Họ và tên: Nguyễn Lâm Quốc Bảo

MSSV: 110121007

Lớp: DA21TTA

Báo cáo môn cơ sở trí tuệ nhân tạo

**Câu 1 (Search)**

Sinh viên hãy sử dụng ngôn ngữ lập trình C# để cài đặt bài thực hành Số 4 (đã cung cấp cho sinh viên). Đây là một dạng bài tập trò chơi tìm đường đi trong mê cung. Sinh viên có thể sử dụng dữ liệu input trong bài tập thực hành để viết báo cáo. Nội dung báo cáo cho phần này gồm có: (1) Mô tả tổng quát bài toán; (2) Mô tả cấu trúc dữ liệu để sử dụng cài đặt. (3) Trình bày Giải thuật chính để giải bài toán bằng mã giả; (4) Trình bày một số phương thức cài đặt chính của chương trình. Lưu ý: Phương thức cài đặt ở mục (4) phải phù hợp với mã giả được trình bày ở mục (3).

**1) Mô tả tổng quát bài toán**

- Đây là bài tập trò chơi tìm đường đi trong mê cung với ma trận cho trước là:

0 0 0 0 0 0 0 0

0 0 0 G 0 1 0 0

0 0 0 0 1 1 1 0

1 1 1 1 0 1 1 0

0 0 1 1 1 1 1 0

0 0 0 1 1 0 1 0

0 0 0 0 S 0 1 0

0 0 0 0 0 0 0 0

Và với nơi bắt đầu là S và kết thúc là G, chúng ta cần tìm đường đi từ S tới G sao cho đường đi từ S tới G là ngắn nhất. Nếu như đường đi đi đến các điểm được đánh số 1 sẽ quay về và không được đi tiếp, đi cho đến khi đến được đích là G sẽ kết thúc.

**(2) Mô tả cấu trúc dữ liệu để sử dụng cài đặt.**

Chúng ta sẽ sử dụng một số cấu trúc như:

* Cấu trúc Point: Đại diện cho tọa độ của điểm trên lưới.
* Lớp DoThi: Quản lý đồ thị, bao gồm điểm bắt đầu, điểm đích, ma trận kề và các điểm trong đồ thị. Đọc và in dữ liệu từ tệp.
* Lớp AStarAlg: Thực hiện thuật toán A\* với các phương thức hỗ trợ như kiểm tra tính hợp lệ của điểm, tính giá trị heuristic, và truy vết đường đi.
* Lớp Program: Điểm bắt đầu của chương trình, khởi tạo và chạy thuật toán A\*.

**(3) Trình bày giải thuật chính để giải bài toán bằng mã giả**

class AStarAlg

function AStar(dothi: DoThi)

ROW := dothi.rows

COL := dothi.cols

grid := dothi.getMatrix()

src := Pair(dothi.start.X, dothi.start.Y)

dest := Pair(dothi.goal.X, dothi.goal.Y)

if not IsValid(src.first, src.second, ROW, COL) or not IsValid(dest.first, dest.second, ROW, COL) then

print "Start hoặc Goal không hợp lệ"

return

end if

if not IsUnBlocked(grid, src.first, src.second) or not IsUnBlocked(grid, dest.first, dest.second) then

print "Start hoặc Goal bị chặn"

return

end if

if src.first = dest.first and src.second = dest.second then

print "Chúng ta đã đến đích rồi"

return

end if

closedList := new bool[ROW, COL]

cellDetails := new Cell[ROW, COL]

for i := 0 to ROW - 1 do

for j := 0 to COL - 1 do

cellDetails[i, j].f := Double.MaxValue

cellDetails[i, j].g := Double.MaxValue

cellDetails[i, j].h := Double.MaxValue

cellDetails[i, j].parent\_i := -1

cellDetails[i, j].parent\_j := -1

end for

end for

x := src.first

y := src.second

cellDetails[x, y].f := 0.0

cellDetails[x, y].g := 0.0

cellDetails[x, y].h := 0.0

cellDetails[x, y].parent\_i := x

cellDetails[x, y].parent\_j := y

openList := new SortedSet<Tuple<double, Pair>>(

Comparer<Tuple<double, Pair>>.Create((a, b) => a.Item1.CompareTo(b.Item1)))

openList.add(new Tuple<double, Pair>(0.0, Pair(x, y)))

foundDest := false

while openList.Count > 0 do

p := openList.Min

openList.Remove(p)

x := p.Item2.first

y := p.Item2.second

closedList[x, y] := true

for i := -1 to 1 do

for j := -1 to 1 do

if i = 0 and j = 0 then

continue

end if

newX := x + i

newY := y + j

if IsValid(newX, newY, ROW, COL) then

if IsDestination(newX, newY, dest) then

cellDetails[newX, newY].parent\_i := x

cellDetails[newX, newY].parent\_j := y

print "Đường đi được tìm thấy:"

TracePath(cellDetails, dest)

foundDest := true

return

end if

if not closedList[newX, newY] and IsUnBlocked(grid, newX, newY) then

gNew := cellDetails[x, y].g + 1.0

hNew := CalculateHValue(newX, newY, dest)

fNew := gNew + hNew

if cellDetails[newX, newY].f = Double.MaxValue or cellDetails[newX, newY].f > fNew then

openList.add(new Tuple<double, Pair>(fNew, Pair(newX, newY)))

cellDetails[newX, newY].f := fNew

cellDetails[newX, newY].g := gNew

cellDetails[newX, newY].h := hNew

cellDetails[newX, newY].parent\_i := x

cellDetails[newX, newY].parent\_j := y

end if

end if

end if

end for

end for

end while

if not foundDest then

print "Không tìm thấy đường đi"

end if

end function

function IsValid(row, col, ROW, COL)

return (row >= 0) and (row < ROW) and (col >= 0) and (col < COL)

end function

function IsUnBlocked(grid, row, col)

return grid[row, col] = 1

end function

function IsDestination(row, col, dest)

return (row = dest.first) and (col = dest.second)

end function

function CalculateHValue(row, col, dest)

return SquareRoot(Power(row - dest.first, 2) + Power(col - dest.second, 2))

end function

function TracePath(cellDetails, dest)

print "Con đường là"

row := dest.first

col := dest.second

Path := new Stack<Pair>()

while not (cellDetails[row, col].parent\_i = row and cellDetails[row, col].parent\_j = col) do

Path.Push(Pair(row, col))

temp\_row := cellDetails[row, col].parent\_i

temp\_col := cellDetails[row, col].parent\_j

row := temp\_row

col := temp\_col

end while

Path.Push(Pair(row, col))

while Path.Count > 0 do

p := Path.Peek()

Path.Pop()

print "-> ({0},{1})".format(p.first, p.second)

end while

end function

end class

mã giả này có chức năng:

**Hàm AStar(dothi: DoThi):**

* Đây là hàm chính thực hiện thuật toán A\* trên đồ thị dothi.
* Đầu tiên, nó lấy các thông tin cần thiết như kích thước của đồ thị, ma trận liên kết, điểm xuất phát và điểm đích.
* Kiểm tra xem điểm xuất phát và điểm đích có hợp lệ không, và liệu chúng có bị chặn không. Nếu không hợp lệ hoặc bị chặn, hàm sẽ dừng và in ra thông báo tương ứng.
* Tiếp theo, nó tạo danh sách đóng (closed list) để theo dõi các ô đã được xét đến, và tạo một bảng chi tiết (cellDetails) để lưu thông tin về mỗi ô.
* Sau đó, nó bắt đầu với điểm xuất phát và thêm nó vào danh sách mở (open list).
* Trong vòng lặp, nó duyệt qua các ô trong danh sách mở, chọn ô có chi phí thấp nhất và xét xem có đạt được đích đến hay không.
* Nếu tìm thấy đường đi đến đích, nó in ra đường đi và kết thúc.
* Nếu không tìm thấy đường đi, nó in ra thông báo không tìm thấy và kết thúc.

**Các hàm phụ trợ:**

* IsValid(row, col, ROW, COL): Kiểm tra xem ô với hàng và cột nhất định có nằm trong phạm vi của bảng không.
* IsUnBlocked(grid, row, col): Kiểm tra xem ô đã cho có bị chặn hay không.
* IsDestination(row, col, dest): Kiểm tra xem ô đã cho có phải là đích đến hay không.
* CalculateHValue(row, col, dest): Tính toán giá trị h(n) (giá trị heuristic) cho ô đã cho.
* TracePath(cellDetails, dest): Theo dõi đường đi từ ô đích đến điểm xuất phát và in ra đường đi đó.
* Cả lớp AStarAlg và các hàm phụ trợ được thiết kế để thực hiện thuật toán A\* trên một đồ thị và trả về đường đi ngắn nhất từ điểm xuất phát đến điểm đích.

**(4) Trình bày một số phương thức cài đặt chính của chương trình.**

**Hàm AStar(dothi: DoThi):**

* Đầu vào: Một đối tượng đồ thị dothi chứa thông tin về đồ thị, điểm xuất phát và điểm đích.
* Đầu ra: In ra đường đi ngắn nhất từ điểm xuất phát đến điểm đích (nếu có).
* Thuật toán: Thực hiện thuật toán A\* trên đồ thị dothi để tìm đường đi ngắn nhất từ điểm xuất phát đến điểm đích.

**Hàm IsValid(row, col, ROW, COL):**

* Đầu vào: Chỉ số hàng row, chỉ số cột col, số hàng ROW và số cột COL của bảng.
* Đầu ra: Trả về true nếu ô với hàng và cột nhất định nằm trong phạm vi của bảng, ngược lại trả về false.

**Hàm IsUnBlocked(grid, row, col):**

* Đầu vào: Ma trận liên kết grid, chỉ số hàng row và chỉ số cột col.
* Đầu ra: Trả về true nếu ô đã cho không bị chặn (có giá trị là 1), ngược lại trả về false.

**Hàm IsDestination(row, col, dest):**

* Đầu vào: Chỉ số hàng row, chỉ số cột col, và điểm đích dest.
* Đầu ra: Trả về true nếu ô đã cho là điểm đích, ngược lại trả về false.

**Hàm CalculateHValue(row, col, dest):**

* Đầu vào: Chỉ số hàng row, chỉ số cột col, và điểm đích dest.
* Đầu ra: Tính toán giá trị h(n) (giá trị heuristic) cho ô đã cho và trả về giá trị đó.

**Hàm TracePath(cellDetails, dest):**

* Đầu vào: Một bảng chi tiết cellDetails và điểm đích dest.
* Đầu ra: Theo dõi đường đi từ điểm đích về điểm xuất phát trong bảng chi tiết và in ra đường đi đó.

Các phương thức này cài đặt theo logic của mã giả và thực hiện các chức năng tương ứng để tìm đường đi ngắn nhất từ điểm xuất phát đến điểm đích trên một đồ thị.

# **Câu 2: (Biểu diễn tri thức)**

## 2.1 Cho cơ sở tri thức với các phát biểu như sau:

1. Bhogendra thích tất cả các loại thực phẩm.
2. Oranges là thực phẩm
3. Chicken là thực phẩm.
4. Tất cả mọi thứ mà người ta ăn được và không chết là thực phẩm.
5. Nếu một người thích một thực phẩm nào đó có nghĩa là người đó đã ăn nó.
6. Jogendra ăn đậu phộng và vẫn còn sống.
7. Shailendra ăn mọi thứ mà Bhogendra ăn.

### Hãy viết logic vị từ để biểu diễn tri thức cho các phát biểu trên.

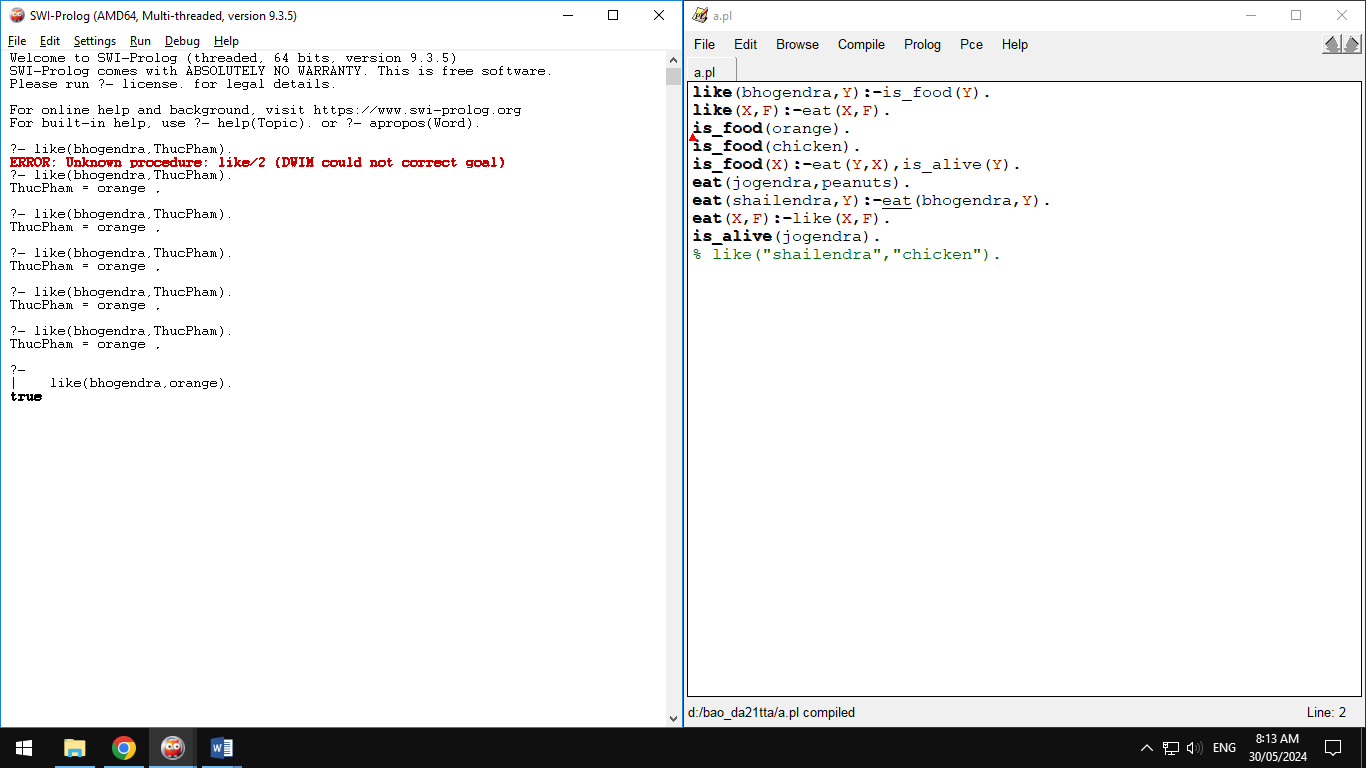
### Sử dụng hợp giải Robinson để trả lời câu hỏi: Có phải Shailendra thích Chicken?

### Sử dụng ngôn ngữ lập trình Prolog để cài đặt và kiểm chứng các suy diễn

**Giải**

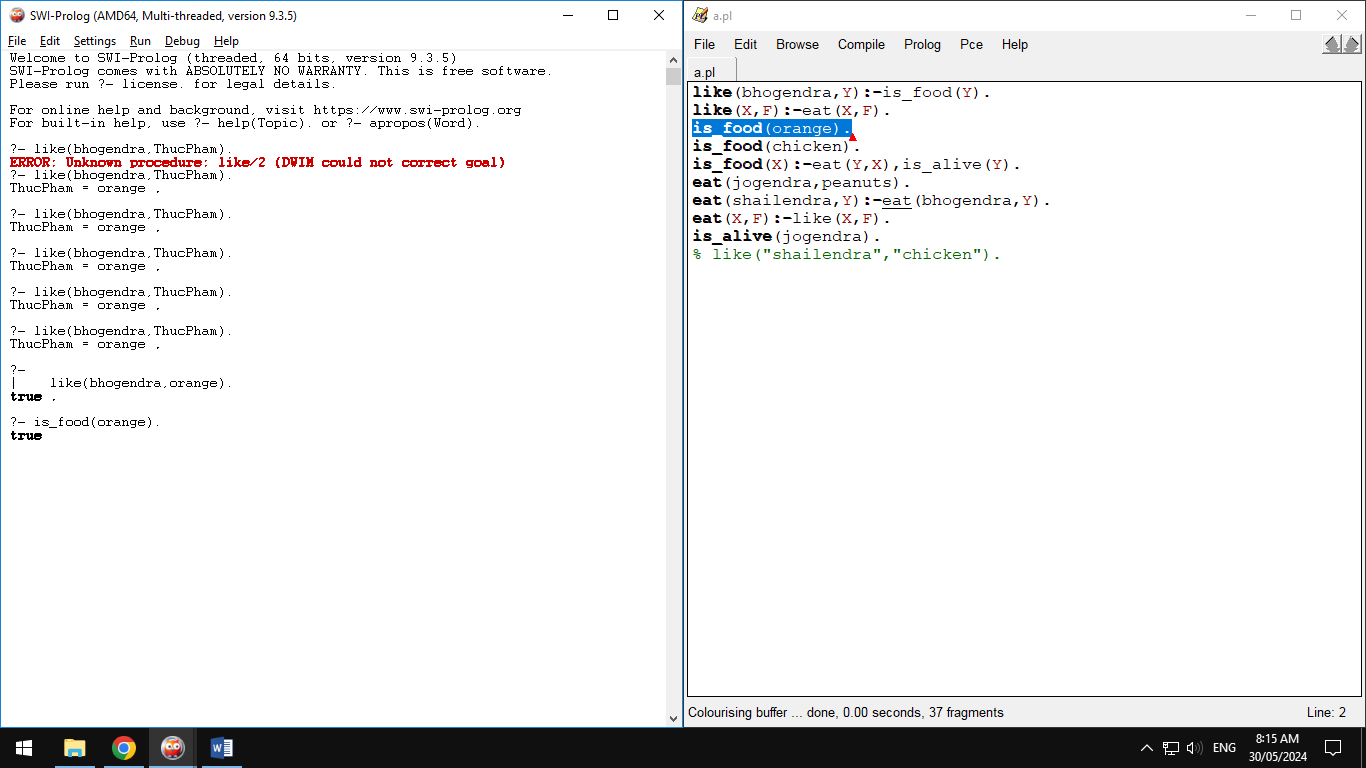
1. Bhogendra thích tất cả các loại thực phẩm.

∀x(T(x)⇒P(Bhogendra))



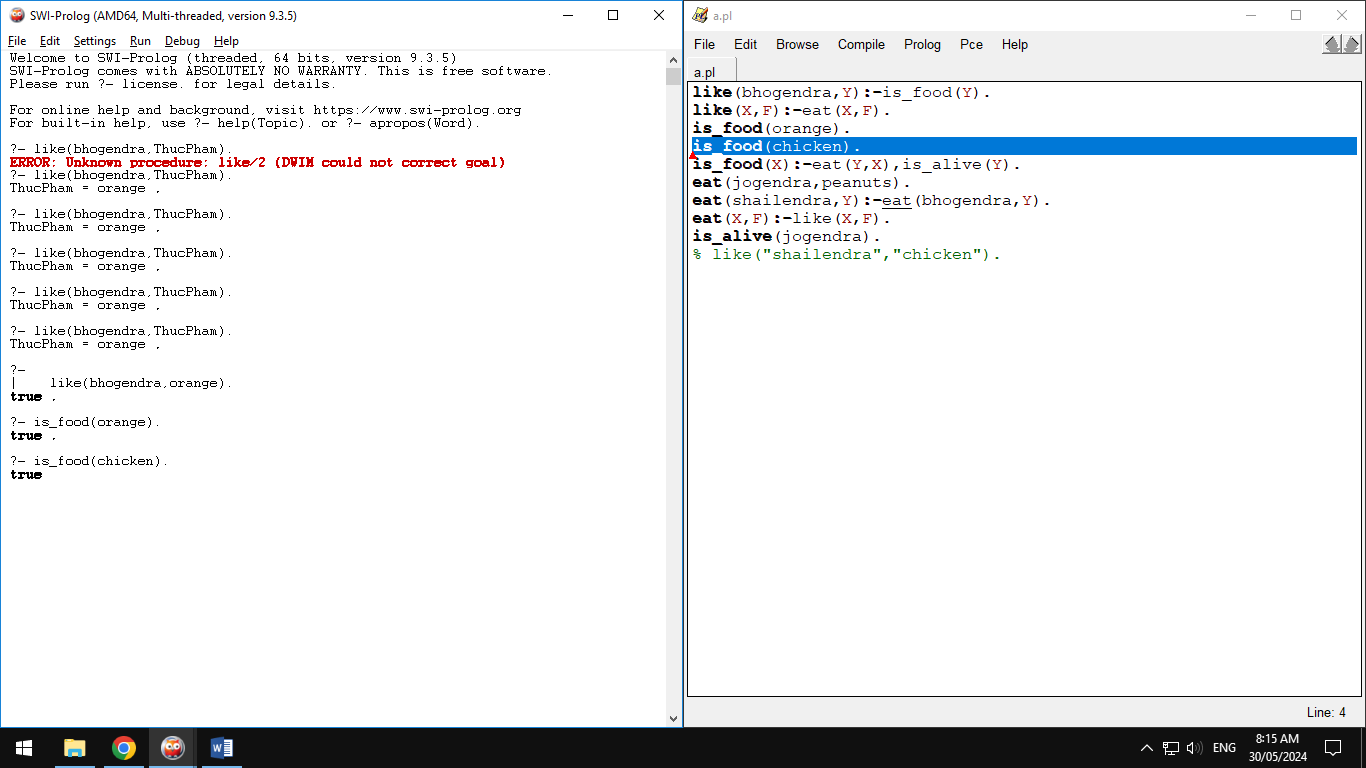
1. Oranges là thực phẩm

∀x,(Oranges)⇒ T(x)



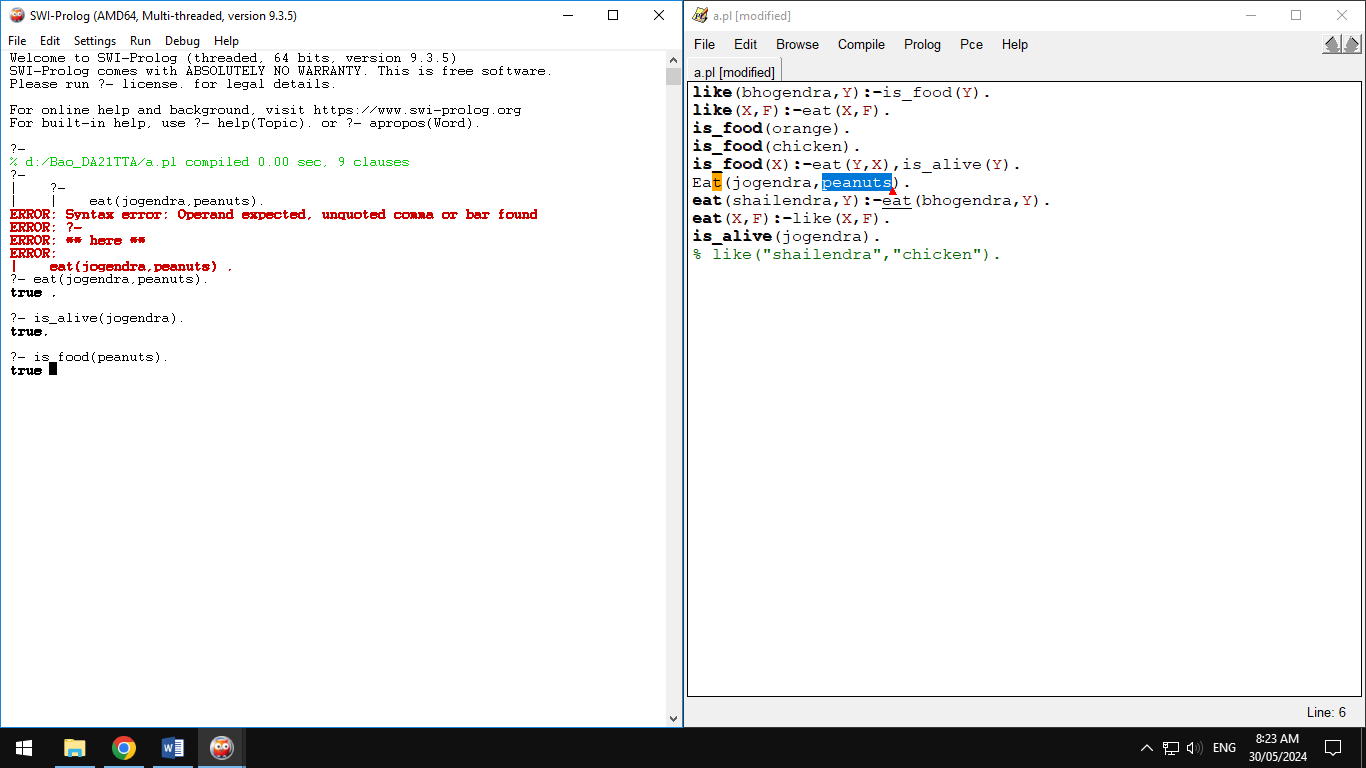
1. Chicken là thực phẩm.

∀x,(chicken)⇒ T(x)



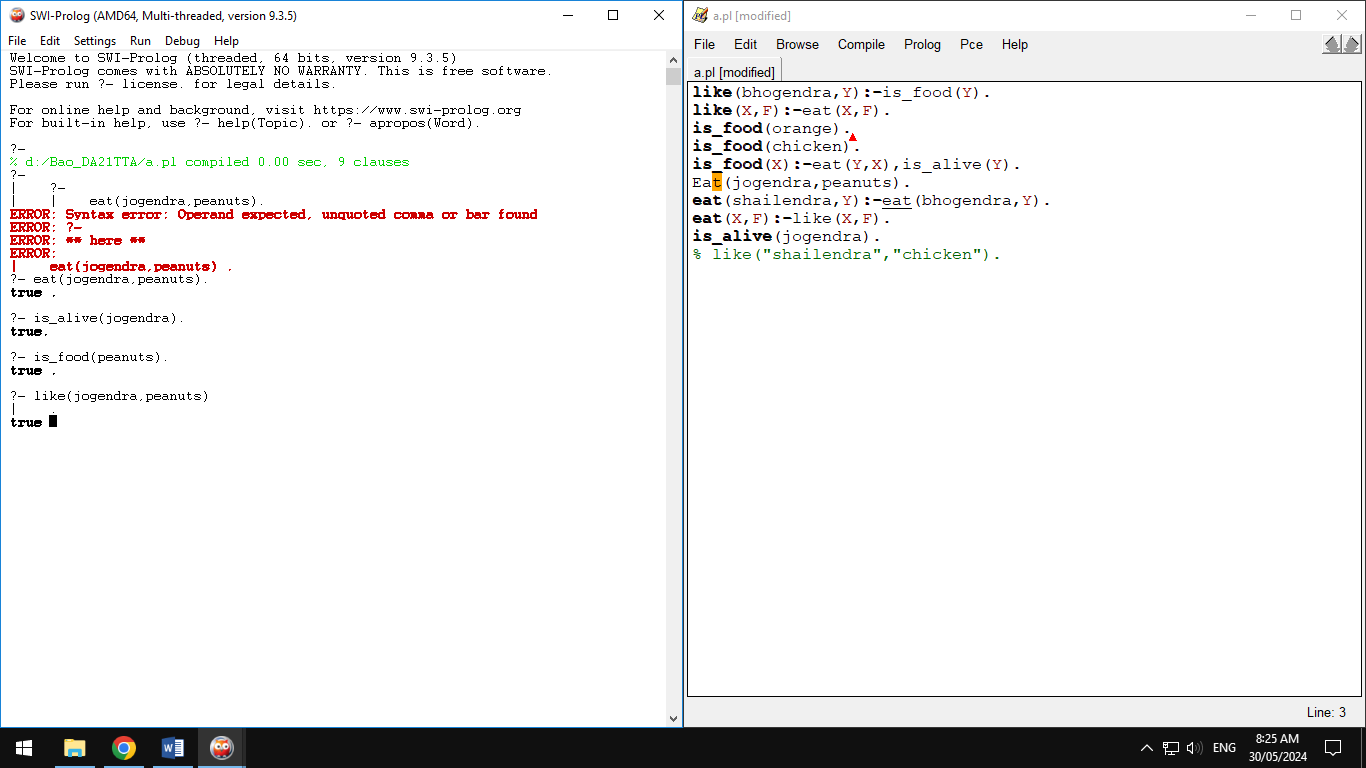
1. Tất cả mọi thứ mà người ta ăn được và không chết là thực phẩm.

∀x(things),((Eat(x)∧Survives(x))⇒Food(x))



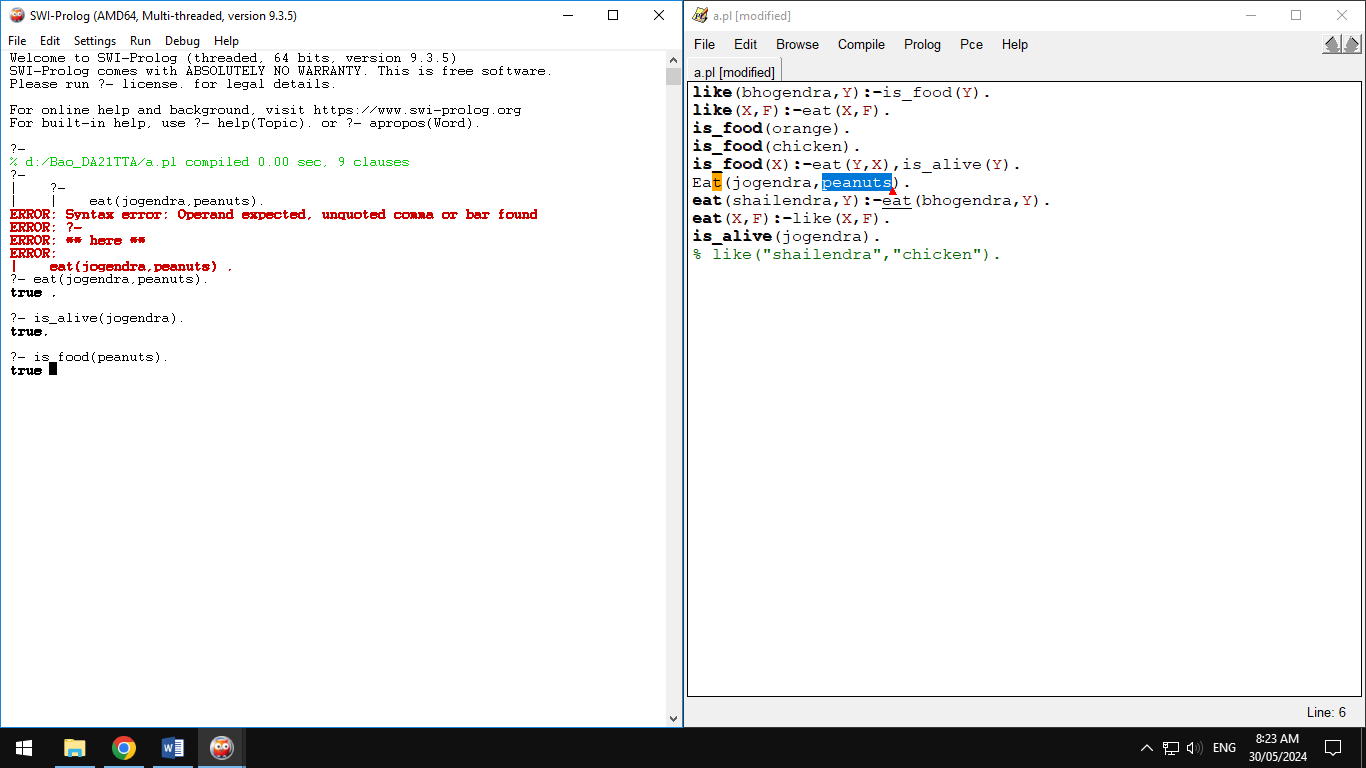
1. Nếu một người thích một thực phẩm nào đó có nghĩa là người đó đã ăn nó.

∀x(người),∀y(thực phẩm),(thích(x,y)⇒ăn(x,y))



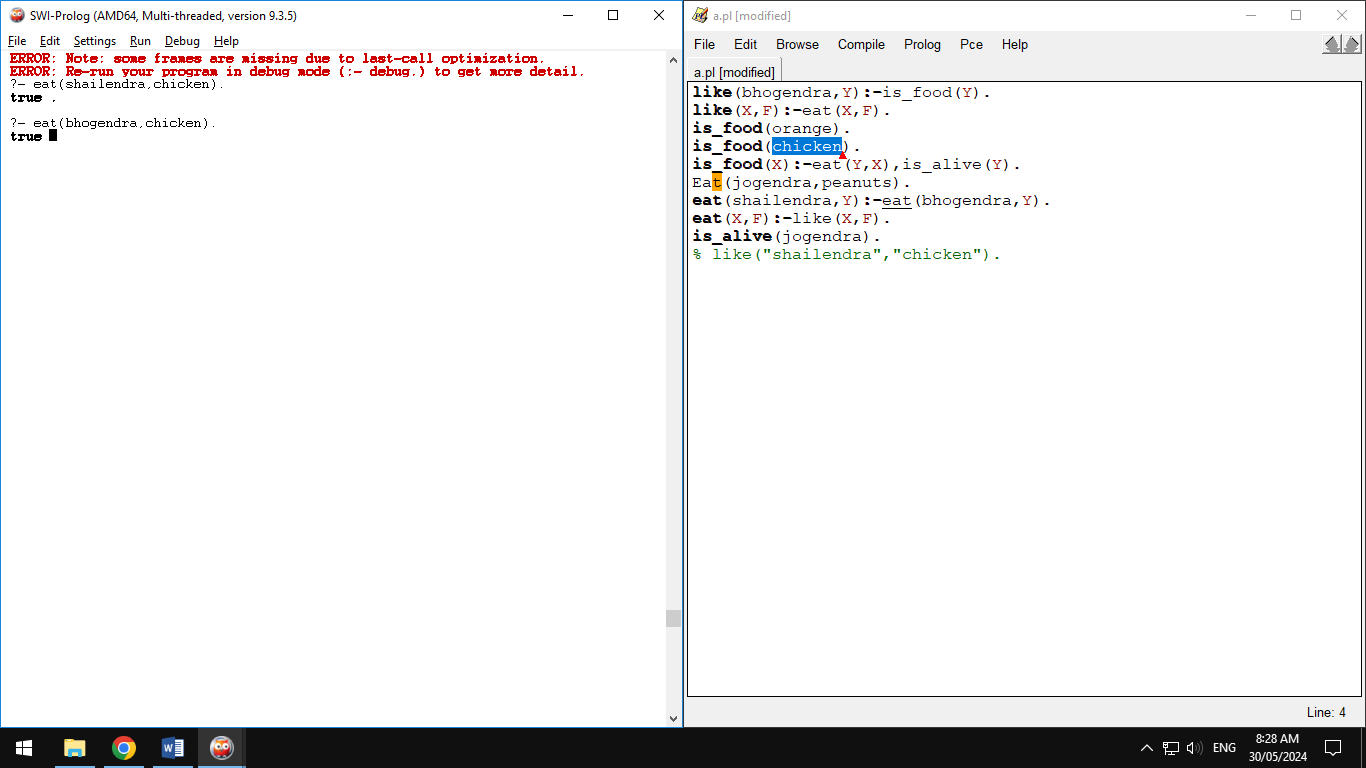
1. Jogendra ăn đậu phộng và vẫn còn sống.

Eat(Jogendra,đậu phộng)∧Sống(Jogendra)



1. Shailendra ăn mọi thứ mà Bhogendra ăn.

∀y(Eats(Bhogendra,y)⇒Eát(Shailendra,y))



**b. Sử dụng hợp giải Robinson để trả lời câu hỏi: Có phải Shailendra thích Chicken?**

Bhogendra thích tất cả các loại thực phẩm.

∀x(T(x)⇒P(Bhogendra))

Oranges là thực phẩm

∀x,(Oranges)⇒ T(x)

Chicken là thực phẩm.

∀x,(chicken)⇒ T(x)

Tất cả mọi thứ mà người ta ăn được và không chết là thực phẩm.

∀x(things),((Eat(x)∧Survives(x))⇒Food(x))

Nếu một người thích một thực phẩm nào đó có nghĩa là người đó đã ăn nó.

∀x(người),∀y(thực phẩm),(thích(x,y)⇒ăn(x,y))

Jogendra ăn đậu phộng và vẫn còn sống.

Eat(Jogendra,đậu phộng)∧Sống(Jogendra)

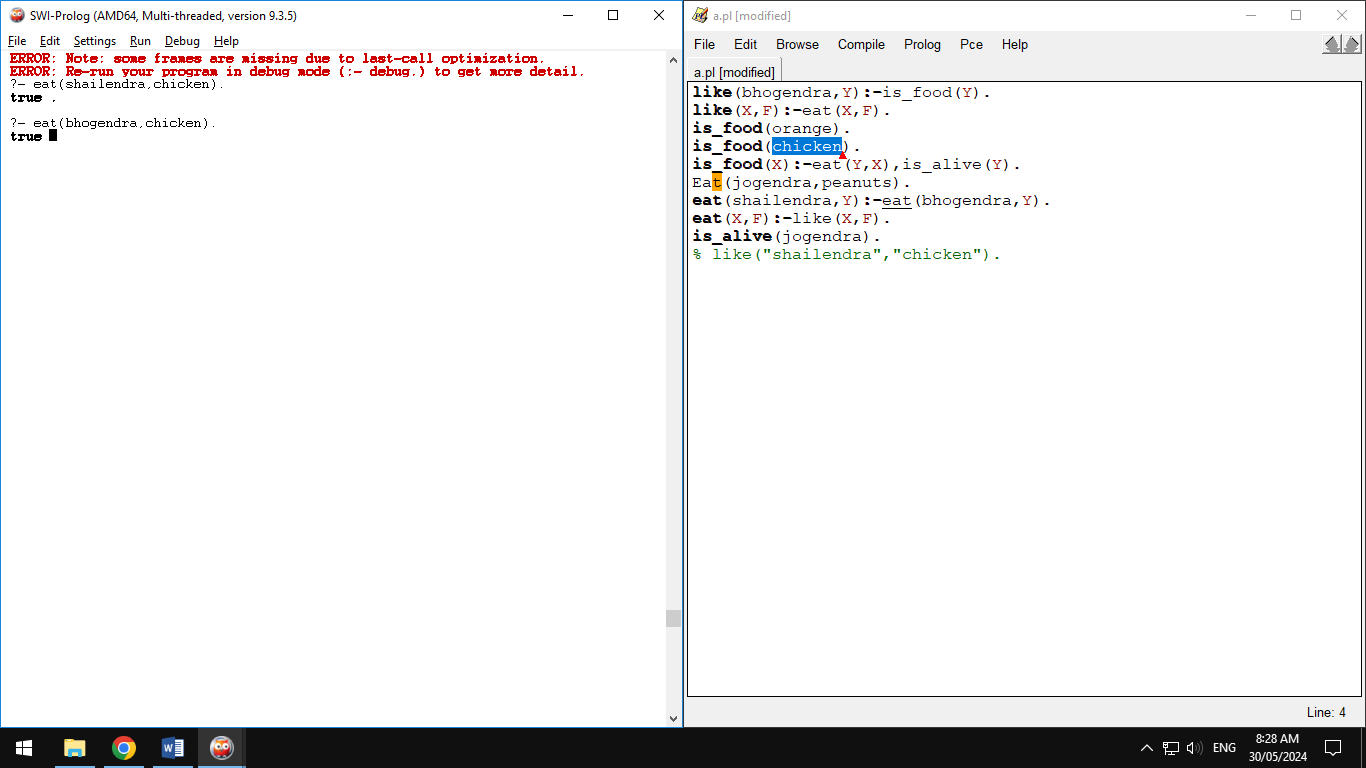
Shailendra ăn mọi thứ mà Bhogendra ăn.

∀y(Eats(Bhogendra,y)⇒Eát(Shailendra,y))

8. thích(Bhogendra,Chicken) (mệnh đề 2, mệnh đề 1)

9. Eat(Bhogendra,Chicken) (mệnh đề 8, mệnh đề 5)

10. Eat(Shailendra,Chicken) (mệnh đề 9, mệnh đề 4)



## **2.2. Cho cơ sở tri thức với các phát biểu như sau:**

**1. Nam là bố của Lan**

**2. Nam là chồng của Hương**

**3. Nếu A là bố của B thì B là con của A**

**4. Nếu A là chồng của B thì B là vợ của A**

**5. Nếu A là vợ của B, và C là con của B, thì A là mẹ của C**

### Hãy viết logic vị từ để biểu diễn tri thức cho các phát biểu trên.

### Sử dụng hợp giải Robinson để trả lời câu hỏi: Lan là con của ai?

### Sử dụng ngôn ngữ lập trình Prolog để cài đặt và kiểm chứng các suy diễn

**Giải**

a. Hãy viết logic vị từ để biểu diễn tri thức cho các phát biểu trên.

**1. Nam là bố của Lan**

**Bố(Nam,Lan)**

**2. Nam là chồng của Hương**

**` Chồng(Nam,Hương)**

**3. Nếu A là bố của B thì B là con của A**

∀A∀B (Bố(A,B) => Con(B,A))

**4. Nếu A là chồng của B thì B là vợ của A**

∀A∀B (Chồng(A,B) => Vợ(B,A))

**5. Nếu A là vợ của B, và C là con của B, thì A là mẹ của C**

∀A∀B∀C ((Vợ(A,B)∧Con(C,B))⇒Mẹ(A,C))

b. Sử dụng hợp giải Robinson để trả lời câu hỏi: Lan là con của ai?

**1. Bố(Nam,Lan)**

**2. Chồng(Nam,Hương)**

**3.** ∀A∀B (Bố(A,B) => Con(B,A))

**4.** ∀A∀B (Chồng(A,B) => Vợ(B,A))

**5.** ∀A∀B∀C ((Vợ(A,B)∧Con(C,B))⇒Mẹ(A,C))

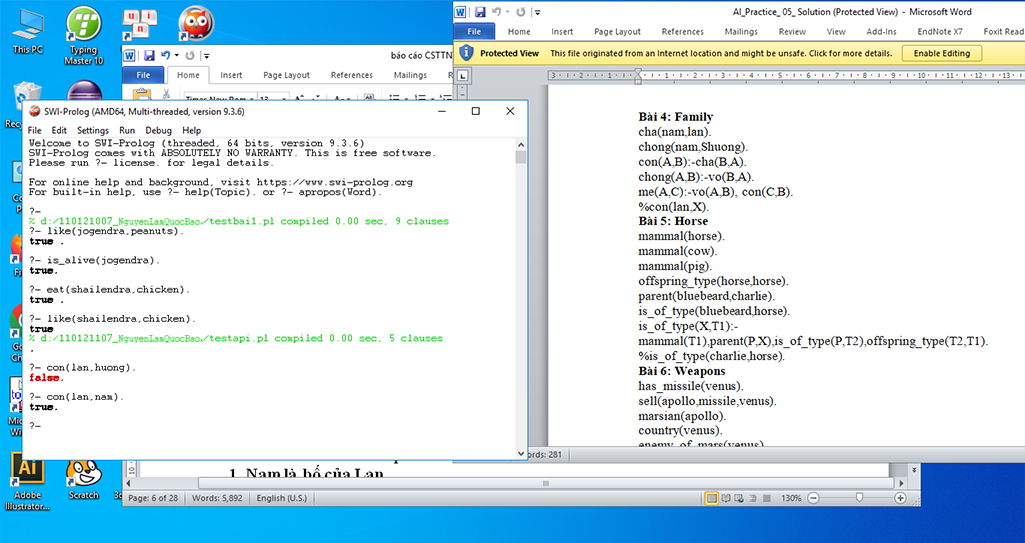
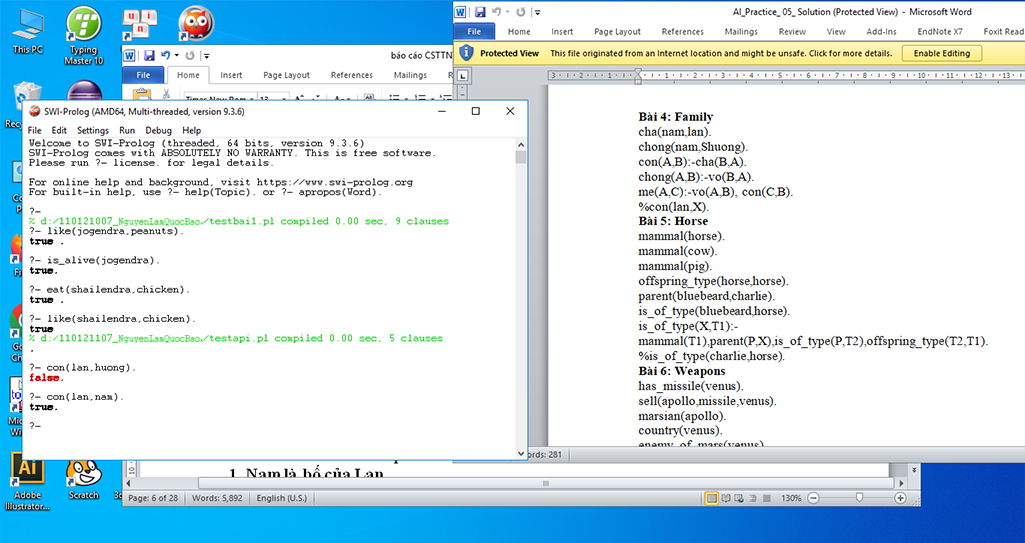
6. Con(Lan,A)

7. Con(Lan,Nam) (mệnh đề 6, phép thế) (mệnh đề 1)

8. Bố(Nam,Lan) (mệnh đề 3 với A = Nam, B = Lan)

9. Mẹ(Hương,Lan) (mệnh đề 5 với A = Nam, B = Hương, C = Lan)

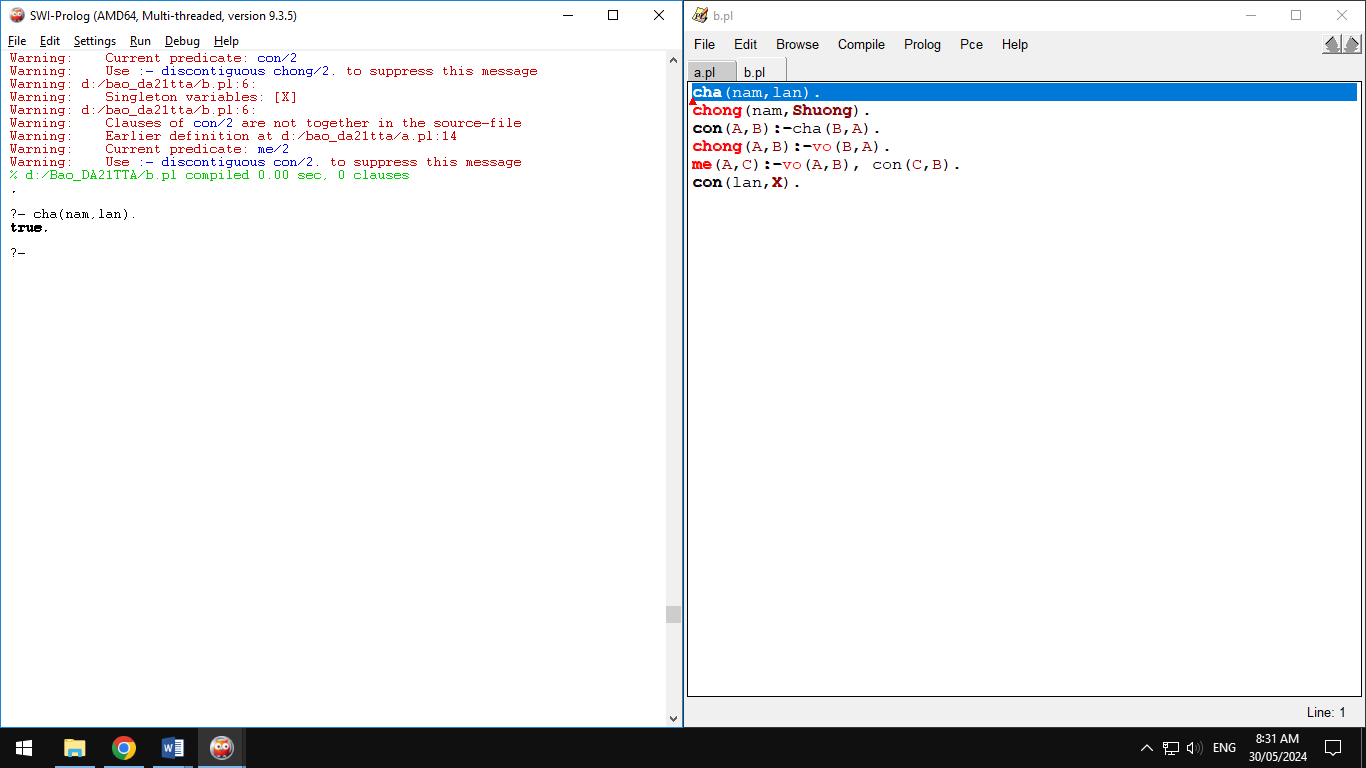
Vậy Lan là con của Hương và Nam



c. Sử dụng ngôn ngữ lập trình Prolog để cài đặt và kiểm chứng các suy diễn

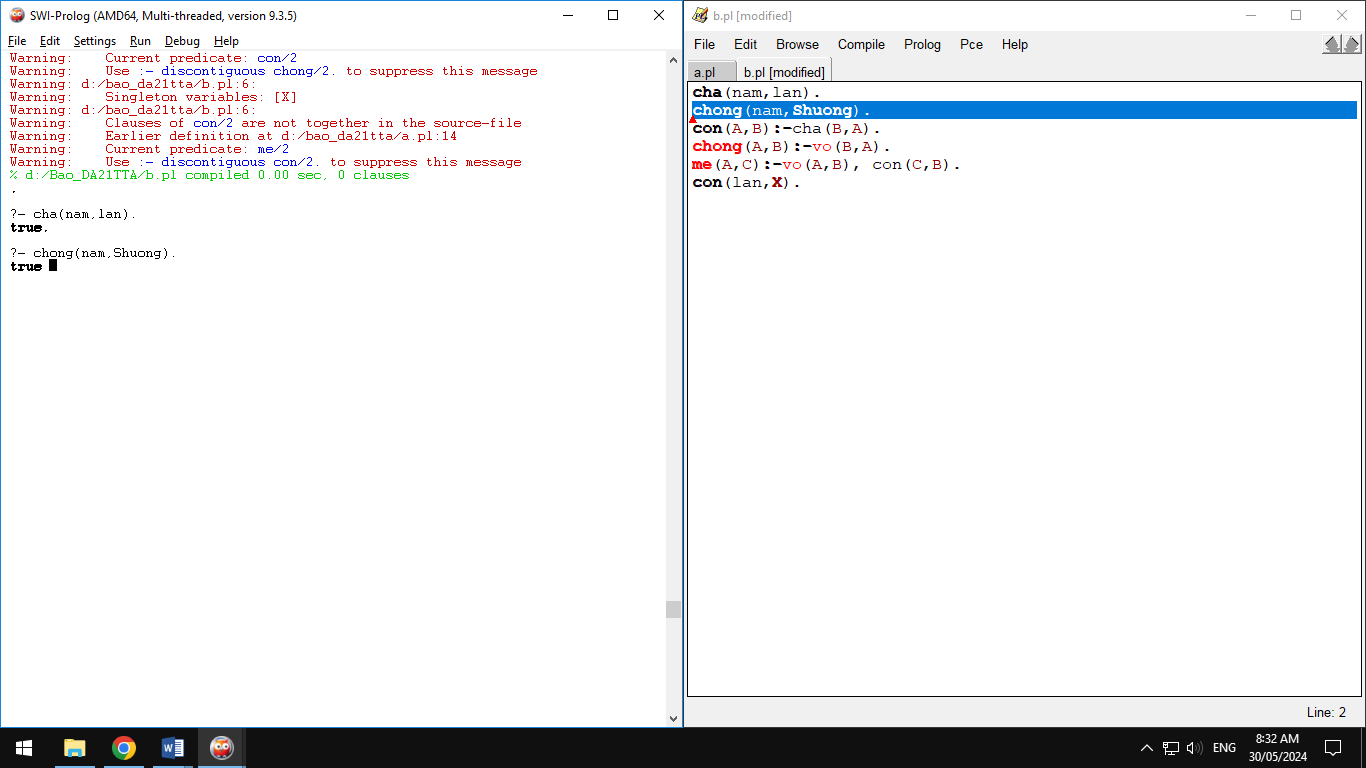
**1. Nam là bố của Lan**

**Bố(Nam,Lan)**



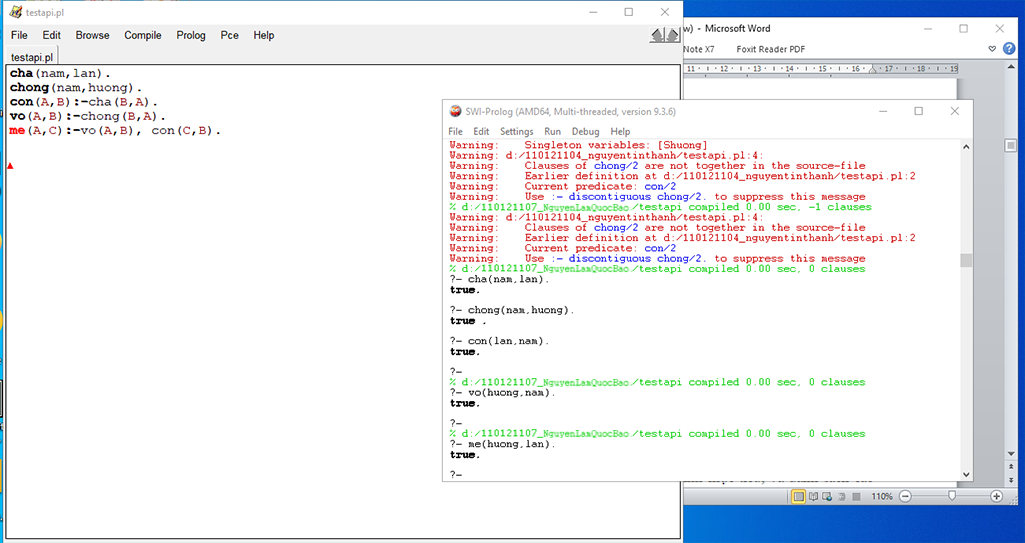
**2. Nam là chồng của Hương**

**Chồng(Nam,Hương)**



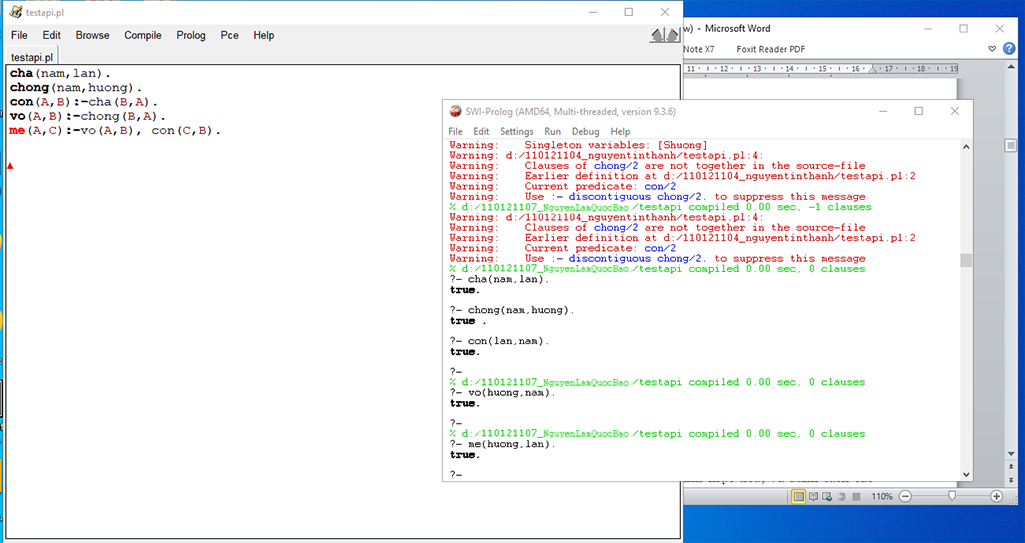
**3. Nếu A là bố của B thì B là con của A**

∀A∀B (Bố(A,B) => Con(B,A))



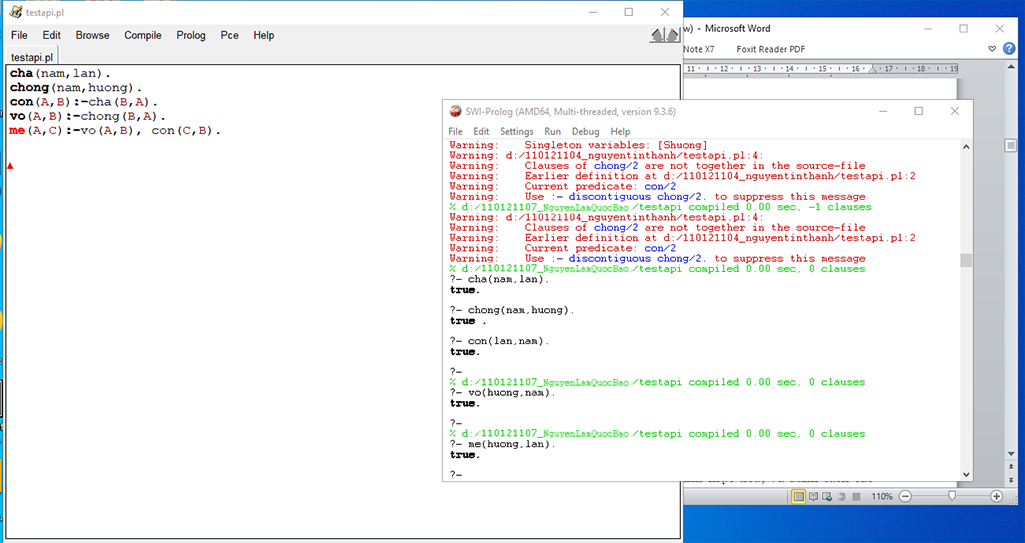
**4. Nếu A là chồng của B thì B là vợ của A**

∀A∀B (Chồng(A,B) => Vợ(B,A))



**5. Nếu A là vợ của B, và C là con của B, thì A là mẹ của C**

∀A∀B∀C ((Vợ(A,B)∧Con(C,B))⇒Mẹ(A,C))



# **Câu 3: (Machine Learning)**

Sử dụng phần mềm ứng dụng Weka để thực hiện kiểm chứng 2 giải thuật ID3 and NaiveBayes Classification cho 2 tập dữ liệu Weather và Diabetes (GVHD cung cấp dữ liệu trong buổi học tiếp theo). Nội dung báo cáo gồm:

* Mô tả tập dữ liệu bao gồm:
  + Số thuộc tính, thuộc tính dẫn xuất, thuộc tính mục tiêu, và danh sách các thuộc tính
  + Liệt kê giá trị của từng thuộc tính
* Sử dụng giải thuật phân loại ID3 để rút ra các luật từ tập dữ liệu trên
* Sử dụng giải thuật phân lớp Naïve Bayes Classification để thực hiện phân loại 2 tập dữ liệu trên.
* Thực hiện so sánh kết quả của 2 phương pháp phân loại trên từng tập dữ liệu.
* Viết báo cáo cho các bước thực hiện và so sánh kết quả.

**Làm**

Tập dữ liệu Weather bao gồm các thuộc tính:

* Outlook (Tình trạng trời): Đây là thuộc tính mô tả tình trạng thời tiết vào một ngày cụ thể. Ví dụ, nó có thể được phân loại thành "Sunny" (Nắng), "Overcast" (Trời âm u) hoặc "Rainy" (Mưa).
* Temperature (Nhiệt độ): Đây là thuộc tính mô tả nhiệt độ vào một ngày cụ thể. Nó có thể được đo bằng đơn vị đo nhiệt độ như Celsius hoặc Fahrenheit.
* Humidity (Độ ẩm): Đây là thuộc tính mô tả mức độ độ ẩm trong không khí vào một ngày cụ thể. Nó có thể được đo bằng phần trăm.
* Windy (Có gió hay không): Đây là thuộc tính mô tả có gió hay không vào một ngày cụ thể. Nó có thể được phân loại thành "True" (Có) hoặc "False" (Không).
* Play (Chơi): Đây là thuộc tính mục tiêu mà chúng ta muốn dự đoán. Nó biểu thị liệu có thích hợp để chơi bóng đá vào một ngày cụ thể dựa trên các điều kiện thời tiết đã được cung cấp.

Có 14 dòng

Số thuộc tính: 5

Thuộc tính dẫn xuất: Không có thông tin cụ thể về thuộc tính dẫn xuất trong bộ dữ liệu.

Thuộc tính mục tiêu: "Play" (Chơi)

Các giá trị của từng thuộc tính:

1.outlook (tình trạng trời)

+ Sunny (Nắng)

+ Overcast (Trời âm u)

+ Rainy (Mưa)

2.Temperature (nhiệt độ): gồm các giá trị đo lường nhiệt độ cụ thể (ví dụ: 70oF, 90oF)

3.Humidity (Độ ẩm): Các giá trị đo lường độ ẩm cụ thể (ví dụ 70%, 90%)

4.Windy (có gió hay không):

+ True (có)

+ False (không)

5.Play (có chơi hay không):

+ True (có)

+ False (không)

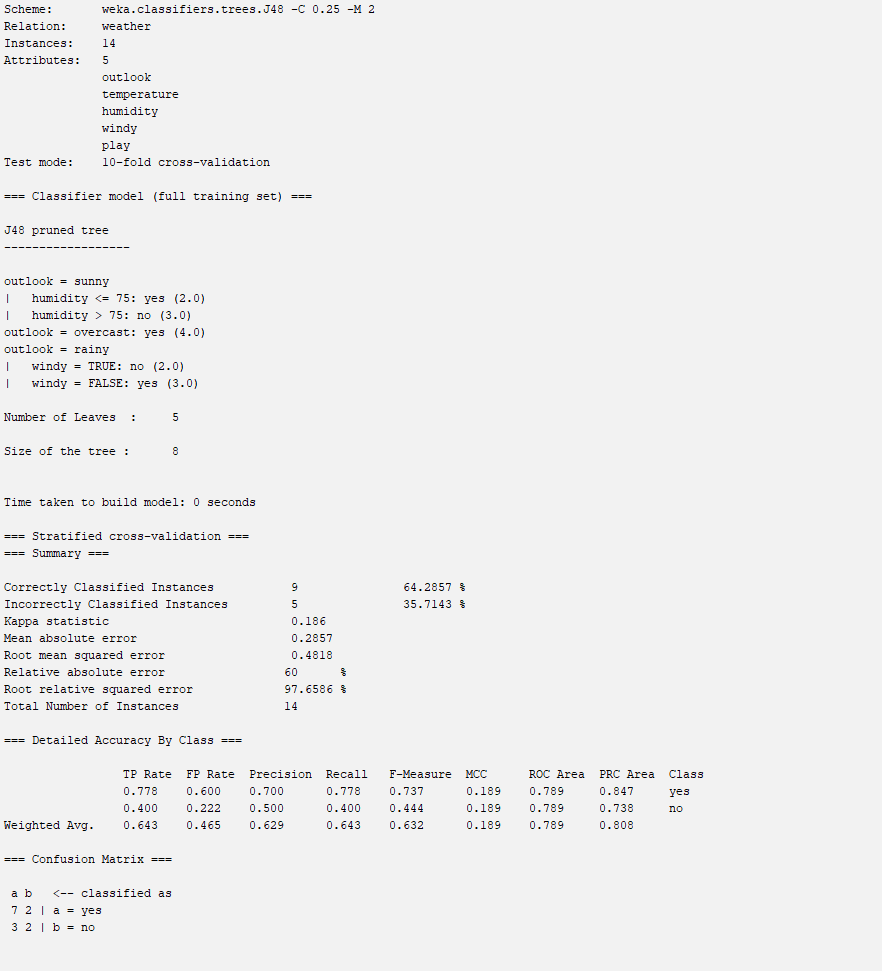
Cách chạy giải thuật ID3 trên weka:

Bước 1: chọn bộ dữ liệu weka: vào program file (x86) => weka => data => chọn bộ dữ liệu có tên là Weather.

Bước 2: trên ứng dụng weka chọn classify => chọn choose => chọn giải thuật J4.8

Bước 3: lựa chọn các thuộc tính (ở đây đang chọn là Play).

Sau khi chạy giải thuật ID3



Dưới đây là một phân tích và giải thích về cây quyết định đã được cải thiện bằng cách cắt tỉa (pruned tree):

**1. \*\*Cây quyết định cắt tỉa\*\*:**

- Có tổng cộng 5 lá và 8 nút.

- Mỗi nút đại diện cho một điều kiện phân loại và mỗi lá đại diện cho một dự đoán cuối cùng.

**2. \*\*Quyết định của cây\*\*:**

- Khi trạng thái thời tiết là "sunny":

- Nếu độ ẩm <= 75, dự đoán là "yes".

- Nếu độ ẩm > 75, dự đoán là "no".

- Khi trạng thái thời tiết là "overcast", dự đoán là "yes".

- Khi trạng thái thời tiết là "rainy":

- Nếu có gió (windy = TRUE), dự đoán là "no".

- Nếu không có gió (windy = FALSE), dự đoán là "yes".

**\*\*Kết quả của mô hình\*\*:**

- Tỉ lệ phân loại đúng: 64.2857%.

- Giá trị Kappa statistic: 0.186.

- Độ lỗi tuyệt đối trung bình: 0.2857.

- Độ lỗi bình phương trung bình: 0.4818.

- Đánh giá chi tiết theo từng lớp cho thấy một sự cải thiện nhất định so với mô hình trước khi cắt tỉa.

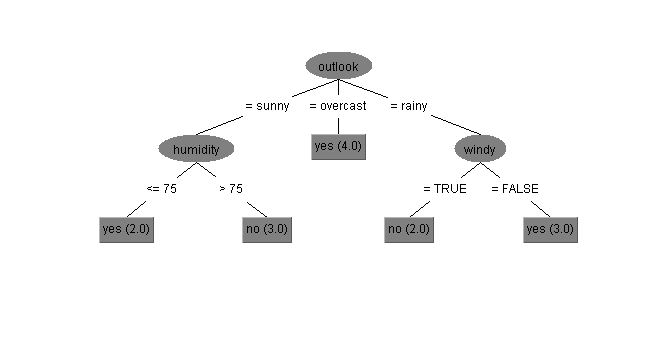
- Confusion matrix cho thấy có 7 mẫu dự đoán là yes và 2 mẫu dự đoán là no được phân loại đúng, trong khi có 3 mẫu dự đoán là no và 2 mẫu dự đoán là yes được phân loại sai.

**\*\*Nhận xét\*\*:**

- Mô hình đã cải thiện so với trước khi cắt tỉa, nhưng vẫn còn không đủ chính xác với tỉ lệ phân loại đúng là 64.2857%.

- Có sự cải thiện đáng kể trong giá trị Kappa statistic so với trước khi cắt tỉa.

- Confusion matrix cho thấy mô hình đang gặp khó khăn trong việc phân loại các mẫu nhất định. Có thể cần xem xét thêm để cải thiện hiệu suất của mô hình.



Trong dạng tree ta rút ra được một số thuộc tính

Outlook = sunny:

+ nếu humidity <= 75 thì sẽ có 2 mẫu yes

+ nếu humidity > 75 thì sẽ có 3 mẫu no

=> nếu độ ẩm càng cao thì số lượng người chơi bóng đá sẽ giảm

Outlook = overcast:

+ nếu outlook = overcast thì sẽ có 4 mẫu yes

=> nếu trời âm u thì tất cả mọi người sẽ đi chơi bóng đá

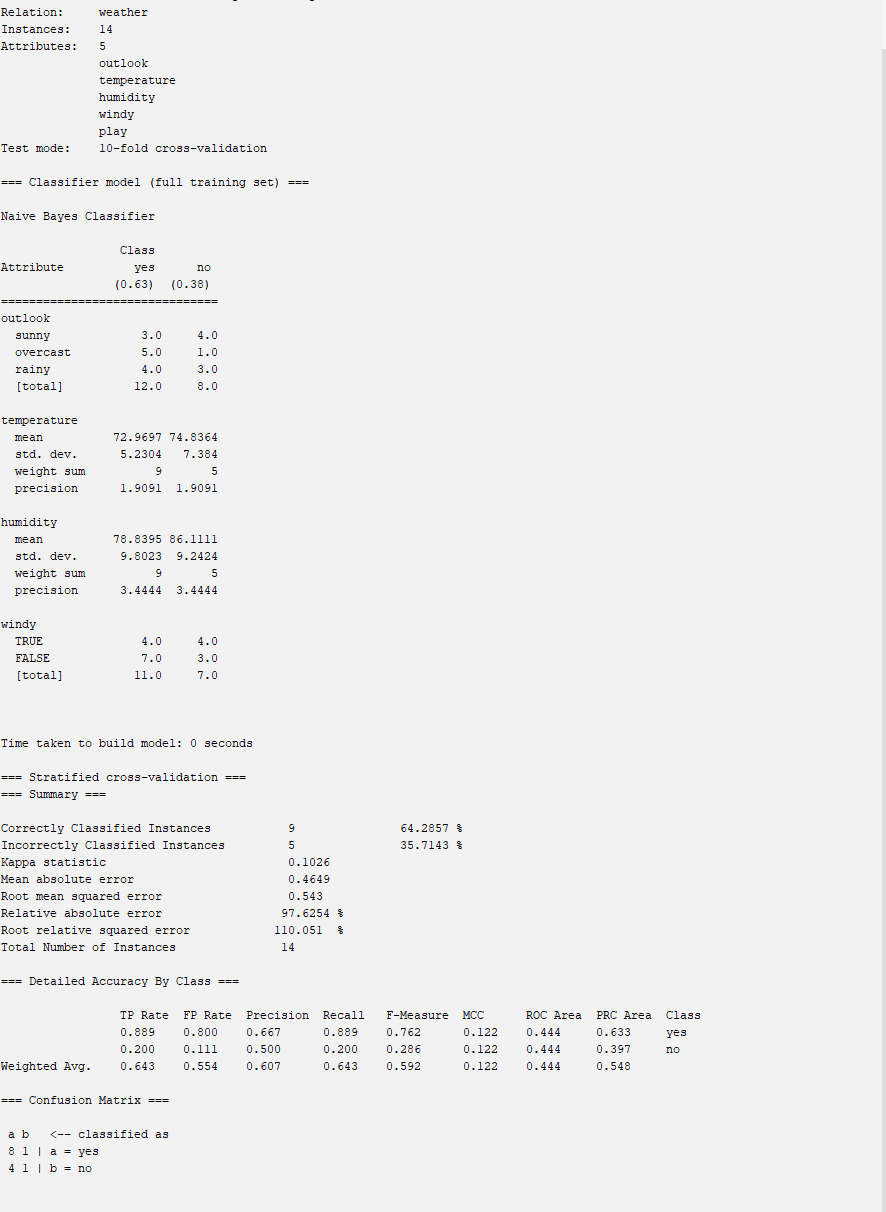
Outlook = rainy:

+ nếu windy = true thì sẽ có 2 mẫu no

+ nếu windy = false sẽ có 3 mẫu yes

=> nếu trời mưa mà có gió thì hầu như mọi người sẽ không chơi bóng đá, nhưng nếu không có gió thì họ sẽ có thể chơi bóng kể cả trời mưa.

Sau khi chạy giải thuật Bayes



Tình trạng thời tiết (Outlook):

Có ba loại: sunny, overcast, và rainy.

Đối với mỗi loại thời tiết, có số lượng mẫu và tỷ lệ phân loại dự đoán là yes hoặc no.

Ví dụ: Có 3 mẫu dự đoán là yes và 4 mẫu dự đoán là no khi thời tiết là sunny.

Nhiệt độ (Temperature):

Được mô tả bằng trung bình và độ lệch chuẩn của nhiệt độ.

Có các thông số như trọng số (weight sum) và độ chính xác (precision).

Độ ẩm (Humidity):

Tương tự như nhiệt độ, được mô tả bằng trung bình và độ lệch chuẩn của độ ẩm.

Cũng có các thông số như trọng số và độ chính xác.

Gió (Windy):

Có hai loại: TRUE và FALSE.

Mỗi loại có số lượng mẫu và tỷ lệ phân loại dự đoán là yes hoặc no.

Dưới đây là kết quả của mô hình phân loại:

Tổng số mẫu: 14.

Tỉ lệ phân loại đúng: 64.2857%.

Tỉ lệ phân loại sai: 35.7143%.

Kappa statistic: 0.1026.

Mean absolute error: 0.4649.

Root mean squared error: 0.543.

Đánh giá chi tiết theo từng lớp (yes và no) bao gồm tỷ lệ TP (true positive), FP (false positive), Precision, Recall, F-Measure, MCC (Matthews correlation coefficient), ROC Area và PRC Area.

Nhận xét:

Mô hình có tỉ lệ phân loại đúng không cao (64.2857%).

Giá trị Kappa statistic (0.1026) thấp, ngụ ý rằng sự đồng thuận giữa dự đoán và kết quả thực tế không cao.

Đánh giá chi tiết theo từng lớp (yes và no) cho thấy một số tham số như Precision và Recall không đạt mức cao.

Confusion matrix cho thấy có 8 mẫu dự đoán là yes và 1 mẫu dự đoán là no được phân loại đúng, trong khi có 4 mẫu dự đoán là no và 1 mẫu dự đoán là yes được phân loại sai.

So sánh kết quả:

Dựa trên dữ liệu và kết quả bạn đã cung cấp, chúng ta có thể so sánh hai thuật toán J48 (cây quyết định) và Naive Bayes để phân loại dữ liệu. Dưới đây là một so sánh giữa hai thuật toán này:

1. \*\*J48 (Cây quyết định)\*\*:

- Tỉ lệ phân loại đúng: 64.2857%

- Giá trị Kappa statistic: 0.186

- Độ lỗi tuyệt đối trung bình: 0.2857

- Độ lỗi bình phương trung bình: 0.4818

- Thời gian xây dựng mô hình: 0 giây

- Số lá của cây: 5

- Kết quả được cải thiện sau khi áp dụng cắt tỉa

2. \*\*Naive Bayes\*\*:

- Tỉ lệ phân loại đúng: 64.2857%

- Giá trị Kappa statistic: không có thông tin

- Độ lỗi tuyệt đối trung bình: 0.4649

- Độ lỗi bình phương trung bình: 0.543

- Thời gian xây dựng mô hình: 0 giây

\*\*Nhận xét\*\*:

- Cả hai thuật toán đều có tỉ lệ phân loại đúng tương đương là 64.2857%, dựa trên kết quả của Stratified cross-validation.

- Tuy J48 (cây quyết định) có giá trị Kappa statistic (0.186) cao hơn so với Naive Bayes, nhưng độ lỗi tuyệt đối trung bình và độ lỗi bình phương trung bình của nó (0.2857 và 0.4818) cao hơn so với Naive Bayes (0.4649 và 0.543).

- Thời gian xây dựng mô hình của cả hai thuật toán đều rất nhanh (0 giây).

- J48 đã được cải thiện sau khi áp dụng cắt tỉa để tạo ra cây quyết định cắt tỉa, điều này giúp cải thiện hiệu suất so với cây quyết định ban đầu.

Tóm lại, trong trường hợp này, không có sự khác biệt rõ ràng về hiệu suất giữa J48 và Naive Bayes. Tuy nhiên, cần lưu ý rằng J48 đã được cải thiện sau khi áp dụng cắt tỉa, trong khi Naive Bayes không yêu cầu bất kỳ điều chỉnh nào đặc biệt. Điều này có thể là một điểm mạnh của Naive Bayes nếu không cần thiết phải điều chỉnh thêm.

Bộ dữ liệu diabetes bao gồm các thuộc tính:

**1. Số thuộc tính:**

Bộ dữ liệu Diabetes bao gồm **9 thuộc tính**:

* **6 thuộc tính gốc (thuộc tính đầu vào):**
  + plas: Nồng độ glucose huyết tương (mg/dL)
  + pres: Huyết áp (mmHg)
  + skin: Độ dày nếp da sau gáy (mm)
  + mass: Chỉ số khối cơ thể (kg/m²)
  + age: Tuổi (năm)
  + pedi: Chức năng tế bào beta tụy (0.0 - 1.0)
* **3 thuộc tính dẫn xuất:**
  + BMI: Chỉ số khối cơ thể (tính từ mass và height)
  + diabetes: Chẩn đoán bệnh tiểu đường (0: không mắc, 1: mắc)
  + insulin: Liều insulin (đơn vị)

**2. Thuộc tính mục tiêu:**

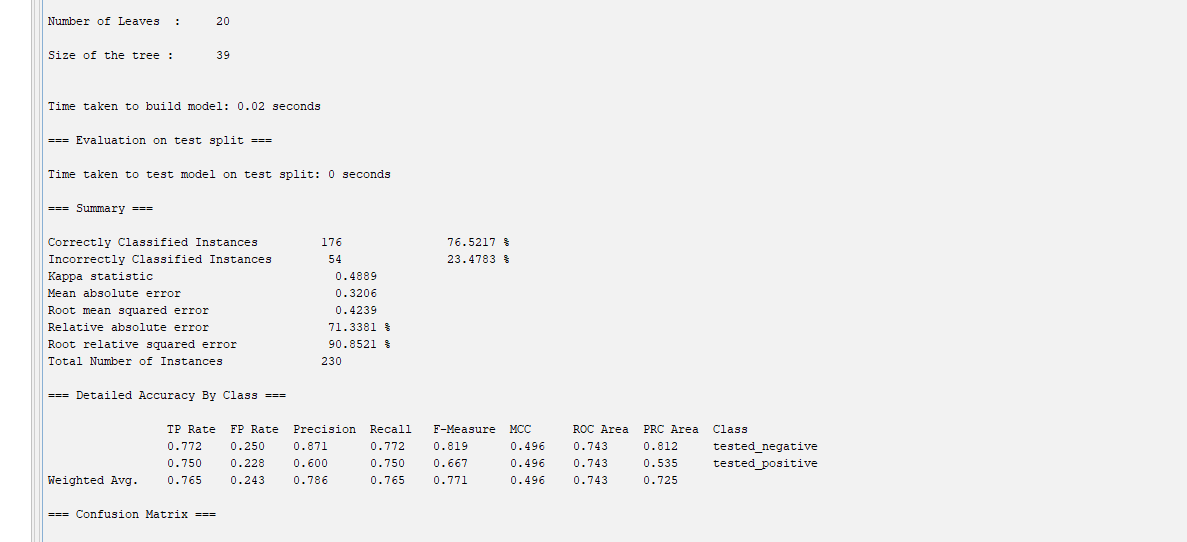
Thuộc tính mục tiêu là **diabetes**, thể hiện trạng thái mắc bệnh tiểu đường của bệnh nhân (0: không mắc, 1: mắc).

**3. Danh sách các thuộc tính:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** |  | **Mô tả** |
| plas | Số |  | Nồng độ glucose huyết tương |
| pres | Số |  | Huyết áp |
| skin | Số |  | Độ dày nếp da sau gáy |
| mass | Số |  | Chỉ số khối cơ thể |
| age | Số |  | Tuổi |
| pedi | Số |  | Chức năng tế bào beta tụy |
| BMI | Số |  | Chỉ số khối cơ thể (tính toán) |
| diabetes | Số |  | Chẩn đoán bệnh tiểu đường (0: không mắc, 1: mắc) |
| insulin | Số |  | Liều insulin (đơn vị) |

**4. Giá trị của từng thuộc tính:**

* **Các thuộc tính gốc:**
  + Có thể có các giá trị khác nhau trong phạm vi hợp lý cho mỗi thuộc tính.
  + Ví dụ: plas có thể nằm trong khoảng 50 - 300 mg/dL, pres có thể nằm trong khoảng 60 - 200 mmHg, v.v.
* **Thuộc tính dẫn xuất:**
  + BMI được tính toán từ mass và height theo công thức: BMI = mass / (height \* height) (m²/kg²).
  + diabetes có giá trị 0 hoặc 1, tương ứng với không mắc hoặc mắc bệnh tiểu đường.
  + insulin có thể có các giá trị khác nhau tùy theo mức độ nghiêm trọng của bệnh tiểu đường và nhu cầu điều trị



Ta nhận được 1 số thuộc tính trong weka :

**Kích Thước và Số Lá Của Cây**

* Số lá (Leaves): 20
* Kích thước của cây: 39

**Xây Dựng và Đánh Giá Mô Hình**

* Thời gian để xây dựng mô hình: 0.02 giây
* Thời gian để kiểm tra mô hình trên tập kiểm tra: 0 giây

**Tóm Tắt Kết Quả Phân Loại**

* + Số lượng mẫu phân loại đúng: 176 (76.5217%)
  + Số lượng mẫu phân loại sai: 54 (23.4783%)
  + Thống kê Kappa: 0.4889
  + Lỗi tuyệt đối trung bình: 0.3206
  + Căn bậc hai của lỗi trung bình: 0.4239
  + Lỗi tuyệt đối tương đối: 71.3381%
  + Lỗi căn bậc hai tương đối: 90.8521%
  + Tổng số mẫu: 230

**Độ Chính Xác Chi Tiết Theo Lớp**

* Tested Negative:

Tỷ lệ dương tính thực (TP Rate): 0.772

Tỷ lệ dương tính giả (FP Rate): 0.250

Độ chính xác: 0.871

Độ nhớ (Recall): 0.772

F-Measure: 0.819

Hệ số tương quan Matthews (MCC): 0.496

Diện tích ROC: 0.743

Diện tích PRC: 0.812

* Tested Positive:

Tỷ lệ dương tính thực (TP Rate): 0.750

Tỷ lệ dương tính giả (FP Rate): 0.228

Độ chính xác: 0.600

Độ nhớ (Recall): 0.750

F-Measure: 0.667

Hệ số tương quan Matthews (MCC): 0.496

Diện tích ROC: 0.743

Diện tích PRC: 0.535

* Trung bình có trọng số:

TP Rate: 0.765

FP Rate: 0.243

Độ chính xác: 0.786

Độ nhớ (Recall): 0.765

F-Measure: 0.771

MCC: 0.496

Diện tích ROC: 0.743

Diện tích PRC: 0.725

**Ma Trận Nhầm Lẫn**

* Tested Negative (a)

Phân loại đúng: 122

Phân loại sai: 36

* Tested Positive (b)

Phân loại đúng: 54

Phân loại sai: 18

**Diễn Giải**

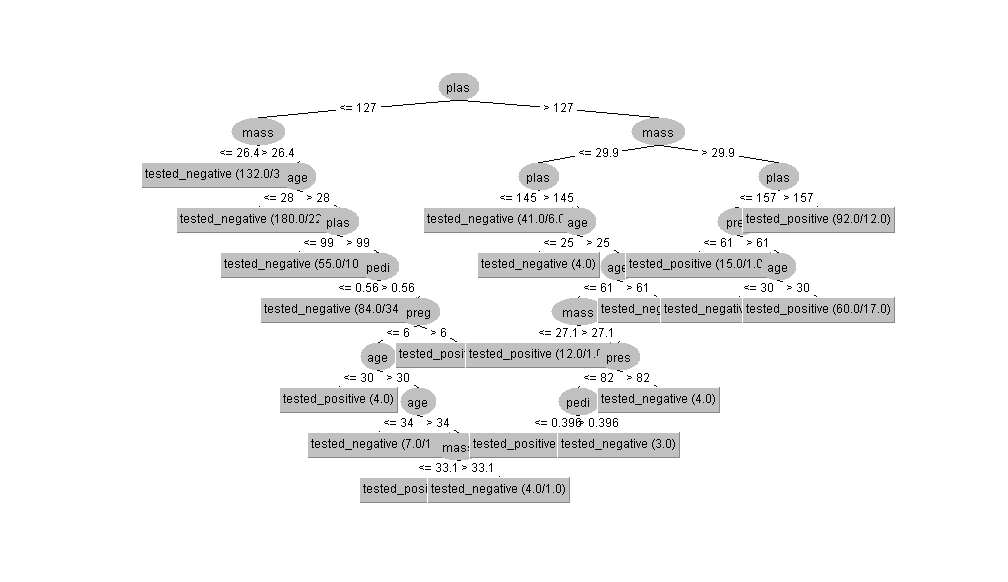
Bộ phân loại có hiệu suất khá tốt đối với lớp tested\_negative (âm tính) với tỷ lệ dương tính thực (TP Rate) là 0.772 và độ chính xác là 0.871.

Hiệu suất đối với lớp tested\_positive (dương tính) cũng khá, với tỷ lệ dương tính thực là 0.750 và độ chính xác là 0.600.

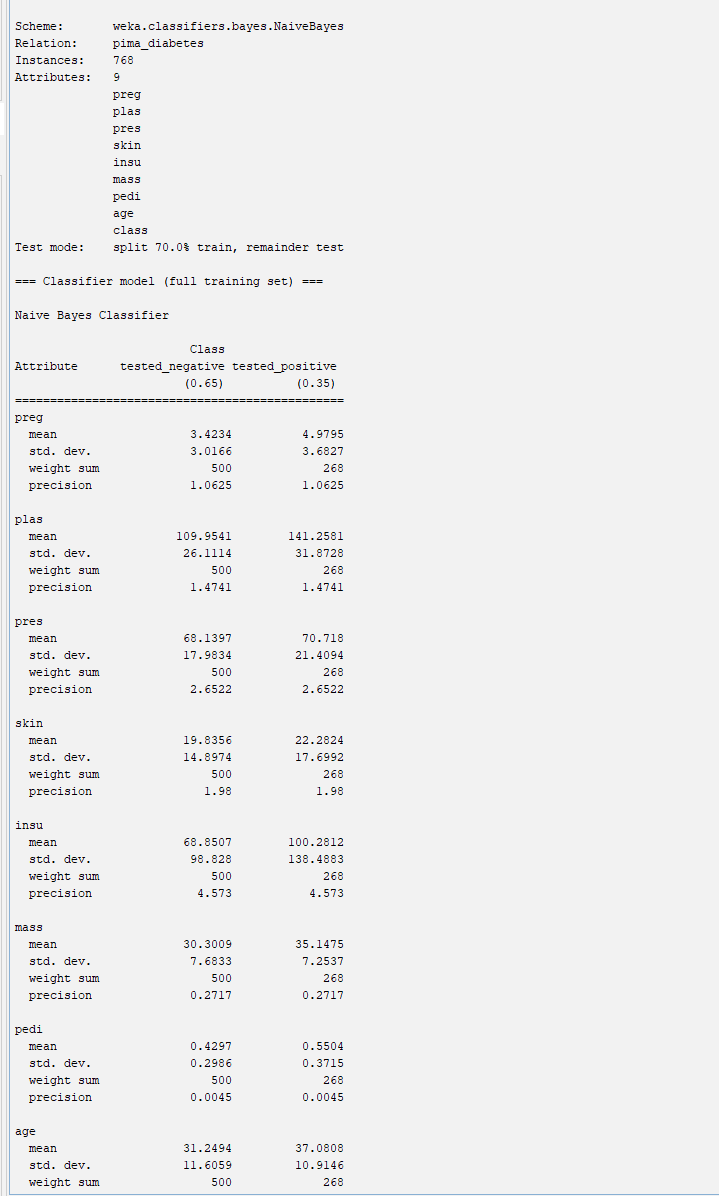
Tổng thể độ chính xác khoảng 76.52%, với thống kê Kappa là 0.4889 cho thấy mức độ đồng thuận vừa phải giữa dự đoán và thực tế.

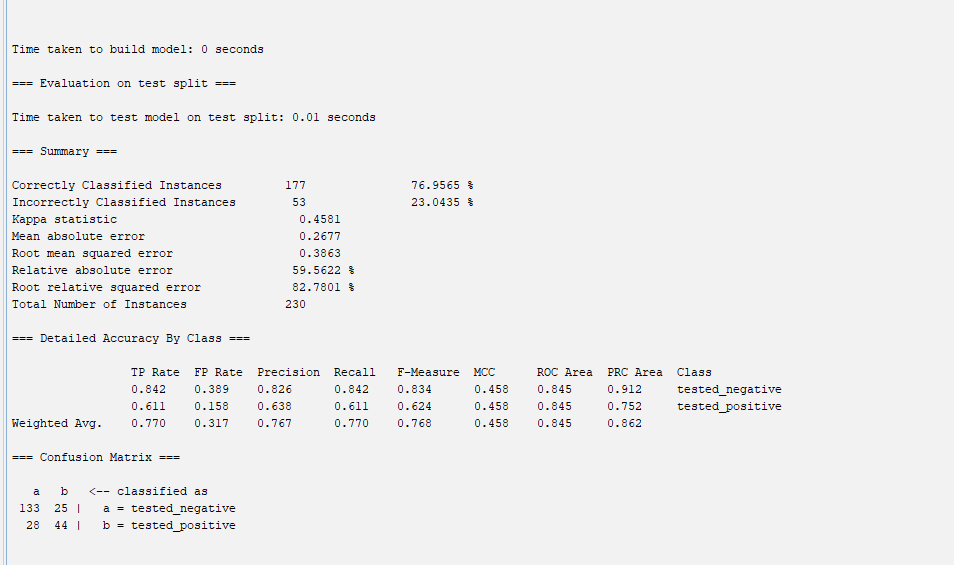
Diện tích ROC 0.743 cho cả hai lớp chỉ ra khả năng phân biệt khá tốt của bộ phân loại.

Ngoài ra ta có thể xem được mô hình cây của giải thuật:



Sau khi chạy giải thuật Bayes





**Preg (Số lần mang thai)**

* Tested Negative

Trung bình: 3.4234

Độ lệch chuẩn: 3.0166

Tổng trọng lượng: 500

Độ chính xác: 1.0625

* Tested Positive

Trung bình: 4.9795

Độ lệch chuẩn: 3.6827

Tổng trọng lượng: 268

Độ chính xác: 1.0625

**Plas (Nồng độ glucose trong huyết tương)**

* Tested Negative

Trung bình: 109.9541

Độ lệch chuẩn: 26.1114

Tổng trọng lượng: 500

Độ chính xác: 1.4741

* Tested Positive

Trung bình: 141.2581

Độ lệch chuẩn: 31.8728

Tổng trọng lượng: 268

Độ chính xác: 1.4741

**Pres (Huyết áp)**

* Tested Negative

Trung bình: 68.1397

Độ lệch chuẩn: 17.9834

Tổng trọng lượng: 500

Độ chính xác: 2.6522

* Tested Positive

Trung bình: 70.718

Độ lệch chuẩn: 21.4094

Tổng trọng lượng: 268

Độ chính xác: 2.6522

**Skin (Độ dày nếp gấp da triceps)**

* Tested Negative

Trung bình: 19.8356

Độ lệch chuẩn: 14.8974

Tổng trọng lượng: 500

Độ chính xác: 1.98

* Tested Positive

Trung bình: 22.2824

Độ lệch chuẩn: 17.6992

Tổng trọng lượng: 268

Độ chính xác: 1.98

**Insu (Insulin trong huyết thanh sau 2 giờ)**

* Tested Negative

Trung bình: 68.8507

Độ lệch chuẩn: 98.828

Tổng trọng lượng: 500

Độ chính xác: 4.573

* Tested Positive

Trung bình: 100.2812

Độ lệch chuẩn: 138.4883

Tổng trọng lượng: 268

Độ chính xác: 4.573

**Mass (Chỉ số khối cơ thể)**

* Tested Negative

Trung bình: 30.3009

Độ lệch chuẩn: 7.6833

Tổng trọng lượng: 500

Độ chính xác: 0.2717

* Tested Positive

Trung bình: 35.1475

Độ lệch chuẩn: 7.2537

Tổng trọng lượng: 268

Độ chính xác: 0.2717

**Pedi (Chức năng phả hệ bệnh tiểu đường)**

* Tested Negative

Trung bình: 0.4297

Độ lệch chuẩn: 0.2986

Tổng trọng lượng: 500

Độ chính xác: 0.0045

* Tested Positive

Trung bình: 0.5504

Độ lệch chuẩn: 0.3715

Tổng trọng lượng: 268

Độ chính xác: 0.0045

**Age (Tuổi)**

* Tested Negative

Trung bình: 31.2494

Độ lệch chuẩn: 11.6059

Tổng trọng lượng: 500

Độ chính xác: 1.1765

* Tested Positive

Trung bình: 37.0808

Độ lệch chuẩn: 10.9146

Tổng trọng lượng: 268

Độ chính xác: 1.1765

**Xây Dựng và Đánh Giá Mô Hình**

Thời gian để xây dựng mô hình: 0 giây

Thời gian để kiểm tra mô hình trên tập kiểm tra: 0.01 giây

**Tóm Tắt Kết Quả Phân Loại**

* Số lượng mẫu phân loại đúng: 177 (76.9565%)
* Số lượng mẫu phân loại sai: 53 (23.0435%)
* Thống kê Kappa: 0.4581
* Lỗi tuyệt đối trung bình: 0.2677
* Căn bậc hai của lỗi trung bình: 0.3863
* Lỗi tuyệt đối tương đối: 59.5622%
* Lỗi căn bậc hai tương đối: 82.7801%
* Tổng số mẫu: 230

**Độ Chính Xác Chi Tiết Theo Lớp**

* Tested Negative:

Tỷ lệ dương tính thực (TP Rate): 0.842

Tỷ lệ dương tính giả (FP Rate): 0.389

Độ chính xác: 0.826

Độ nhớ (Recall): 0.842

F-Measure: 0.834

Hệ số tương quan Matthews (MCC): 0.458

Diện tích ROC: 0.845

Diện tích PRC: 0.912

* Tested Positive:

Tỷ lệ dương tính thực (TP Rate): 0.611

Tỷ lệ dương tính giả (FP Rate): 0.158

Độ chính xác: 0.638

Độ nhớ (Recall): 0.611

F-Measure: 0.624

Hệ số tương quan Matthews (MCC): 0.458

Diện tích ROC: 0.845

Diện tích PRC: 0.752

* Trung bình có trọng số:

TP Rate: 0.770

FP Rate: 0.317

Độ chính xác: 0.767

Độ nhớ (Recall): 0.770

F-Measure: 0.768

MCC: 0.458

Diện tích ROC: 0.845

Diện tích PRC: 0.862

**Ma Trận Nhầm Lẫn**

* Tested Negative (a)

Phân loại đúng: 133

Phân loại sai: 25

* Tested Positive (b)

Phân loại đúng: 44

Phân loại sai: 28

**Diễn Giải**

Bộ phân loại có hiệu suất tốt đối với lớp tested\_negative (âm tính) với tỷ lệ dương tính thực (TP Rate) cao (0.842) và độ chính xác khá (0.826).

Hiệu suất đối với lớp tested\_positive (dương tính) thấp hơn, với tỷ lệ dương tính thực là 0.611 và độ chính xác là 0.638.

Tổng thể độ chính xác khoảng 77%, với thống kê Kappa là 0.4581 cho thấy mức độ đồng thuận vừa phải giữa dự đoán và thực tế.

Diện tích ROC 0.845 cho cả hai lớp chỉ ra khả năng phân biệt tốt của bộ phân loại.

**So sánh số liệu 2 giải thuật ID3 và bayes:**

### So sánh tổng quan

| **Tiêu chí** | **Naive Bayes** | **J48** |
| --- | --- | --- |
| **Thời gian xây dựng mô hình** | 0 giây | 0.02 giây |
| **Thời gian kiểm tra mô hình** | 0.01 giây | 0 giây |
| **Số lượng mẫu phân loại đúng** | 177 (76.96%) | 176 (76.52%) |
| **Số lượng mẫu phân loại sai** | 53 (23.04%) | 54 (23.48%) |
| **Thống kê Kappa** | 0.4581 | 0.4889 |
| **Lỗi tuyệt đối trung bình** | 0.2677 | 0.3206 |
| **Căn bậc hai của lỗi trung bình** | 0.3863 | 0.4239 |
| **Lỗi tuyệt đối tương đối** | 59.56% | 71.34% |
| **Lỗi căn bậc hai tương đối** | 82.78% | 90.85% |
| **Tổng số mẫu** | 230 | 230 |

### Độ chính xác chi tiết theo lớp

| **Tiêu chí** | **Naive Bayes** | **J48** |
| --- | --- | --- |
| **Tested Negative** |  |  |
| Tỷ lệ dương tính thực (TP Rate) | 0.842 | 0.772 |
| Tỷ lệ dương tính giả (FP Rate) | 0.389 | 0.250 |
| Độ chính xác (Precision) | 0.826 | 0.871 |
| Độ nhớ (Recall) | 0.842 | 0.772 |
| F-Measure | 0.834 | 0.819 |
| Hệ số tương quan Matthews (MCC) | 0.458 | 0.496 |
| Diện tích ROC | 0.845 | 0.743 |
| Diện tích PRC | 0.912 | 0.812 |
| **Tested Positive** |  |  |
| Tỷ lệ dương tính thực (TP Rate) | 0.611 | 0.750 |
| Tỷ lệ dương tính giả (FP Rate) | 0.158 | 0.228 |
| Độ chính xác (Precision) | 0.638 | 0.600 |
| Độ nhớ (Recall) | 0.611 | 0.750 |
| F-Measure | 0.624 | 0.667 |
| Hệ số tương quan Matthews (MCC) | 0.458 | 0.496 |
| Diện tích ROC | 0.845 | 0.743 |
| Diện tích PRC | 0.752 | 0.535 |
| **Trung bình có trọng số** |  |  |
| TP Rate | 0.770 | 0.765 |
| FP Rate | 0.317 | 0.243 |
| Độ chính xác (Precision) | 0.767 | 0.786 |
| Độ nhớ (Recall) | 0.770 | 0.765 |
| F-Measure | 0.768 | 0.771 |
| MCC | 0.458 | 0.496 |
| Diện tích ROC | 0.845 | 0.743 |
| Diện tích PRC | 0.862 | 0.725 |

### Ma trận nhầm lẫn

| **Naive Bayes** |  |  |
| --- | --- | --- |
|  | **Tested Negative** | **Tested Positive** |
| **Tested Negative** | 133 | 25 |
| **Tested Positive** | 28 | 44 |

| **J48** |  |  |
| --- | --- | --- |
|  | **Tested Negative** | **Tested Positive** |
| **Tested Negative** | 122 | 36 |
| **Tested Positive** | 18 | 54 |

### So sánh và đánh giá

* **Độ chính xác tổng thể**: Cả hai thuật toán đều có độ chính xác tương đương, với Naive Bayes đạt 76.96% và J48 đạt 76.52%.
* **Thống kê Kappa**: J48 có chỉ số Kappa cao hơn (0.4889) so với Naive Bayes (0.4581), cho thấy mức độ đồng thuận giữa dự đoán và thực tế của J48 cao hơn.
* **Lỗi tuyệt đối và căn bậc hai của lỗi trung bình**: Naive Bayes có lỗi tuyệt đối và căn bậc hai của lỗi trung bình thấp hơn so với J48, cho thấy độ chính xác của dự đoán trung bình cao hơn.
* **Độ chính xác theo lớp**:
  + Với lớp **tested\_negative**, J48 có độ chính xác (Precision) cao hơn (0.871 so với 0.826) nhưng tỷ lệ dương tính thực (TP Rate) thấp hơn (0.772 so với 0.842).
  + Với lớp **tested\_positive**, J48 có tỷ lệ dương tính thực (TP Rate) cao hơn (0.750 so với 0.611) nhưng độ chính xác (Precision) thấp hơn (0.600 so với 0.638).
* **Diện tích ROC và PRC**: Naive Bayes có diện tích ROC và PRC cao hơn so với J48, cho thấy khả năng phân biệt giữa hai lớp của Naive Bayes tốt hơn.

### Kết luận

* **Naive Bayes**: Ưu điểm về khả năng phân biệt tốt hơn (ROC và PRC cao hơn) và lỗi dự đoán thấp hơn.
* **J48**: Ưu điểm về độ chính xác cao hơn trong lớp **tested\_negative** và tỷ lệ dương tính thực cao hơn trong lớp **tested\_positive**, cùng với chỉ số Kappa cao hơn.

**DANH MỤC TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | "A\* Search Algorithm," 07 05 2024. [Online]. Available: https://www.geeksforgeeks.org/a-search-algorithm/. |
| [2] | "A\* search algorithm," [Online]. Available: https://en.wikipedia.org/wiki/A\*\_search\_algorithm. |
| [3] | M. Tamzid, "A\* Search Algorithm," [Online]. Available: https://viblo.asia/p/a-search-algorithm-aWj53BN1l6m. |
| [4] | A. T. o. Pathfinding, "Introduction to A\*," [Online]. Available: theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. |
| [5] | OpenAI, "ChatGPT," 11 2022. [Online]. Available: chatgpt.com. |
| [6] | Google AI, "Gemini," [Online]. Available: gemini.google.com. |