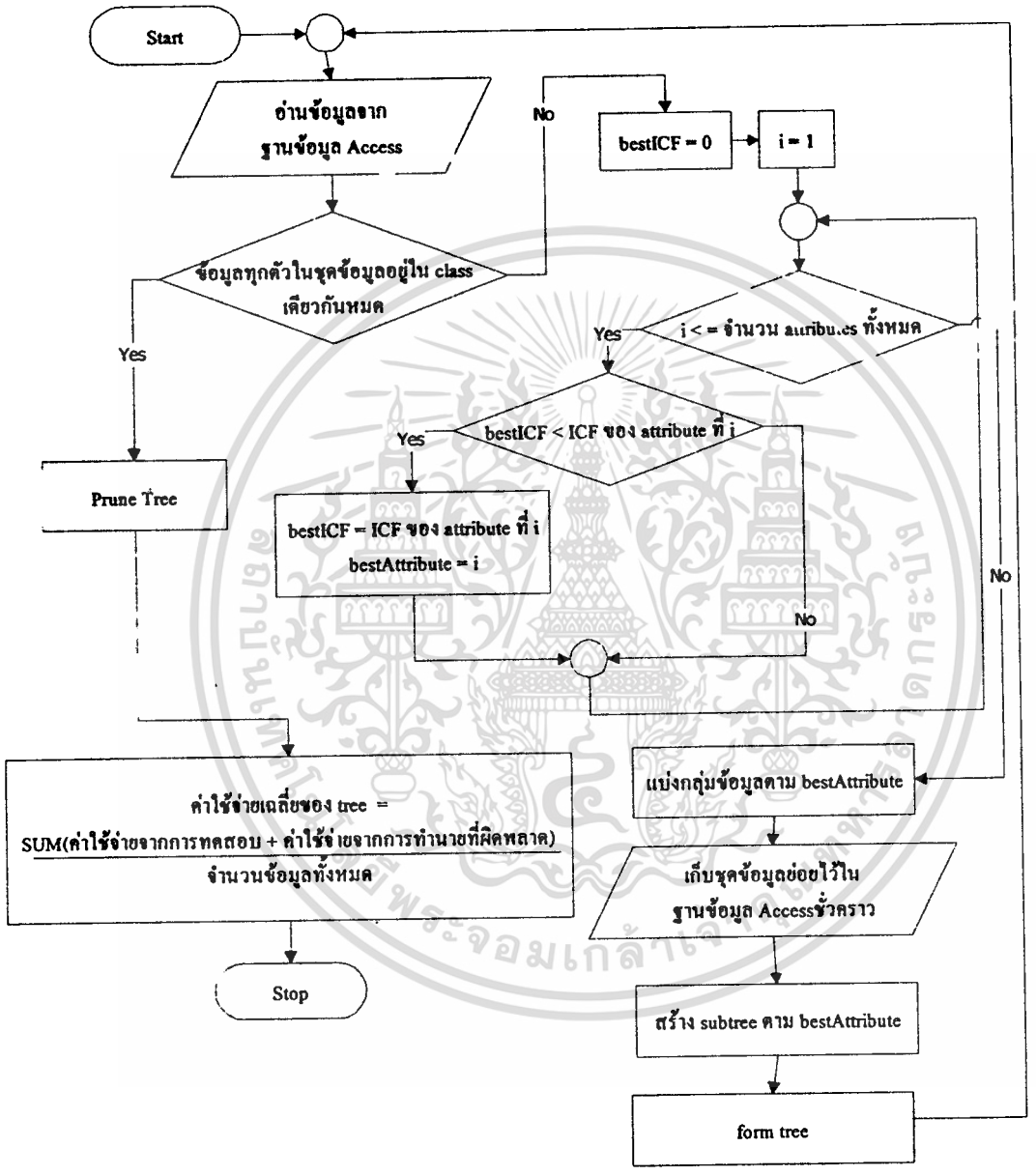
มีการทำงานหลัก ๆ ดังแสดงใน Flowchart รูปที่ 5.2



รูปที่ 5.2 กระบวนการเจเนติก

5.2.3 โปรแกรมในส่วนของกระบวนการสร้างดิสิชั่นทรี

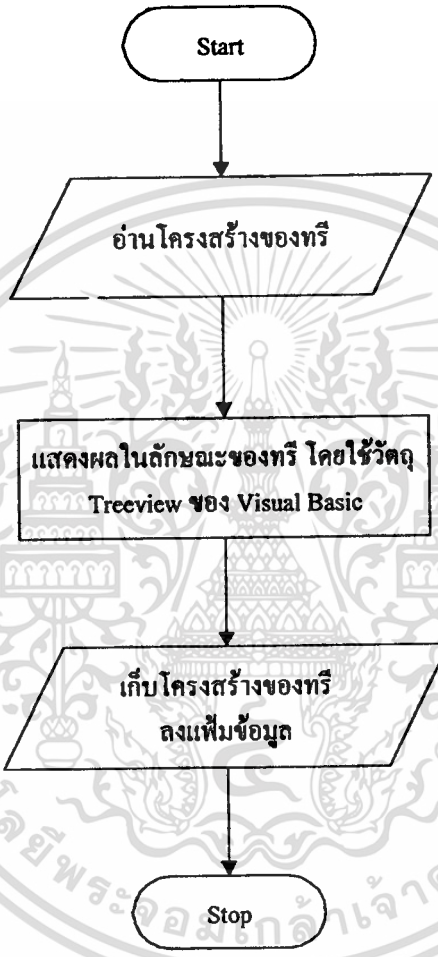
มีการทำงานหลัก ๆ ดังแสดงใน Flowchart รูปที่ 5.3



รูปที่ 5.3 กระบวนการสร้างดิสิชั่นทรี

5.2.4 โปรแกรมในส่วนของการแสดงผลต่อผู้ใช้

มีการทำงานหลัก ๆ ดังแสดงใน Flowchart รูปที่ 5.4

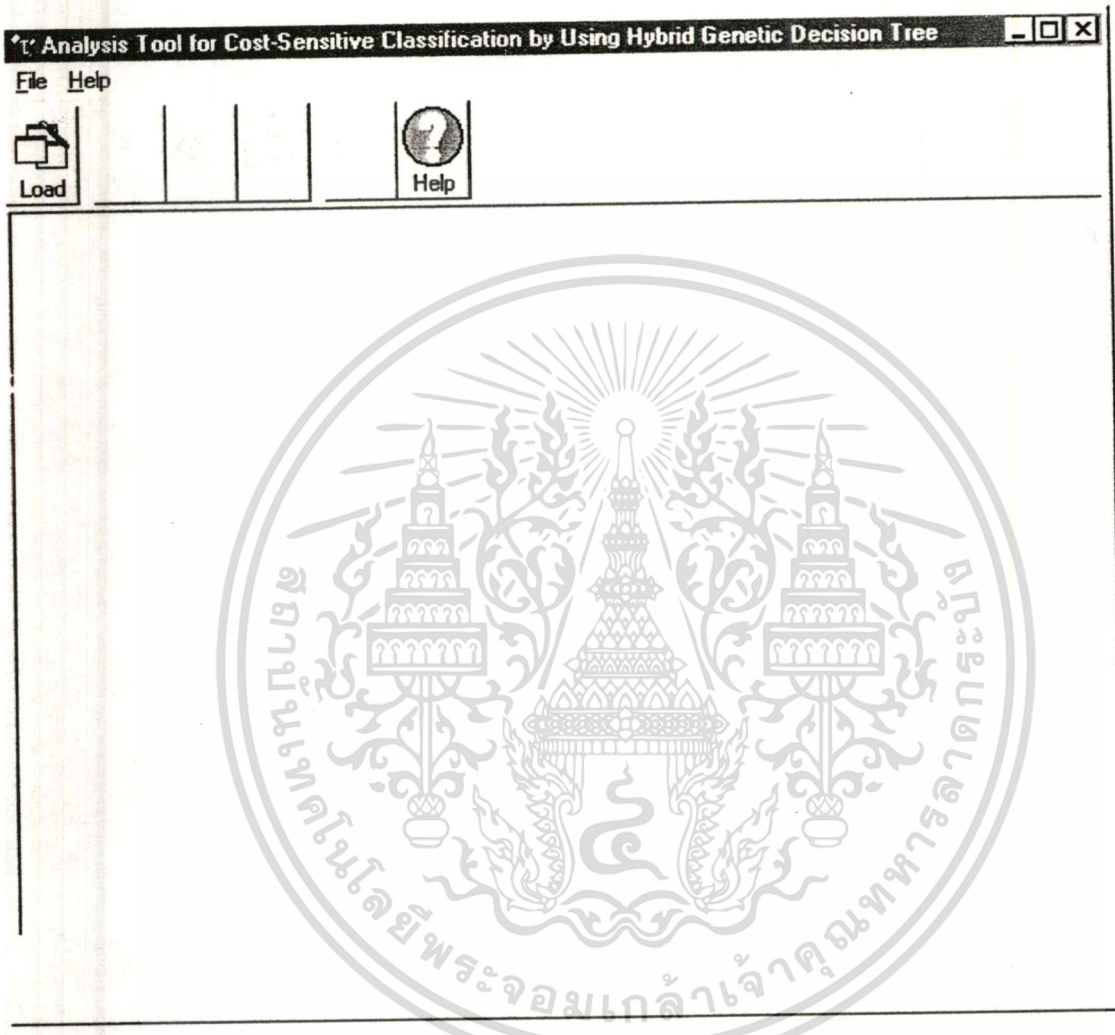


รูปที่ 5.4 การแสดงผลต่อผู้ใช้

5.3 การทดสอบโปรแกรม

ก่อนที่จะนำไปใช้งานจริงจะต้องมีการทดสอบการทำงานของโปรแกรมว่าสามารถทำงานได้จริงตามที่ออกแบบไว้หรือไม่ โดยระหว่างที่ได้ทำการทดสอบได้มีการแก้ไขปรับปรุงข้อผิดพลาดต่าง ๆ ควบคู่กันไปด้วย ซึ่งสามารถอธิบายวิธีการดำเนินการอย่างคร่าว ๆ ได้ดังนี้

การทดสอบความถูกต้องของขั้นตอนการทำงานของโปรแกรม เริ่มด้วยการสั่ง run โปรแกรม จะพบหน้าต่างหลักของโปรแกรมดังแสดงในรูปที่ 5.5



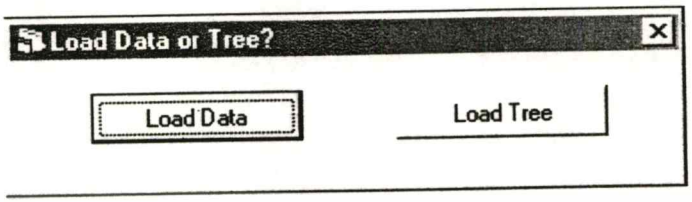
รูปที่ 5.5 หน้าต่างหลักของโปรแกรม

เมื่อเปิดโปรแกรมขึ้นมาตอนแรก ปุ่มและเมนูที่เลือกได้จะมีเพียงแค่การสั่งให้รับข้อมูลเข้า หรือการขอความช่วยเหลือเท่านั้น

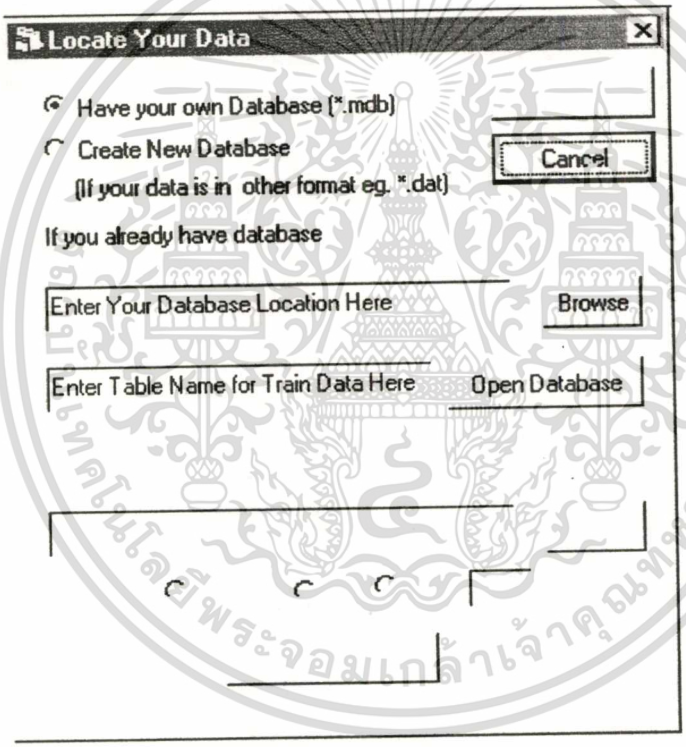
เมื่อผู้ใช้สั่งให้รับข้อมูลเข้าจะปรากฏหน้าต่างให้เลือกว่า จะทำการรับข้อมูลเข้าเพื่อทำการสร้างดิสชั่นทรี หรือจะเรียกข้อมูลโครงสร้างทรีที่เคยเก็บไว้มาแสดงผล ดังแสดงในรูปที่ 5.6

ถ้าผู้ใช้เลือกการรับข้อมูลเข้าเพื่อทำการสร้างดิสชั่นทรี ก็จะปรากฏหน้าต่างดังรูปที่ 5.7 เพื่อให้ผู้ใช้ระบุที่มาของข้อมูลที่จะรับเข้ามา โดยข้อมูลที่รับเข้ามามีได้ 2 แบบ คือ เป็นฐานข้อมูลหรือเป็นเท็กซ์ไฟล์ โดยถ้าผู้ใช้มีข้อมูลแบบเท็กซ์ไฟล์และเลือกปุ่ม “Create Database” ระบบก็จะทำการ

นำข้อมูลเท็กซ์ไฟล์ของผู้ใช้มาสร้างเป็นฐานข้อมูลใหม่ เพื่อใช้ในเก็บและเรียกดูข้อมูลระหว่างการสร้างดิสิชั่นทรี

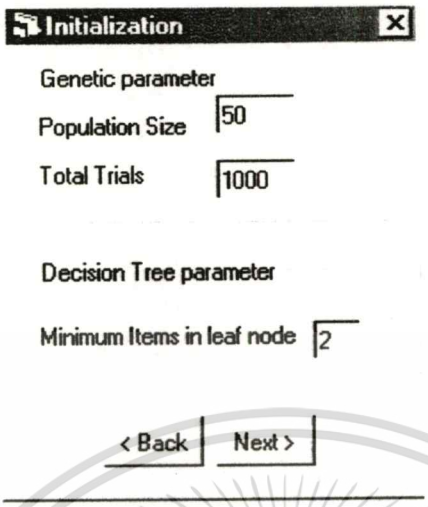


รูปที่ 5.6 หน้าต่างให้ผู้ใช้เลือกรับข้อมูลเพื่อสร้างทรี หรือเรียกดูโครงสร้างทรีที่เคยเก็บไว้



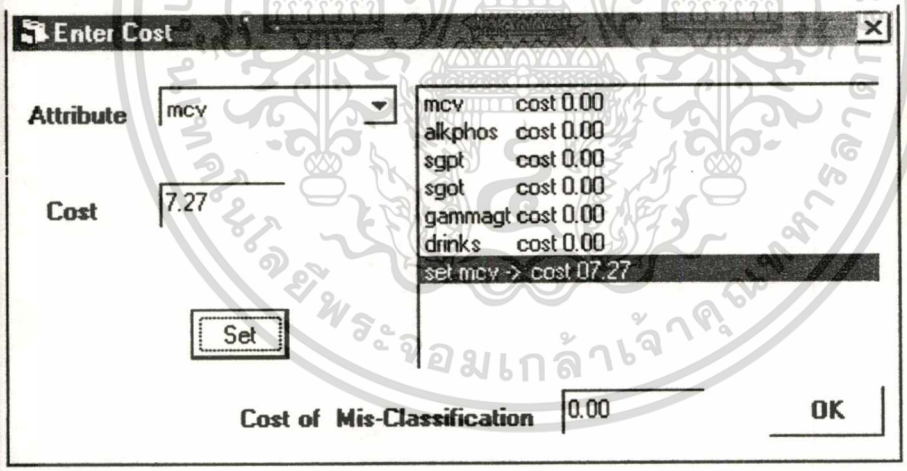
รูปที่ 5.7 หน้าต่างให้ผู้ใช้ใส่แหล่งที่มาของข้อมูลที่จะใช้ในการสร้างดิสิชั่นทรี

หลังจากนั้นจะเป็นหน้าต่างให้ผู้ใช้กำหนดพารามิเตอร์เริ่มต้นที่จะใช้ในการทำงานของกระบวนการเจเนติกและดิสิชั่นทรี โดย "Population size" จะเป็นการกำหนดว่าในแต่ละรอบจะให้เจเนติกทำการสร้าง bitstring มาทั้งหมดกี่ชุด และ "Total Trials" จะเป็นการกำหนดจำนวน bitstring ทั้งหมดที่ต้องการให้สร้างขึ้น ดังนั้นจำนวนรอบ (Generation) ในการทำงานของกระบวนการเจเนติกจะเท่ากับค่า Total trials หารด้วยค่า Population size อีกส่วนเป็นการกำหนดว่าจำนวนข้อมูลที่น้อยที่สุดที่จะอยู่ส่วนปลายสุดของทรี (leaf node) จะเป็นเท่าใด



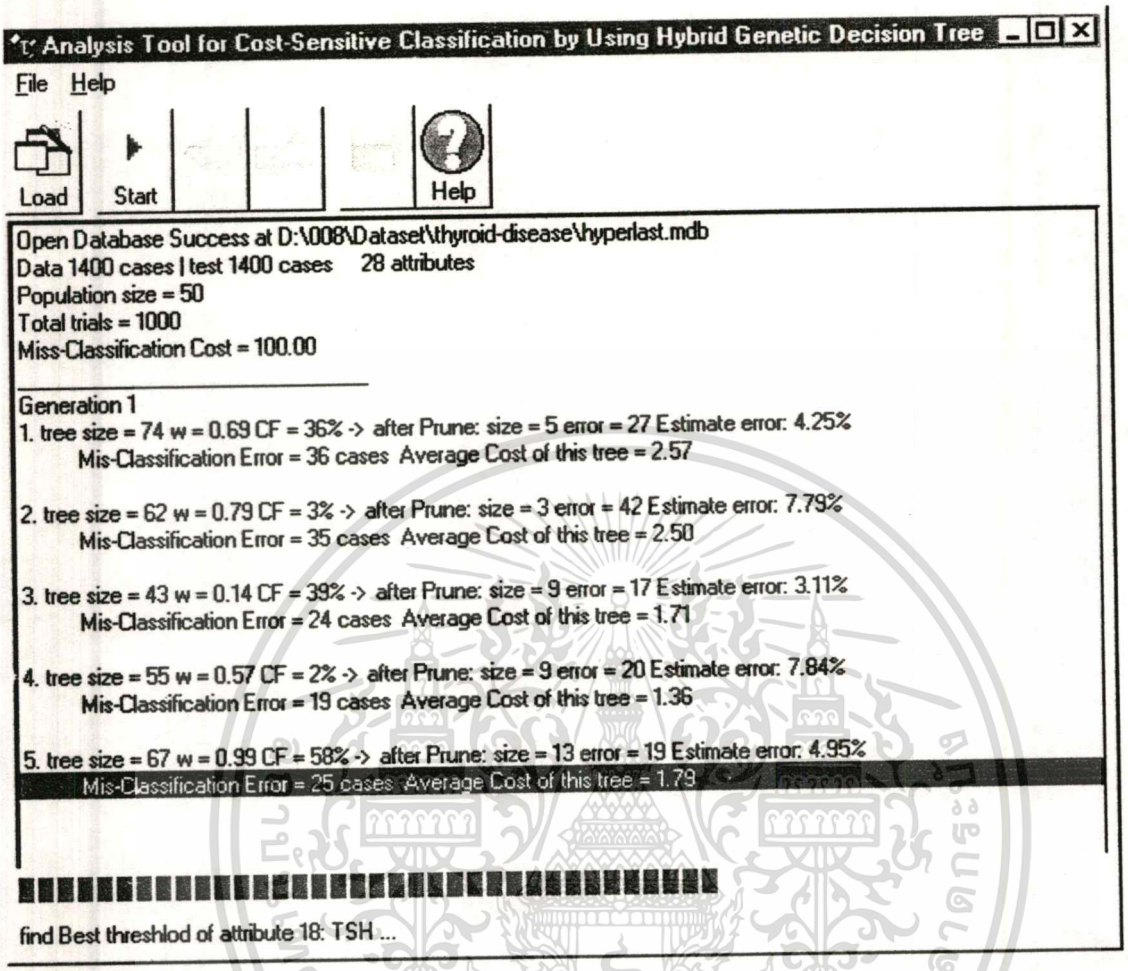
รูปที่ 5.8 หน้าต่างให้ผู้ใช้กำหนดพารามิเตอร์เริ่มต้นให้กับเจเนติกและดิสิชั่นทรี

จากนั้นจึงเป็นหน้างต่างให้ผู้ใช้ใส่ข้อมูลค่าใช้จ่ายในการทดสอบแต่ละแอททริบิวต์ และค่าใช้จ่ายสำหรับการจัดประเภทที่ผิดพลาดดังแสดงในรูปที่ 5.9



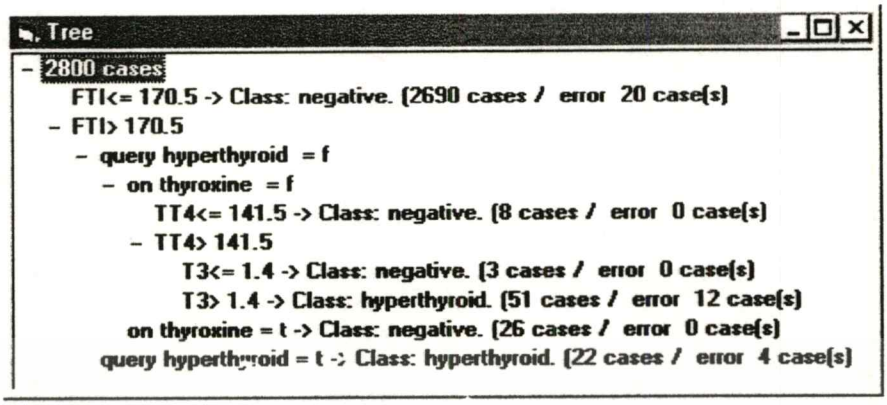
รูปที่ 5.9 หน้าต่างรับข้อมูลเกี่ยวกับค่าใช้จ่าย

เมื่อทำการรับข้อมูลเข้าเรียบร้อยแล้ว ก็จะแสดงผลที่ผู้ใช้ใส่ไปในหน้าต่างก่อนหน้านี้ที่พื้นที่แสดงผลในหน้าต่างหลักของระบบ จากนั้นผู้ใช้จะสามารถกดปุ่ม "Start" เพื่อสั่งให้เริ่มทำงานได้ โดยระหว่าที่โปรแกรมทำงานจะแสดงผลการทำงานที่พื้นที่แสดงผลดังแสดงในรูปที่ 5.10



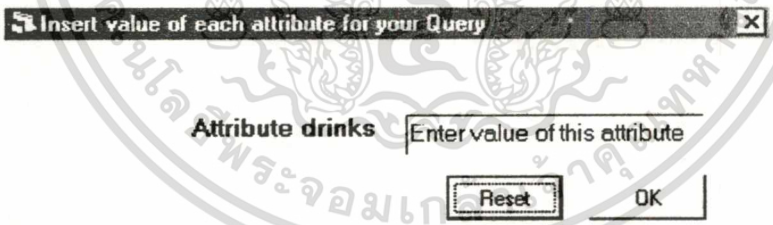
รูปที่ 5.10 หน้าต่างหลักของโปรแกรมหลังจากรับข้อมูลเข้าแล้ว

เมื่อโปรแกรมทำงานจนได้ผลลัพธ์เป็นดิสิชั่นทรีเรียบร้อยแล้ว จะแสดงผลเป็นโครงสร้างต้นไม้ดังแสดงในรูปที่ 5.11 โดยจะบอกว่าข้อมูลตกอยู่ในกิ่งใดเป็นจำนวนเท่าใดและข้อมูลที่ผิดพลาดเป็นจำนวนเท่าใด รวมทั้งบอกประเภท (class) ที่ข้อมูลส่วนใหญ่ในโหนดนั้นตกอยู่ โดยผู้ใช้สามารถเรียกดูโครงสร้างนี้ได้อีกครั้งโดยเลือกที่ปุ่ม "View" หรือเลือกที่เมนูย่อย View ในเมนู File และผู้ใช้สามารถบันทึกโครงสร้างทรีนี้เก็บไว้เพื่อเรียกดูภายหลังได้โดยการเลือกที่ปุ่ม "Save" ซึ่งเป็นเก็บเป็นแฟ้มข้อมูล 2 แฟ้มคือ Tree.tree และ Fldname.tree ไว้ในไดเรกทอรีที่ผู้ใช้กำหนด

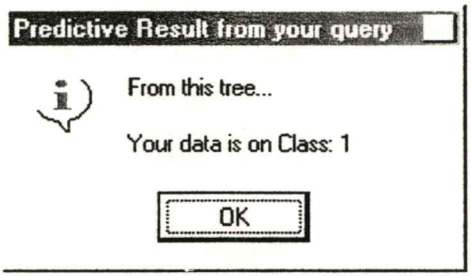


รูปที่ 5.11 ผลลัพธ์จากการทำงานของโปรแกรม

เมื่อผู้ใช้ต้องการจะเรียกโครงสร้างทรีนี้มาดูอีกครั้งก็ให้เลือกปุ่ม "Load" ในหน้าต่างหลัก และเลือกปุ่ม "Load Tree" ดังรูปที่ 5.6 จากนั้นจึงระบุเส้นทางที่โครงสร้างทรีถูกเก็บอยู่ ซึ่งเมื่อผู้ใช้ทำการเรียกโครงสร้างทรีขึ้นมาดูแล้ว ผู้ใช้ยังสามารถสอบถามเกี่ยวกับข้อมูลของผู้ใช้ว่าจะจัดอยู่ในประเภทใด โดยเลือกปุ่ม "Query" โดยหลังจากผู้เลือกปุ่ม "Query" แล้วจะมีหน้าต่างให้ผู้ใช้ใส่ข้อมูลในแต่ละแอททริบิวต์ของผู้ใช้ดังรูปที่ 5.12 โดยไม่ต้องใส่ข้อมูลทุกแอททริบิวต์ แต่จะไล่ไปตามทรีจากรากไปยังปลาย เมื่อถึงส่วนปลายของทรีซึ่งระบุประเภทที่คาดว่าข้อมูลจะจัดอยู่ ระบบก็จะแสดงผลการทำนายว่าข้อมูลควรจะจัดอยู่ในประเภทใดดังรูปที่ 5.13



รูปที่ 5.12 หน้าต่างให้ผู้ใช้ใส่ข้อมูลเพื่อสอบถามการจัดประเภทของข้อมูลนั้น



รูปที่ 5.13 หน้าต่างแสดงผลการทำนายประเภทที่ข้อมูลจะถูกจัดอยู่

บทที่ 6

บทสรุป

6.1 สรุปผลการศึกษา

เครื่องมือช่วยวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย โดยใช้วิธีการทางเจเนติกร่วมกับดิสิชั่นทรีที่ได้พัฒนาขึ้นมานั้น เป็นเครื่องมือที่ช่วยให้สามารถจัดประเภทข้อมูลได้โดยมีค่าใช้จ่ายทั้งจากการทดสอบและการจัดประเภทที่ผิดพลาดน้อยที่สุด

ในการศึกษานี้ได้ทำการสร้างโปรแกรมสำหรับช่วยวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย โดยใช้วิธีการทางเจเนติกร่วมกับดิสิชั่นทรี ซึ่งตัวโปรแกรมสามารถแบ่งขั้นตอนออกเป็น 4 ขั้นตอนหลัก ๆ คือ

- การนำข้อมูลเข้า
- การทำงานตามกระบวนการเจเนติกโดยใช้วิธีการ GENESIS
- การทำงานตามกระบวนการดิสิชั่นทรีโดยใช้หลักการของอัลกอริทึม EG2 มาปรับใช้ร่วมกับอัลกอริทึม C4.5 ของดิสิชั่นทรี เพื่อให้สามารถนำค่าใช้จ่ายมาเป็นพารามิเตอร์หนึ่งในการสร้างทรีด้วย
- การนำผลดิสิชั่นทรีที่ได้มาแสดงต่อผู้ใช้

ผลจากการพัฒนาโปรแกรมทำให้ได้เครื่องมือสำหรับช่วยวิเคราะห์จัดประเภทข้อมูลที่คำนึงถึงค่าใช้จ่าย ซึ่งผลลัพธ์ที่ได้ออกมานั้นผู้ใช้อาจนำมาประยุกต์ใช้ให้เกิดประโยชน์ เช่นนำดิสิชั่นทรีที่ได้มาไปสร้างเป็นกฎ เป็นต้น

6.2 ข้อเสนอแนะ

เนื่องจากระยะเวลาที่ใช้ในการพัฒนาระบบงานนี้ค่อนข้างจำกัด และการทดสอบการทำงานของโปรแกรมแต่ละครั้งต้องใช้เวลานานมาก ทำให้ระบบงานที่ได้ยังไม่ค่อยสมบูรณ์มากนัก โปรแกรมนี้ยังมีข้อจำกัดอยู่หลายประการที่ควรจะต้องปรับปรุงแก้ไข เพื่อให้โปรแกรมมีความยืดหยุ่นเหมาะสมกับการนำไปใช้ประโยชน์ สิ่งที่ควรแก้ไขสำหรับผู้ที่สนใจจะพัฒนาต่อไปมีดังนี้

ขั้นตอนการสร้างดิสิชั่นทรีใช้เวลาค่อนข้างมากสำหรับข้อมูลปริมาณมาก และโดยเฉพาะถ้ามีแอททริบิวต์ที่มีค่าเป็นแบบต่อเนื่อง (continuous) มาก เนื่องจากในการหาค่า threshold ว่าจะใช้ตัว

เลขใดเป็นตัวแบ่งในการทดสอบแอททริบิวต์นั้น จะต้องนำทุกค่าในแอททริบิวต์มาคำนวณหาค่าที่ดีที่สุด

ข้อมูลที่นำมาใช้ในการทดสอบโปรแกรม อาจมีจำนวนไม่มากพอที่จะใช้วัดประสิทธิภาพการทำงานจริงได้ เนื่องจากลักษณะข้อมูลที่ต้องการใช้ยังไม่ค่อยมีการจัดเก็บในเมืองไทย

นอกจากนั้นผู้ที่สนจะพัฒนาต่อควรดำเนินการศึกษาเสียแต่เนิ่น ๆ เพื่อจะได้มีระยะเวลาในการทดสอบและปรับปรุงแก้ไขมากขึ้น และในส่วนของการแสดงผลนั้น อาจพัฒนาให้แสดงผลในรูปของกฎด้วย หรือให้สามารถเก็บโครงสร้างดิสิชั่นทรีที่ดีที่สุดไว้ในรูปของแฟ้มข้อมูลเพื่อให้สามารถเรียกใช้ได้อีกก็จะเป็นประโยชน์ต่อผู้ใช้มากขึ้น

บรรณานุกรม

Berson, A. and Smith, S. J. 1997. "Data warehousing, data mining, and OLAP." New York: McGraw-Hill.

Grefenstette, J. J. 1990. "A User's Guide to GENESIS Version 5.0." [Online]. Available: http://ftp.aic.nrl.navy.mil/galist/src/.

Quinlan, J. R. 1993. "C4.5: Programs for Machine Learning." California: Morgan Kaufmann Publishers, Inc.

Quinlan, J. R. 1996. "Improved Use of Continuous Attributes in C4.5." 77-90. In *Journal of Artificial Intelligence Research 4.*

Turney, P. D. 1995. "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm." [Online]. Available: http://extractor.iit.nrc.ca/cgi-bin/jair-abstract.pl?turney95a.

Turney, P.D. 2000. "Types of cost in inductive concept learning". [Online]. Available: http://extractor.iit.nrc.ca/bibliographies/cost-sensitive.html.

ประวัติผู้เขียน

ชื่อ นามสกุล	นางสาวแววดาว อดุลย์พิจิตร
สถานที่เกิด	กรุงเทพฯ
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยธรรมศาสตร์
ปีการศึกษาที่สำเร็จการศึกษา	ปีการศึกษา 2540