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

J. Major & J. Mangano (1993). Selecting among rules induced from a hurricane database. In *Proceedings of the AAAI-93 Workshop on KDD*.

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

J. Ortega & D. Fisher (1995). Flexibly exploiting prior knowledge in empirical learning. *IJCAI*.

M. Pazzani & D. Kibler (1992). The utility of knowledge in inductive learning. *Machine Learning*, 9.

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

G. Piatetsky-Shapiro & C.J. Matheus (1994). The interestingness of deviations. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*.

G. Piatetsky-Shapiro, C.J. Matheus, P. Smyth, & R. Uthurusamy (1994). KDD-93:Progress and challenges. *AI Magazine*, Fall, 77-87.

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.

A. Silberschatz & A. Tuzhilin (1995). On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada.

A. Silberschatz & A. Tuzhilin (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6).

E. Suzuki (1997). Autonomous discovery of reliable exception rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.*

K. Wang, S.H.W. Tay, & B. Liu (1998). Interestingness-based interval merger for numeric association rules. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining.*

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

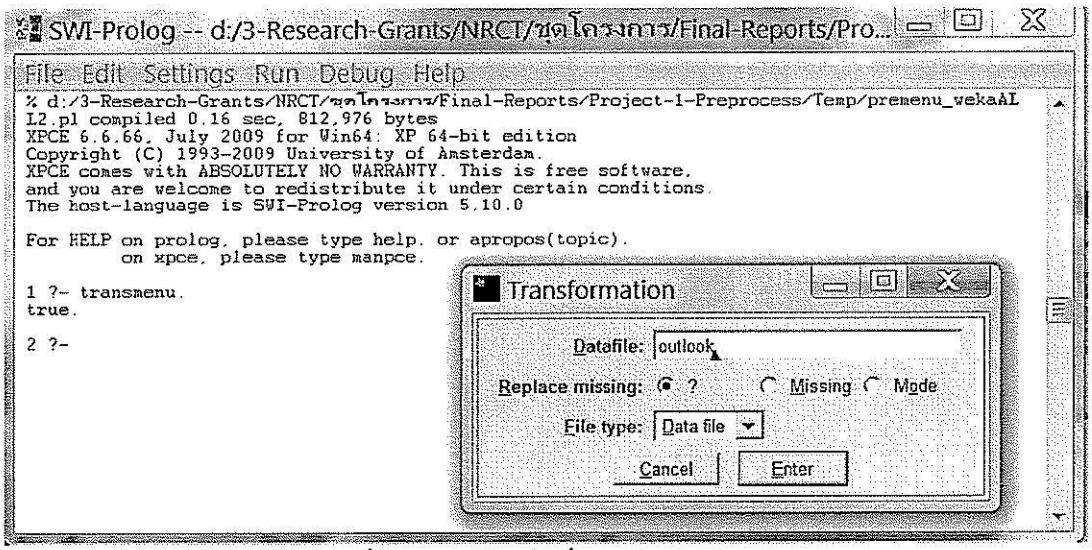
ภาคผนวก ก

คู่มือการใช้งานระบบ SUT Miner

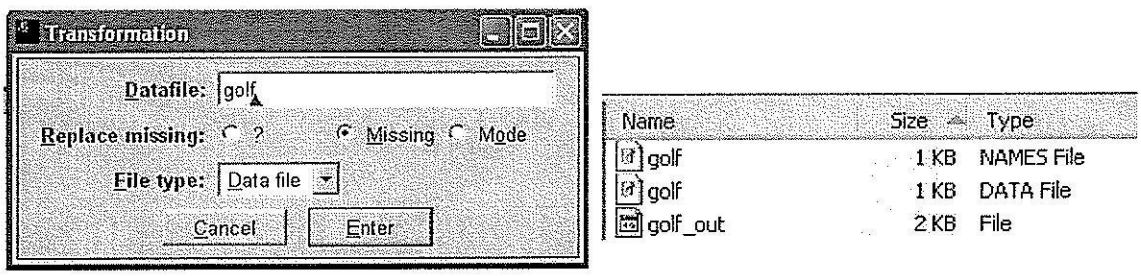
(1) การนำเข้าข้อมูล

เริ่มต้นเรียกใช้ระบบ SUT Miner ด้วยการเรียกใช้คำสั่ง transmenu ใน SWI Prolog (การพิมพ์คำสั่งในภาษาโปรล็อกจะต้องจบทุกคำสั่งด้วยเครื่องหมายจุด เช่น "transmenu.") เพื่อแสดงจอภาพเริ่มต้นของการแปลงรูปแบบแฟ้มข้อมูล (Data transformation) ชื่อไฟล์ข้อมูลในกรอบ Datafile ดังแสดงในรูปที่ ก1 เป็นค่า default ผู้ใช้สามารถลบชื่อไฟล์นี้แล้วพิมพ์ชื่อไฟล์ที่ถูกต้องโดยไม่ต้องระบุส่วนขยายลงในกรอบ เช่น พิมพ์ชื่อแฟ้มข้อมูล golf ดังรูปที่ ก2 จากนั้นคลิกที่ปุ่ม Enter ระบบจะสร้างแฟ้มข้อมูลในรูปแบบคลอสของภาษาโปรล็อก (Horn clauses) บันทึกไว้ในไฟล์ชื่อ golf_out

จากภาพจะสังเกตได้ว่าพร้อมกับการระบุชื่อไฟล์เพื่อแปลงรูปแบบแฟ้มข้อมูล ผู้ใช้จะสามารถเลือกวิธีการจัดการกับกรณีข้อมูลสูญหายได้ด้วย ซึ่งจะเลือกวิธีจัดการได้สามรูปแบบ คือ แทนด้วย '?' หรือแทนด้วยข้อความ 'missing' หรือแทนด้วยค่าส่วนใหญ่ของข้อมูล (mode)



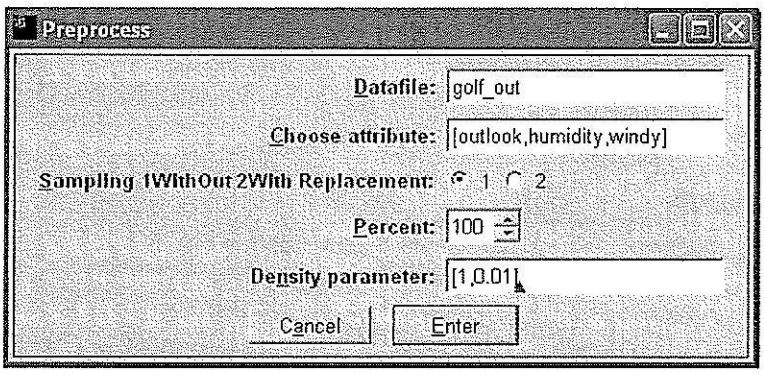
รูปที่ ก1. การใช้คำสั่งเพื่อนำเข้าแฟ้มข้อมูล



รูปที่ ก2. การระบุชื่อแฟ้มข้อมูลเข้าและไฟล์ใหม่ที่เกิดขึ้นหลังการแปลงรูปแบบข้อมูล

(2) การคัดเลือกแอททริบิวต์และการสุ่มข้อมูล

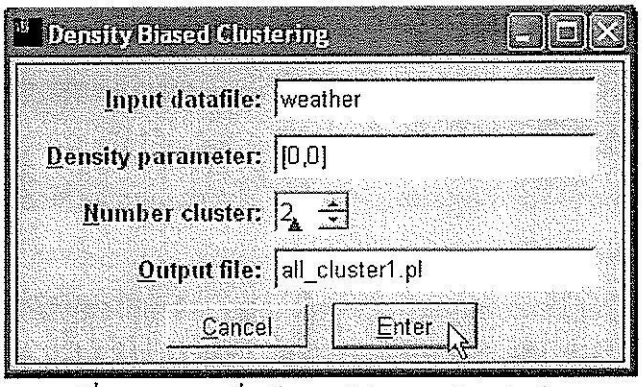
ในขั้นตอนของการทำ pre-data mining ถ้าผู้ใช้ต้องการคัดเลือกแอททริบิวต์ หรือลดขนาดข้อมูลด้วยการสุ่มเลือกเฉพาะข้อมูลตัวแทน สามารถทำได้ด้วยการพิมพ์คำสั่ง "premenu" ที่จอภาพ SWI Prolog จะปรากฏจอภาพดังรูป ก3 ในหน้าจอโต้ตอบกับผู้ใช้จะปรากฏกรอบข้อความให้ผู้ใช้พิมพ์ชื่อไฟล์ที่มีรูปแบบข้อมูลเป็น Horn clauses ซึ่งจากตัวอย่างก่อนหน้านี้ไฟล์ที่ใช้จะเป็นชื่อ golf_out ในบรรทัดต่อมาจะเป็นส่วนที่ให้ผู้ใช้ระบุชื่อแอททริบิวต์ที่ต้องการ รายชื่อแอททริบิวต์จะพิมพ์อยู่ภายในวงเล็บ [ ] ซึ่งแทนโครงสร้างลิสต์ในภาษาโปรล็อก จากภาพระบุชื่อแอททริบิวต์ [outlook, humidity, windy] โดยตัดทิ้งแอททริบิวต์ temperature กรอบข้อความในบรรทัดที่สามและสี่ของจอภาพ เป็นการระบุวิธีสุ่มข้อมูลและปริมาณข้อมูลที่ต้องการ ถ้าผู้ใช้ต้องการทำ feature selection อย่างเดียวโดยไม่ต้องการสุ่มข้อมูลสามารถระบุค่าเปอร์เซนต์เป็น 100 ผลลัพธ์ที่ได้จะเป็นข้อมูลใหม่อยู่ในไฟล์ ple.pl ซึ่งสามารถนำไปใช้ต่อในขั้นตอนการทำเหมืองข้อมูลแบบจัดกลุ่มหรือแบบจำแนก



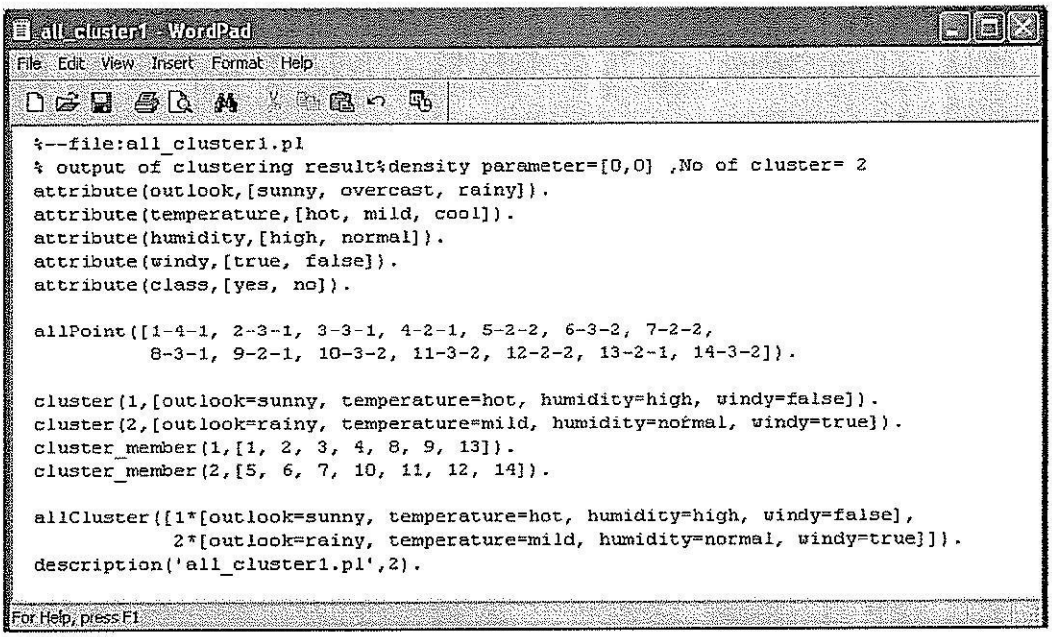
รูปที่ ก3. จอภาพส่วนคัดเลือกแอททริบิวต์และสุ่มข้อมูล

(3) การทำเหมืองข้อมูลแบบจัดกลุ่ม

การเรียกใช้โปรแกรมจัดกลุ่มข้อมูล ผู้ใช้พิมพ์คำสั่ง "dens_clust_menu" ที่จอภาพ SWI Prolog จะปรากฏจอภาพดังรูป ก4 เพื่อให้ผู้ใช้ระบุชื่อแฟ้มข้อมูล (ที่ข้อมูลถูกแปลงให้อยู่ในรูปแบบคลอสของภาษาโปรล็อกจากขั้นตอนการเตรียมข้อมูลแล้ว) ระบุจำนวนกลุ่ม หรือ Number cluster และระบุชื่อแฟ้มข้อมูลที่จะบันทึกผลลัพธ์ของการจัดกลุ่มข้อมูล ในรูประบุค่า density parameter เป็น [0,0] หมายถึงการจัดกลุ่มข้อมูลไม่ต้องให้ค่าน้ำหนักเบี่ยงเบนตามความหนาแน่นของข้อมูล ผลลัพธ์ของการจัดกลุ่มข้อมูลแสดงได้ดังรูปที่ ก5



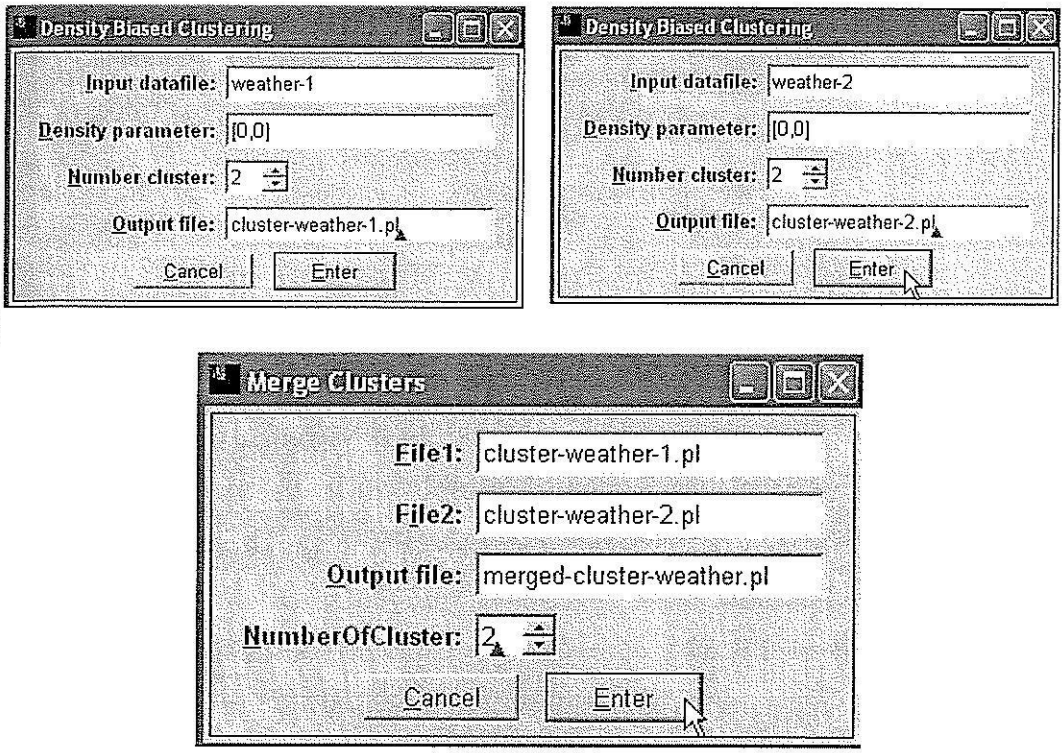
รูปที่ ก4. จอภาพเริ่มต้นของโปรแกรมจัดกลุ่มข้อมูล



รูปที่ ก5. ผลลัพธ์ของการจัดกลุ่มข้อมูลเมื่อระบุจำนวนกลุ่มเป็นสองกลุ่ม

ในกรณีของการจัดกลุ่มข้อมูลแบบเพิ่มพูน แนวคิดเกี่ยวกับการจัดกลุ่มข้อมูลแบบเพิ่มพูน หรือ incremental clustering เกิดจากการพยายามจัดกลุ่มกับข้อมูลที่มีขนาดใหญ่มาก ทำให้ต้องแบ่งจัดข้อมูลเป็นกลุ่มย่อยๆ จากนั้นจึงจะรวมข้อมูลในกลุ่มย่อย (merge clusters) ให้เป็นกลุ่มใหญ่ ในงานวิจัยนี้ใช้วิธีการรวมกลุ่มหรือคลัสเตอร์ โดยพิจารณาแต่ละ cluster mean ให้เป็นเสมือนหนึ่งรายการข้อมูล จากนั้น merge cluster means ให้ได้ค่า means ใหม่ ผู้ใช้จะต้องแบ่งข้อมูลเข้าที่มีขนาดใหญ่ให้อยู่ในหลายไฟล์ข้อมูล จากนั้นเรียกเมนูจัดกลุ่มข้อมูลเหมือนในขั้นตอนก่อนหน้านี้เพื่อจัดกลุ่มข้อมูลในแต่ละไฟล์ย่อย

เมื่อได้ผลการจัดกลุ่มข้อมูลย่อยหลายกลุ่มแล้ว จะเป็นขั้นตอนการรวม cluster ซึ่งจะเริ่มด้วยการเรียกคำสั่ง "merge_clust_menu" ที่จอภาพ SWI Prolog รูปที่ ก6 แสดงการสั่งจัดกลุ่มกับข้อมูลย่อยสองชุด คือ weather-1 และ weather-2 จากนั้นระบุให้บันทึกผลลัพธ์ไว้ในไฟล์ cluster-weather-1.pl และ cluster-weather-2.pl ตามลำดับ ในจอภาพที่ระบุการ merge clusters จึงระบุชื่อไฟล์สองไฟล์นี้เป็นข้อมูลเข้า พร้อมทั้งระบุชื่อไฟล์ที่จะบันทึกผลลัพธ์เป็น merged-cluster-weather.pl

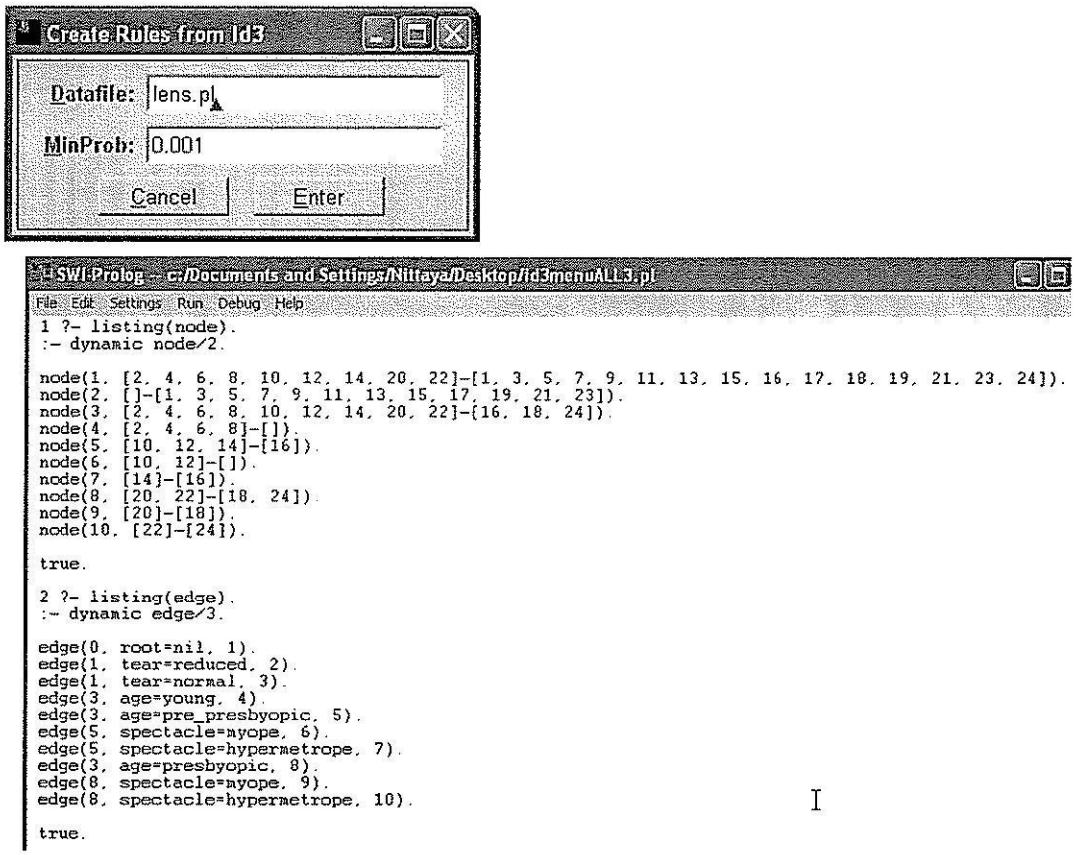


รูปที่ ก6. การจัดกลุ่มข้อมูลแบบเพิ่มพูนกับข้อมูลในสองกลุ่มย่อย

(4) การทำเหมืองข้อมูลแบบจำแนก

การเรียกใช้โปรแกรมทำเหมืองข้อมูลแบบการจำแนก เริ่มต้นด้วยการใช้คำสั่ง "id3menu" จากจอภาพ SWI Prolog เมื่อกด Enter จะปรากฏจอภาพโต้ตอบกับผู้ใช้ เพื่อให้ผู้ใช้ระบุค่าความน่าจะเป็นขั้นต่ำเพื่อเป็นเกณฑ์ในการคัดเลือกโมเดลที่ใช้ประโยชน์ได้ ถ้าผู้ใช้ไม่ระบุค่านี้ ระบบจะกำหนดให้เป็น 0.0 เมื่อคลิกที่ปุ่ม Enter โปรแกรมจะสร้างโมเดลข้อมูลในลักษณะของต้นไม้ตัดสินใจ โมเดล

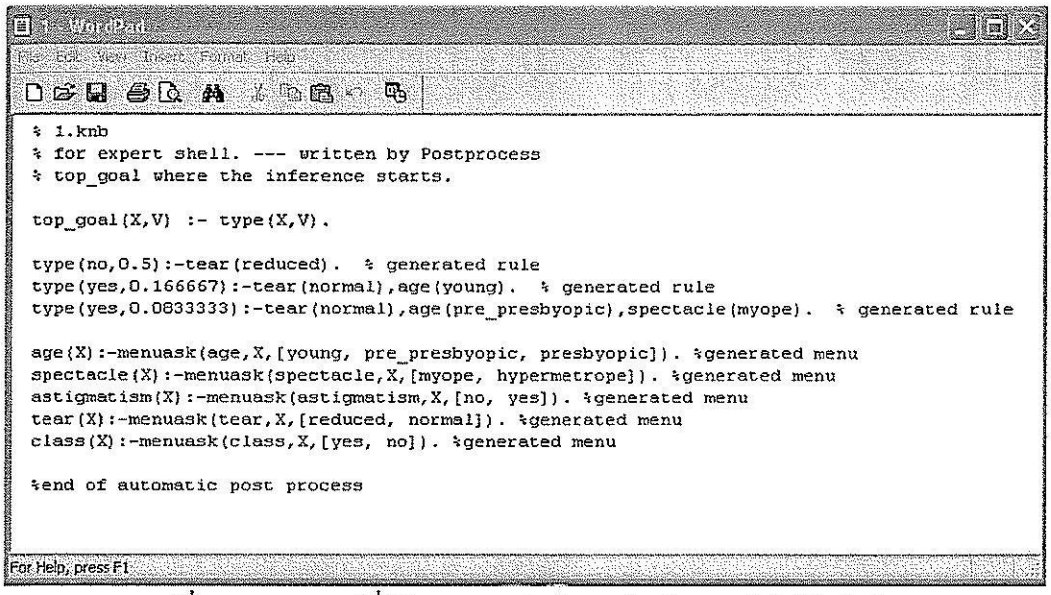
ข้อมูลนี้สามารถเรียกดูได้จากหน้าจอของ SWI Prolog ดังรูปที่ ก7 ด้วยการใช้คำสั่ง listing(node) และ listing(edge) ตามลำดับ



รูปที่ ก7. โมเดลข้อมูลในลักษณะของต้นไม้ตัดสินใจเมื่อเรียกใช้โปรแกรมทำเหมืองข้อมูลแบบจำแนก

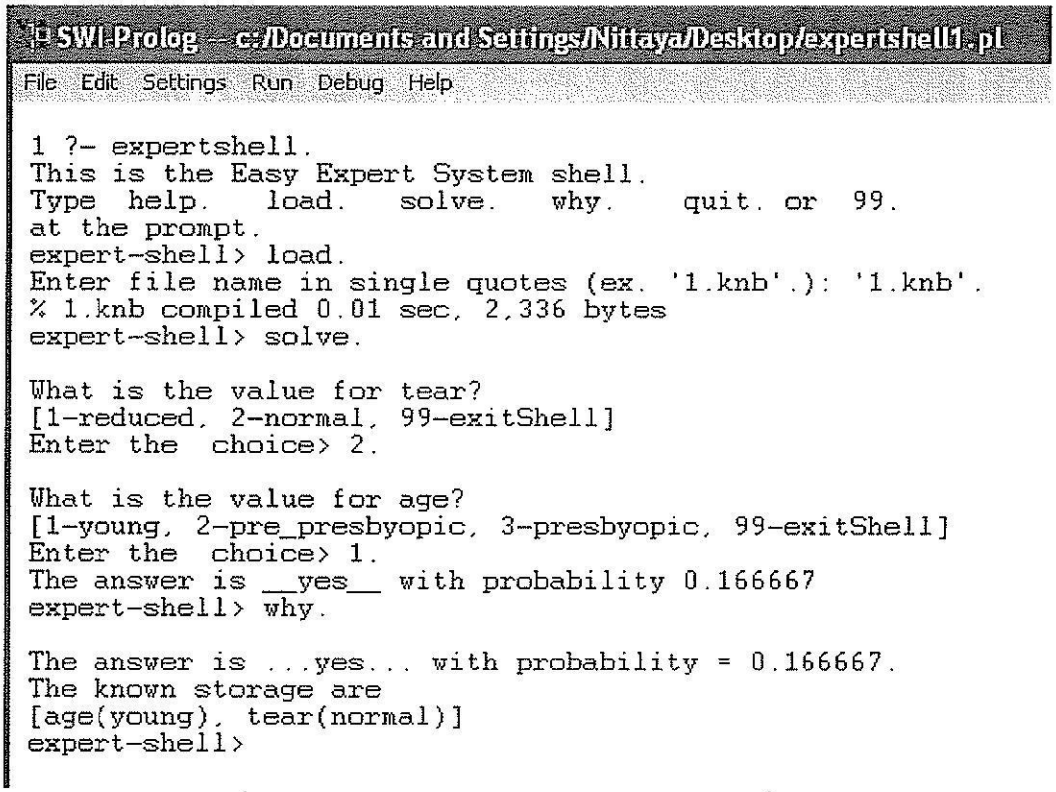
(5) การประมวลผลหลังการทำเหมืองข้อมูลแบบจำแนก

ผู้ใช้เรียกโปรแกรมทำเหมืองข้อมูลแบบจำแนกด้วยการพิมพ์คำสั่ง "id3menu" ที่จอภาพ SWI Prolog โมเดลที่ได้จากการทำเหมืองข้อมูลแบบจำแนกจะอยู่ในลักษณะของต้นไม้ตัดสินใจ และจากโครงสร้างต้นไม้ที่บันทึกอยู่ในเพรดิเคต node และ edge โปรแกรม post-data mining สามารถแปลงเป็นฐานความรู้ได้โดยอัตโนมัติ จากนั้นจะบันทึกผลลัพธ์ที่ได้ลงในไฟล์ชื่อ '1.knb' ซึ่งจากตัวอย่างข้อมูล lens.pl เมื่อแปลงกฎเสร็จจะได้ฐานข้อมูลดังรูปที่ ก8



รูปที่ ก8. ฐานความรู้ที่ได้จากการแปลงโมเดลในลักษณะต้นไม้ตัดสินใจ

เมื่อต้องการดูโมเดลจากการทำเหมืองข้อมูลแบบจำแนกผ่านระบบผู้เชี่ยวชาญ ผู้ใช้สามารถสั่งงานได้ด้วยการรันโปรแกรม expertshell1.pl จากนั้นที่จอภาพ SWI Prolog ให้พิมพ์คำสั่ง "expertshell" (ดังตัวอย่างในรูป ก9) เมื่อโปรแกรมระบบผู้เชี่ยวชาญเริ่มทำงาน ที่จอภาพ SWI Prolog ตำแหน่งต้นบรรทัดจะปรากฏข้อความ expert-shell> เพื่อเตรียมรับคำสั่งจากผู้ใช้ คำสั่งแรกของการใช้งานคือคำสั่ง load เพื่อเรียกใช้ไฟล์ฐานความรู้ที่เกี่ยวข้อง ซึ่งในตัวอย่างนี้ใช้ไฟล์ '1.knb' จากนั้นใช้คำสั่ง solve โปรแกรมจะเริ่มถามข้อมูลต่างๆจากผู้ใช้ เมื่อได้ข้อมูลที่ต้องการเพียงพอแล้ว จะแสดงคำแนะนำให้ผู้ใช้ทราบ พร้อมทั้งค่าความน่าจะเป็นเพื่อให้ผู้ใช้ทราบว่าจะเชื่อมั่นในคำแนะนำนั้นได้มากน้อยเพียงใด และถ้าผู้ใช้ต้องการคำอธิบายประกอบคำแนะนำ สามารถเรียกดูคำอธิบายได้โดยการพิมพ์คำสั่ง why



รูปที่ ก9. ตัวอย่างการเรียกดูโมเดลข้อมูลผ่านระบบผู้เชี่ยวชาญ

ภาคผนวก ข

รหัสต้นฉบับของชุดโปรแกรม SUT Miner

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

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

trans(Fn,T,M):-
            retractall(modeM(_,_)),retractall(instanceM(_,_,_)),
            retractall(missingT(_)),
            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]),
            ( attribute(A,B),format('~n~w.',[attribute(A,B)]),fail;true),
            (  MissT=mode-> ( modeSelect,
                        findall(_,(instance(A1,B1,L),maplist(change,L,Lnew),
                                    assert(instanceM(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,
%  c(N),N1 is N+1,mem(N1),
   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])) %add missing value
       )% ******* other rows
           ).

checkLen(Num):-
       instance(No,_,L),length(L,N),  % check length of attribute
       (   N < Num ->write(-No+N),nl ),
       fail.

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

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).

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
            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, 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

gen(N):-N>0, write(N),L1 is N-1,gen(L1).
gen(0):-!.

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

add_inst(N,L):-
            last(L,H,X),col(C1),
            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).

mcheck:- instance(N,class=Class,RecL),  % attri domain check
             attribute(class,ClassL),
                          write(+N), %show OK
            %show error when detected
            (not( (member(Class, ClassL),memberL(RecL)) )
                          ,nl,write(record-N='notOK '),nl )
            ,fail %keep going
            .
%-------aux. predicate---
%
memberL([]).
memberL([H=V|T]):-attribute(H,VL),member(V,VL),memberL(T).
```

```
%----- for spliting the continuous value.

split(X,[H|T],[H2|T2],Hres):- X<H,Hres=H2,! ; split(X,T,T2,Hres).
split(_,_,[L],L). % the default is the highest value.

%------------------------- end of transform--------
%--------------------Find density ----------

:- dynamic dens_list/1,dens_rec/1,no_rec/2.

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 .

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)),!.
%
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---------------
%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,  %<<<< here density
            (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)). %<<<<<<< HERE density

%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(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,'[c1,c3,c4]')),
    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(H, append, [histrogram1, histrogram2]),
  send_list(S, append, [1,2]),
  send_list(O, append, [research, development, marketing]),
  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(O, menu(file_type, cycle)),
    button(cancel, message(Dialog, destroy)),
    button(enter, and( message(@prolog,trans,
                              D1?selection,
                              O?selection,
                              S?selection      ),
                message(Dialog, destroy)))  % enter&destroy
  ]),

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


% ========= End of Pre-Data Mining Program ===========
```

```
/* ============  Data Clustering ========= */
%
% To run the clustering program, call this procedure:
%                              dens_clust_menu.
%
% To run the merge clustering program, call this procedure:
%                              merge_clust_menu.
%-------------------------------------

:-dynamic amountItem/1,instance/3,attribute/2,columnN/1 ,no_rec/2,c/1.

%--------------------Find density ----------
:- dynamic instanceR/3,attributeR/2, dens_list/1,dens_rec/1.

:-dynamic cluster/2,description/2.

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),
                           amountItem(Len),
                           sumlist(VL,SumDens),
                           Dens is SumDens/Len .

%+3,-assert(dens_list([1-1,2-1,...]))
%+No_of_att_match,-assert(dens_list)
all_inst(M):-
       findall(X,instance(X,_,_),AllAttr),
       length(AllAttr,Len),
       numlist(1,Len,L1),  all_dens(L1,L1,M,L),
       maplist(each_dens(L),L1,DL),
       retractall(dens_list(_)), assert(dens_list(DL)),!.

maindens(M):- all_inst(M),  listing(dens_list).

%
all_rec_dens(M,D):- retractall(dens_rec(_)),
             ((M==0;D==0)-> ( writeln(m_______+M),
             findall(X,instance(X,_,_),AllRec))
             ;
             (maindens(M),dens_list(L),
             findall(X,(member(X-D1,L),D1>=D),AllRec))),
             writeln(allRec_________+AllRec),
             assert(dens_rec(AllRec)),!.
```

```
%--------------------PREPROCESS---------------
% pre.pl
%
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(c(_)),
       findall(X,(instance(X,_,_)),AttL),
       length(AttL,Len),
       numlist(1,Len,ColList),
       assert(c(ColList)),
       retractall(amountItem(_)),
       retractall(instanceR(_,_,_)),
       retractall(attributeR(_,_)),
       retractall(no_rec(_,_)),
       assert(amountItem(Len)),!,true.

%random without replacement- L1 for temp List
%randl(+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,  %<<<< here density
       (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),
       writeln(no_____________sel+NoSel),
       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),
           (Per==100->(dens_rec(DensRec),LS=DensRec);rand(C,N,NumSel,[],LS)),
       create_rec(0,LS,A).
```