BÀI THỰC HÀNH – CSTTNT – BUỔI 8
PHẦN 1- DECISION TREE

I. LÝ THUYẾT

Cây quyết định là lớp các giải thuật học

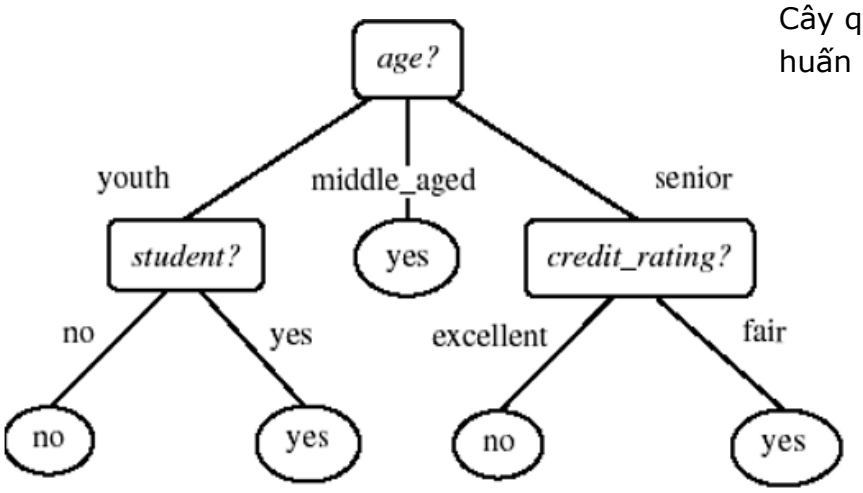
- biểu diễn trực quan (cây) của thủ tục phân lớp
- kết quả sinh ra dễ dịch (luật dạng if ... then ...)
- khá đơn giản, nhanh, hiệu quả được sử dụng nhiều
- liên tục trong nhiều năm qua, cây quyết định được bình chọn là giải thuật được sử dụng nhiều nhất và thành công nhất
- giải quyết các vấn đề của phân loại, hồi quy
- làm việc cho dữ liệu kiểu số và rời rạc
- được ứng dụng thành công trong hầu hết các lãnh vực về phân tích dữ liệu, phân loại text, spam, phân loại gien, etc
- có rất nhiều giải thuật sẵn dùng : C4.5 (Quinlan, 1993), CART (Breiman et al., 1984), . .

VD: CSDL huấn luyện *AllElectronics* học có các thuộc tính (age, income, student, credit _rating), quyết định/phân lớp (yes/no)

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- Cây quyết định (decision tree) – mô hình phân loại
  - Node nội: phép kiểm thử (test) trên một thuộc tính
  - Node lá: nhãn/mô tả của một lớp (class label)
  - Nhánh từ một node nội: kết quả của một phép thử trên thuộc tính tương ứng

Cây quyết định học được từ CSDL huấn luyện *AllElectronics*



.Giải thuật cây quyết định

- xây dựng cây Top-down
  - bắt đầu nút gốc, tất cả các dữ liệu học ở nút gốc
  - phân hoạch dữ liệu một cách đệ quy bằng việc chọn 1 thuộc tính để thực hiện phân hoạch tốt nhất có thể
- cắt nhánh Bottom-up
  - cắt những cây con hoặc các nhánh từ dưới lên trên, để tránh học vẹt (overfitting, over learning)
- ở mỗi nút, các thuộc tính được đánh giá dựa trên phân tách dữ liệu học tốt nhất có thể
- việc đánh giá dựa trên
  - độ lợi thông tin, information gain (ID3/C4.5)
  - information gain ratio
  - chỉ số gini, gini index (CART)

Giải thuật:

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Đặc điểm của giải thuật

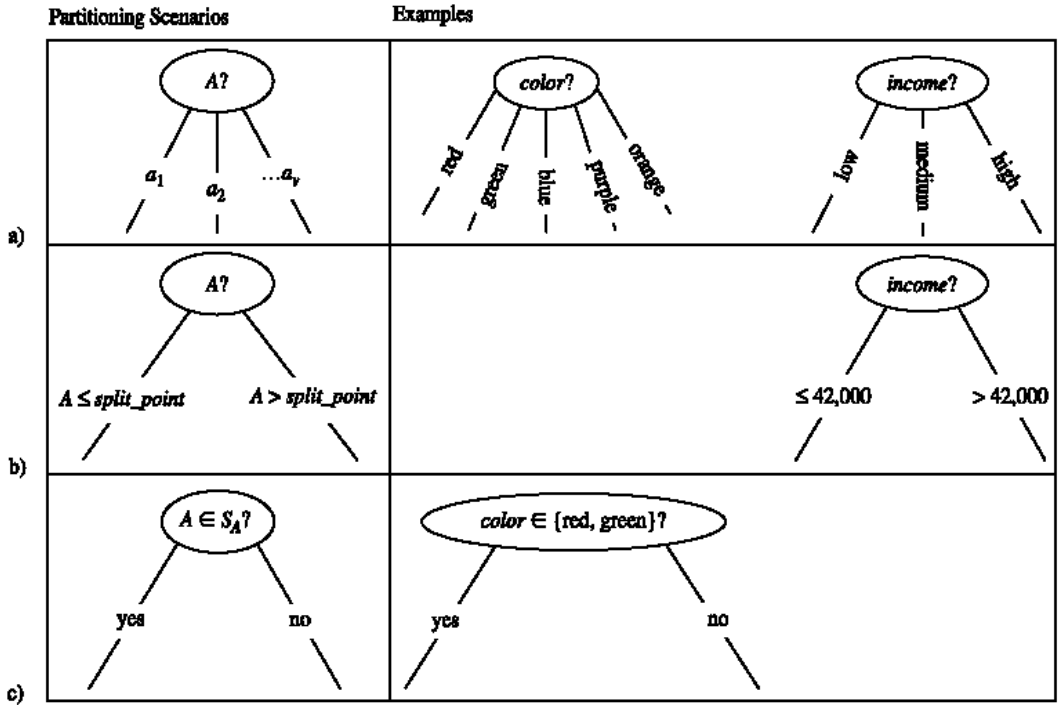
- Giải thuật tham lam (không có quay lui), chia để trị, đệ qui, từ trên xuống
- Độ phức tạp với tập huấn luyện *D* gồm |*D*| phần tử (đối tượng), mỗi phần tử gồm *n* thuộc tính
  - O(*n**|*D*|**log*|*D*|)
    - Mỗi thuộc tính ứng với mỗi mức (level) của cây.
    - Cho mỗi mức của cây, |D| phân tử huấn luyện được duyệt qua.

Chọn thuộc tính phân hoạch ?

- thuộc tính nào tốt ?
  - cho ra kết quả là cây nhỏ nhất
  - heuristics: chọn thuộc tính sinh ra các nút "purest" (thuần khiết nhất – độ hỗn loạn thông tin thấp nhất)
- độ lợi thông tin

- tăng theo giá trị trung bình thuần khiết của các tập con của dữ liệu sau khi thực hiện phân hoạch
- độ hỗn loạn trước khi phân hoạch –độ hỗn loạn sau khi phân hoạch

- chọn thuộc tính có độ lợi thông tin lớn nhất



A là thuộc tính phân tách (splitting attribute).

Độ đo Information Gain

- **Dựa trên lý thuyết thông tin (information theory) của Claude Shannon về giá trị (nội dung thông tin) của tin**
- **Thuộc tính tương ứng với information gain lớn nhất sẽ được chọn làm splitting attribute cho node N.**
  - **Node N là node hiện tại cần phân hoạch các phần tử trong D.**
  - **Splitting attribute đảm bảo sự trùng lắp (impurity)/ngẫu nhiên (randomness) ít nhất giữa các phân hoạch tạo được.**
  - **Cách tiếp cận này giúp tối thiểu số phép thử (test) để phân loại một phần tử.**
- **Lượng thông tin cần để phân loại một phần tử trong D (= Entropy của D): Info(D)**

- p_i: xác suất để một phần tử bất kỳ trong D thuộc về lớp C_i với i = 1..m
- $C_{i,D}$: tập các phần tử của lớp C_i trong D

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$p_i = |C_{i,D}| / |D|$$

- Lượng thông tin cần để phân loại một phần tử trong D dựa trên thuộc tính A: $Info_A(D)$
  - Thuộc tính A dùng phân tách D thành v phân hoạch $\{D_1, D_2, \ldots, D_j, \ldots, D_v\}$.
  - Mỗi phân hoạch D_j gồm $|D_j|$ phần tử trong D.
  - Lượng thông tin này sẽ cho biết mức độ trùng lắp giữa các phân hoạch, nghĩa là một phân hoạch chứa các phần tử từ một lớp hay nhiều lớp khác nhau.
  - Mong đợi: $Info_A(D)$ càng nhỏ càng tốt.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} * Info(D_j)$$

- Information gain chính là độ sai biệt giữa trị thông tin Info(D) ban đầu (trước phân hoạch) và trị thông tin mới $Info_A(D)$ (sau phân hoạch với A).

$$Gain(A) = Info(D) - Info_A(D)$$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- Class P:buys_computer="yes"
- Class N:buys computer="no"

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

Gain (age,D) = Info(D) – $Info_{age}(D)$

= Info(D) - $\sum_{D_j \in \{Youth, Midle-aged, senior\}} \frac{|D_j|}{|D|} * Info(D_j)$

= Info(D) - $\frac{5}{14} Info(D_{youth})$ - $\frac{4}{14} Info(D_{middle-aged})$ - $\frac{5}{14} Info(D_{senior})$

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694 \text{ bits.}$$

Gain(age)=0.246 bits

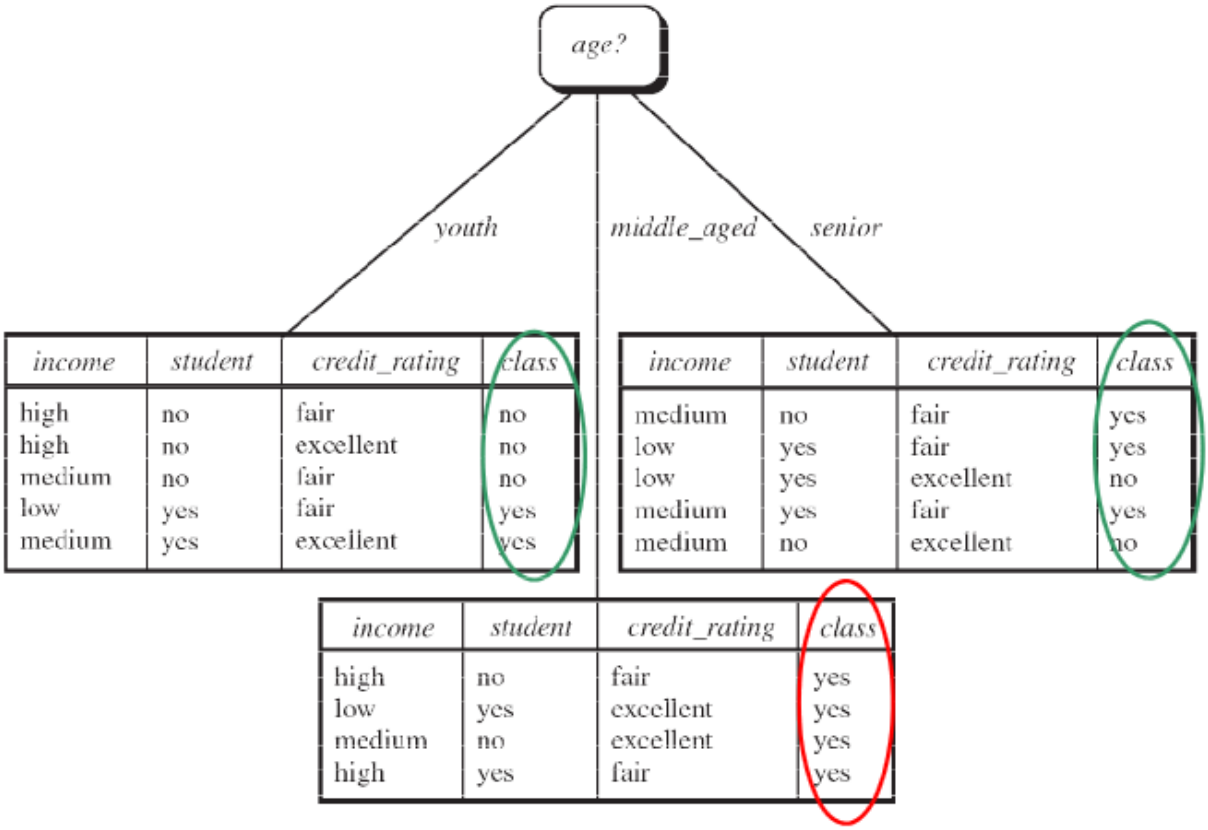
Gain(income)=0.029

Gain(student)=0.151

Gain(credit_rating)=0.048

→ Splitting attribute = Age

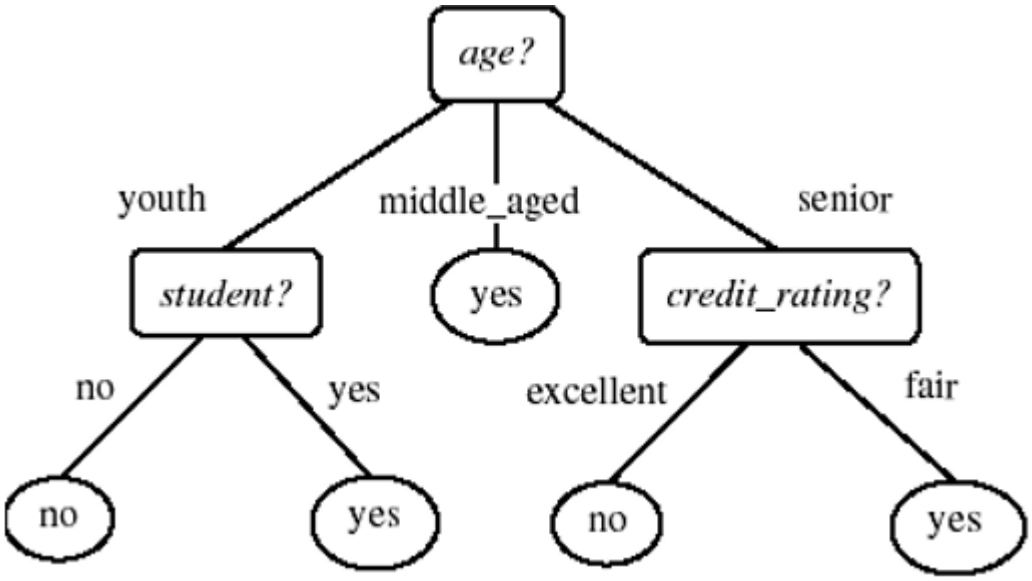
$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



II. THỰC HÀNH

Bài 1. Tập dữ liệu nhận biết người bị bệnh tiểu đường (diabetes)

1. Tạo file .csv bằng Excel có nội dung như sau:

	A	B	C	D	E	F	G	H	I
1	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1

Trong đó có các thuộc tính (pregnant, glucose, bp, skin, insulin, bmi, pedigree, age) và phân lớp label. Sau đó lưu với tên và định dạng file là diabetes.csv lưu ở ổ đĩa D

2. Cài đặt các thư viện sau

pip install sklearn

pip install pandas

3. Code của Decision Tree trong Python

- Chia tập dữ liệu thử thành 2 tập dữ liệu: 70% training data và 30% là test data

```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
# load dataset
pima = pd.read_csv("D:\\diabetes.csv", header=0, names=col_names)
pima.head()

#split dataset in features and target variable
feature_cols = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age']
X = pima[feature_cols] # Features
y = pima.label # Target variable

print(pima[feature_cols])
print(pima.label)

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print(y_pred)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Bài 2. Tạo file .csv trong Excel như sau:

	A	B	C	D	E
1	gender	carownership	travelcost	incomelevel	transportationmode
2	Male	0	Cheap	Low	Bus
3	Male	1	Cheap	Medium	Bus
4	Female	1	Cheap	Medium	Train
5	Female	0	Cheap	Low	Bus
6	Male	1	Cheap	Medium	Bus
7	Male	0	Standard	Medium	Train
8	Female	1	Standard	Medium	Train
9	Female	1	Expensive	High	Car
10	Male	2	Expensive	Medium	Car
11	Female	2	Expensive	High	Car

Viết giải thuật Decision tree để huấn luyện tập trên và trả lời tập test

Female	2	Expensive	Medium	?
Male	1	Standard	Medium	?

- **Gợi ý: dùng preprocessing của sklearn để chuyển đổi các thông tin thành các con số cụ thể. Vd: female tương ứng với 1 và male tương ứng với 0 vì giải thuật chỉ làm việc trên kiểu dữ liệu là các con số cụ thể**

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn import preprocessing

col_names = ['gender', 'carownership', 'travelcost', 'incomelevel','transportationmode']
# load dataset
pima = pd.read_csv("D:\\transport.csv", header=0, names=col_names)
pima.head()

lE = preprocessing.LabelEncoder()
data = pima.apply(lE .fit_transform)
print(data)

#split dataset in features and target variable
feature_cols = ['gender', 'carownership', 'travelcost', 'incomelevel']
X = data[feature_cols] # Features
y = data.transportationmode # Target variable
#print(data[feature_cols])
#print(data.transportationmode)

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#print(X_train)
#print(X_test)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
#print(y_pred)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

d1=np.array([[0,2,1,2]])
d1_pred = clf.predict(d1)
print(d1_pred)
ketqua=lE.inverse_transform(d1_pred)
print(ketqua)
```