## **2. Bài toán phân lớp dùng để phân loại nấm độc hay nấm ăn được sử dụng kỹ thuật cây quyết định (Decision Trees)**

### **2.1 Mô tả bộ dữ liệu**

**Nguồn gốc:**  
  
Hồ sơ về nấm được rút ra từ Hướng dẫn thực địa về nấm ở Bắc Mỹ của Hiệp hội Audubon (1981). GH Lincoff (Pres.), New York: Alfred A. Knopf  
  
**Nhà tài trợ:**  
  
Jeff Schlimmer ( Jeffrey.Schlimmer **'@'** a.gp.cs.cmu.edu )

**Thông tin tập dữ liệu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Đặc điểm của Tập dữ liệu:** | Đa biến | **Số phiên bản:** | 8124 | **Khu vực:** | Đời sống |
| **Đặc điểm thuộc tính:** | Phân loại | **Số thuộc tính:** | 22 | **Ngày được tặng** | 1987-04-27 |
| **Nhiệm vụ liên quan:** | Phân loại | **Giá trị bị mất?** | Đúng | **Số lượt truy cập web:** | 592196 |

Bộ dữ liệu này bao gồm các mô tả về các mẫu giả định tương ứng với 23 loài nấm mang trong họ Agaricus và Lepiota (trang 500-525). Mỗi loài được xác định là chắc chắn ăn được, chắc chắn có độc, hoặc không rõ có thể ăn được và không được khuyến khích. Lớp sau này được kết hợp với lớp độc. Hướng dẫn nêu rõ rằng không có quy tắc đơn giản nào để xác định khả năng ăn được của nấm

**Thông tin thuộc tính (classes: edible=e, poisonous=p):**

1. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s

2. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s

3. cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y

4. bruises: bruises=t, no=f

5. odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s

6. gill-attachment: attached=a, descending=d, free=f, notched=n

7. gill-spacing: close=c, crowded=w, distant=d

8. gill-size: broad=b, narrow=n

9. gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y

10. stalk-shape: enlarging=e, tapering=t

11. stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?

12. stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s

13. stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s

14. stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

15. stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

16. veil-type: partial=p, universal=u

17. veil-color: brown=n, orange=o, white=w, yellow=y

18. ring-number: none=n, one=o, two=t

19. ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

20. spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y

21. population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y

22. habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

### **2.2 Mô tả bài toán**

Bài toán sử dụng bộ dữ liệu các thuộc tính vật lý nấm để phân loại ra 2 loại nấm ăn được và nấm độc. Sử dụng bài toán học máy phân lớp. Áp dụng ký thuật Logistic Regression

### **2.3 Mô tả chi tiết về cách xử lý dữ liệu**

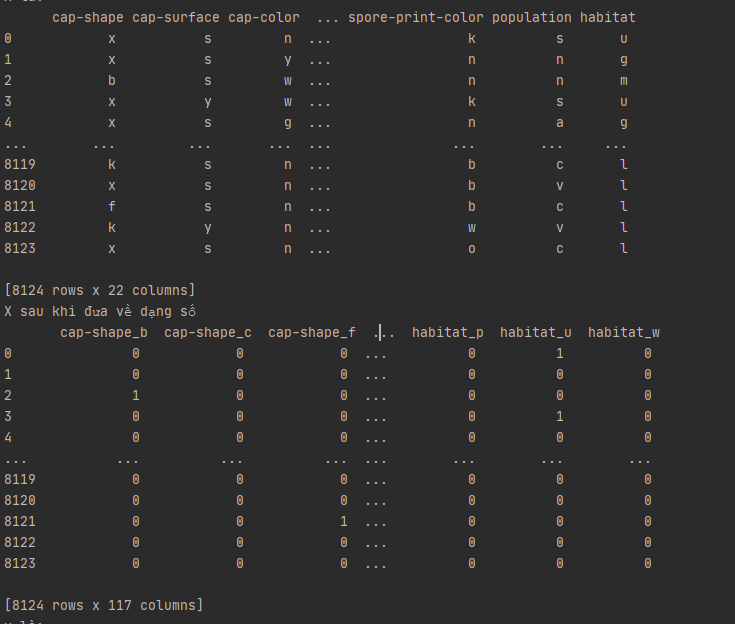
Trong quá trình thu thật dữ liệu do dữ liệu quá lớn nên cũng không tránh khỏi mất mát 1 số dữ liệu để giải quyết vẫn đề này thì chúng ta sử dụng hàm dropna():



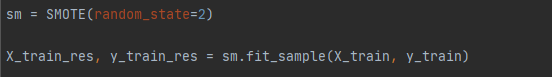
Vì bộ dữ liệu là dạng chữ nên 1 số chữ không thể chạy được trong học máy nên chúng ta cần đưa bộ dữ liệu về dạng số bằng hàm get\_dummies() trong thư viện pandas:



Kết quả trước và sau khi đưa về dạng số:



Sau khi chia train và test xong để tránh khỏi việc mất cân bằng dữ liệu thì chúng ta dùng hàm SMOTE của thư viện imblearn:



Kết quả:





### **2.4 Mô tả kỹ thuật Decision Trees**

#### **2.4.1 Giới thiệu**

Decision tree là một mô hình supervised learning, có thể được áp dụng vào cả hai bài toán classification và regression. Việc xây dựng một decision tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của decision tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về decision tree, các đặc trưng thường được gọi là thuộc tính – attribute) dạng categorical, thường là rời rạc và không có thứ tự. Ví dụ, mưa, nắng hay xanh, đỏ, v.v. Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục (numeric). Một điểm đáng lưu ý nữa là decision tree ít yêu cầu việc chuẩn hoá dữ liệu.

Chúng ta sẽ làm quen với một thuật toán xây dựng decision tree ra đời từ rất sớm và rất phổ biến: [Iterative Dichotomiser 3 (ID3)](https://en.wikipedia.org/wiki/ID3_algorithm).

ID3 là một thuật toán decision tree được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical. Trong bài tiếp theo, chúng ta sẽ làm quen với một thuật toán khác có tên là Classification and Regression Tree (CART)–có thể được áp dụng vào cả hai loại classification và regression, như tên gọi của nó–làm việc với cả thuộc tính dạng categorical và liên tục.

#### **2.4.2 Ý tưởng**

Trong ID3, chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau, việc tìm được nghiệm tối ưu thường là không khả thi. Thay vào đó, một phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó (chúng ta sẽ bàn sớm). Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như thế này được gọi là cách chọn greedy (tham lam). Cách chọn này có thể không phải là tối ưu, nhưng trực giác cho chúng ta thấy rằng cách làm này sẽ gần với cách làm tối ưu. Ngoài ra, cách làm này khiến cho bài toán cần giải quyết trở nên đơn giản hơn.

Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây chính là một thuộc tính, câu trả lời chính là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo.

Trước hết, thế nào là một phép phân chia tốt? Bằng trực giác, một phép phân chia là tốt nhất nếu dữ liệu trong mỗi child node hoàn toàn thuộc vào một class–khi đó child node này có thể được coi là một leaf node, tức ta không cần phân chia thêm nữa. Nếu dữ liệu trong các child node vẫn lẫn vào nhau theo tỉ lệ lớn, ta coi rằng phép phân chia đó chưa thực sự tốt. Từ nhận xét này, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi child node nằm trong cùng một class (tinh khiết nhất), và cho giá trị cao nếu mỗi child node có chứa dữ liệu thuộc nhiều class khác nhau.

Một hàm số có các đặc điểm này và được dùng nhiều trong lý thuyết thông tin là hàm entropy.

#### **2.4.3. Hàm số entropy**

Cho một phân phối xác suất của một biến rời rạc xx có thể nhận nn giá trị khác nhau x1,x2,…,xnx1,x2,…,xn.

Giả sử rằng xác suất để xx nhận các giá trị này là pi=p(x=xi)pi=p(x=xi) với 0≤pi≤1,∑ni=1pi=10≤pi≤1,∑i=1npi=1. Ký hiệu phân phối này là p=(p1,p2,…,pn)p=(p1,p2,…,pn). Entropy của phân phối này được định nghĩa là

H(p)=−n∑i=1pilog(pi)(1)

trong đó loglog là logarit tự nhiên (Một số tài liệu dùng logarit cơ số 2, nhưng giá trị của H(p)H(p) chỉ khác đi bằng cách nhân với một hằng số.) và quy ước 0log(0) =0.

Xét một ví dụ với n=2n=2 được cho trên Hình 3. Trong trường hợp pp là tinh khiết nhất, tức một trong hai giá trị pipi bằng 1, giá trị kia bằng 0, entropy của phân phối này là H(p)=0H(p)=0. Khi pp là vẩn đục nhất, tức cả hai giá trị pi=0.5pi=0.5, hàm entropy đạt giá trị cao nhất.

|  |  |
| --- | --- |
|  | Hình 3: Đồ thị của hàm entropy với n=2n=2. |

Tổng quát lên với n>2n>2, hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị pi=1pi=1, đạt giá trị lớn nhất nếu tất cả các pipi bằng nhau (việc này có thể được chứng minh bằng [phương pháp nhân tử Lagrange](https://machinelearningcoban.com/2017/04/02/duality/#--phuong-phap-nhan-tu-lagrange)).

Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vẩn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

#### **2.4.4. Thuật toán ID3**

Trong ID3, tổng có trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một non-leaf node. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này.

Xét một bài toán với CC class khác nhau. Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập SS với số phần tử là |S|=N|S|=N. Giả sử thêm rằng trong số NN điểm dữ liệu này, Nc,c=1,2,…,CNc,c=1,2,…,C điểm thuộc vào class cc. Xác suất để mỗi điểm dữ liệu rơi vào một class cc được xấp xỉ bằng NcNNcN (maximum likelihood estimation). Như vậy, entropy tại node này được tính bởi:

H(S)=−C∑c=1NcNlog(NcN)(2)

Tiếp theo, giả sử thuộc tính được chọn là xx. Dựa trên xx, các điểm dữ liệu trong SS được phân ra thành KK child node S1,S2,…,SKS1,S2,…,SK với số điểm trong mỗi child node lần lượt là m1,m2,…,mKm1,m2,…,mK. Ta định nghĩa

H(x,S)=K∑k=1mkNH(Sk)(3)

là tổng có trọng số entroy của mỗi child node–được tính tương tự như (2). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau.

Tiếp theo, ta định nghĩa information gain dựa trên thuộc tính xx:

G(x,S)=H(S)−H(x,S)

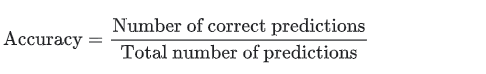
Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên:

x∗=argmaxxG(x,S)=argminxH(x,S)

tức thuộc tính khiến cho information gain đạt giá trị lớn nhất.

### **2.5. Phương pháp đánh giá Accuracy( Độ chính xác)**

Cách đơn giản và hay được sử dụng nhất là accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.



Đối với phân loại nhị phân, độ chính xác cũng có thể được tính theo mặt tích cực và tiêu cực như sau:



Trong đó TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.