Input = {Customer(x).Has\_Height, Customer(x).Has\_Weight, Customer(x).Has\_Age, Customer(x).Gender, Illness[n], Target, HasActiveLevel(Customer(x), PhysicActiveLevel(act\_level\_input)}

Known = {Height, Weight, Gender, Age, PhysicalActiveLevel, ...}

(Fact, Rule)

Fact:

{

}

Function:

1. FindAvoidFood(): return

Example\_rule:

Note: Thỏa 1 luật thì quay lại vòng lặp luật

{ sk\_giả thiết | sk\_suy diễn } ;

Thỏa sk giả thiết mới xét sk\_suy diễn

[a, b] giải thích:

A =1 và B có thuộc Known => True or else: false

A = 2 -> B = [b1, b2, b3] : b1 thuộc Known, if: b2.value == b3 => True

[a,b] , [a,b] -> [] and []

A : Fact và A phải chưa thuộc known , B : Funct(biến) -> A.value => Append A vào trong Know

Nhãn có nhiều biến: CaloForEachMeal(3, biến);

Note: listavoidfood = (extract(avoidgroupfood))

Rule:

{

{[1, Height], [1,Weight] | [BMIvalue, funct:BMIvalueCalc(height,weight)]},

{[1, BMIvalue] | [BMIrange, funct:BMIrangeCalc(bmi\_value)]},

{[1, BMIrange] | [BMIlevel, funct:BMIlevelCalc(bmi\_range)]},

{[2, Gender, gender, “Nam”)], [1, Height], [1,Weight], [1, Age] | [BMRvalue, funct:BMRvalueOfMaleCalc(height, weight, age)]},

{[2, Gender, gender, “Nữ”], [1, Height], [1,Weight], [1, Age] | [BMRvalue, funct:BMRvalueOfFemaleCalc(height, weight, age)]}

{[1, PhysicalActiveLevel] | [PhysicalActiveValue, funct:PAvalueCalc(pa\_level)]},

{[1, BMRvalue], [1,PhysicalActiveValue], [1, Target] | [CaloPerDay, funct:CaloCalc(bmr\_value, pa\_value,target)]},

{[2, Target, target,“Giữ cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToKeepWeightCalc(calo\_per\_day)]}

{[2, Target, target, “Tăng cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToGainWeightCalc(calo\_per\_day)]}

{[2, Target, target, “Giảm cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToLoseWeightCalc(calo\_per\_day)]}

{[1, ListIllness] | [ListAvoidFood, funct:FindAvoidFood(list\_illness)]},//Tách nhóm groupavoid

{[1, ListIllness] | [ListNeedFood, funct:FindNeedFood(list\_illness)]}//Tách nhóm groupneed

{[1, ListIllness] | [ListLimitFood, funct:FindLimitFood(list\_illness)]}//Tách nhóm grouplimit

{[1, ListAvoidFood], [1, ListAllFood] | [ListFood, funct:FilterFood(list\_illness, list\_food)]},

}

Lập thực đơn:

- Tính BMI

- Tính BMR

- Tính Chỉ số hoạt động

- Tính lượng Calo cần trong 1 ngày

- Tính lượng calo tùy đối tượng

- Tìm danh sách các thực phẩm, nhóm thực phẩm phụ thuộc từng tình trạng, nhu cầu, dị ứng (nếu có)

- Lọc ra các thực phẩm không dùng chung

- Đánh giá mức độ quan trọng của từng thực phẩm đối với nhu cầu cũng như tình trạng

- Tùy vào độ ưu tiên, chọn ra các danh sách thực đơn

- Thuwcj

- Tính định lượng từng thực phẩm trong mỗi thực đơn dựa vào tổng calo cần thiết.

- Xếp hạng cho các thực đơn

List of food (food\_name: score (1,0,-1)

Limit\_food\_and\_their\_calo(rau = 20-30 calo)

Món ăn vào menu:

Có: Calo - phương sai

List of food - nb

Món\_fix - calo đi cùng (example: rau - 50calo)

Cần: Buổi sáng: rule = random(list\_of\_rule(name\_food, portion\_food))

Food\_portion = calo/(

Buổi trưa:

Rule = {random(rau, 2), random(meat, seafood, egg\_products; 1-2), randome(fruits, yes\_no)}

Buổi tối:

Rule = {}

[1,ListAvoidFood],[1,ListNeedFood],[1,ListLimitFood],[1,ListAllFood]|[ListFood,funct:FilterFood(known["ListAvoidFood"], known["ListNeedFood"], known["ListLimitFood], known["ListAllFood"])]

* Hàm FilterFood tạo ListFood bằng cách loại bỏ các phần từ ListAvoidFood ra khỏi ListAllFood, và đánh giá score các thực phẩm còn lại trong ListAllFood dựa trên các ListNeedFood, ListLimitFood

Cấu trúc ListAllFood được lấy ra từ đầu gồm các thuộc tính [name, type, calo/100gr], ListFood được tạo ra từ hàm trên có cấu trúc giống ListAllFood nhưng thêm một thuộc tính score được đánh giá.

[1,ListFood],[ListStructureOfBreakFast]|[ListBreakFast,funct:FindListBreakFast(known["ListStructureOfBreakFast"])]

[1,ListFood],[ListStructureOfLunch]|[ListLunch,funct:FindListLunch(known["ListStructureOfLunch"])]

[1,ListFood],[ListStructureOfDinner]|[ListDinner,funct:FindListDinner(known["ListStructureOfDinner"])]

* Các ListStructureOf… được tạo ra trong known ngay từ đầu bằng truy xuất csdl. Các hàm FindList… dựa trên ListstructureOf… để tạo ra các List thực phẩm theo loại trong structure để tìm ra các cách khẩu phần ăn khác nhau , lựa chọn các thực phẩm ngẫu nhiên trong list loại được tạo ra ở trên.

Ví Dụ: Cấu trúc bữa sáng [0.2,Fruit,2],[0.6,Eggs,1] -> tạo các ListFruit, ListEggs, ListVegetable dựa trên ListFood. Chọn thực phẩm cho loại đầu tiên trong cấu trúc thực phẩm đầu tiên trong cấu trúc: Chọn 3 fruit trong ListFruit có score cao nhất, rồi lấy tổ hợp C(k,3) để được một tập hợp cho loại thứ nhất trong cấu trúc, vd: Cần chọn 2 Fruit, lấy ra 3 fruit có score cao nhất [Chuối, Xoài, Ổi], lấy tổ hợp chập 2 C(2,3) ta được tập hợp {{Chuối,Xoài}, {Chuối, Ổi}, {Xoài, Ổi}}, tính số gram(trái) dựa theo (CaloForBreakfast\*tỷ lệ)/Calo chuối rồi lưu lại. Tiếp tục lặp cho các phẩn tử kế tiếp trong cấu trúc. Chọn 3 food có score cao nhất trong ListEggs, ví dụ [Trứng gà, Trững vịt, Trứng ngỗng] Chọn C(1,3) ta được {{Trứng gà}, {Trững vịt}, {Trứng ngỗng}}. Sau khi tìm hết các phần tử trong cấu trúc, ta được các tập hợp khẩu phầnn ăn bằng cách ghép các tập trên lại với nhau:  
{Chuối,Xoài,Trứng gà},{Chuối, Xoài, Trứng vịt}…. Thực hiện loại bỏ các khẩu phần có chứa những thực phẩm không dùng chung, ví dụ Chuối và Trứng vịt không dùng chung, ta loại bỏ các tập có chứa 2 thực phẩm này {Chuối, Xoài, Trứng vịt}….

Các bữa trưa, chiều làm tương tự.

[1,ListBreakFast],[1,ListLunch],[1,ListDinner]|[ListMenu,funct:JoinMeals(known["ListBreakFast"],known["ListLunch"],known["ListDinner"])]

* Ghép các tập hợp ListBreakFast, ListLunch, ListDinner để được các ListMeNu, đánh giá từng Menu dựa trên score của các thực phẩm trong Menu, lưu ý các thực phẩm giống nhau chỉ tính điểm 1 lần, ví dụ Chuối xuất hiện trong cả bữa trưa và sáng, nhưng vẫn tính là xuất hiện 1 lần.
* Chọn top n menu có xếp hạng cao nhất để xuất ra.