Chương 2 Phân loại lớp thú

2.1 Mô tả bài toán

Tập dữ liệu này bao gồm 101 loài động vật từ vườn thú.  
Có 16 biến với các đặc điểm khác nhau để mô tả các loài động vật.  
7 Loại Lớp là: Động vật có vú, Chim, Bò sát, Cá, Lưỡng cư, Bọ và Động vật không xương sống

Mục đích của tập dữ liệu này là có thể dự đoán phân loại của các loài động vật, dựa trên các biến số.

2.2 Mô tả dữ liệu và mô tả bài toán học máy

Trong bài toán phân loại thu vật này, em đã tìm hiểu và thu thập đươc tập dữ liệu các đặc điểm của con vật. Tập dữ liệu nói trên được em thu thập từ **https://www.kaggle.com/uciml/zoo-animal-classification.** Tập dữ liệu bao gồm 101 dòng mẫu với 18 cột đặc điểm của con vật.

2.2.1 Mô tả dữ liệu

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | Animal\_name | Tên của con vật |
| 2 | Hair | Boolean (Có lông hay không) |
| 3 | Feathers | Boolean (Có lông vũ hay không) |
| 4 | eggs | Boolean ( Có đẻ trứng hay không) |
| 5 | milk | Boolean (Có sữa hay không) |
| 6 | airborne | Boolean (Trên không có hay không) |
| 7 | aquatic | Boolean ( Sống dưới nước có haykhông ) |
| 8 | predator | Boolean ( kẻ săn mồi có haykhông ) |
| 9 | toothed | Boolean ( Có răng hay không) |
| 10 | backbone | Boolean( có xương sống hay không ) |
| 11 | Breathes | Boolean |
| 12 | venomous | Boolean ( có nọc độc hay không) |
| 13 | fins | Boolean ( có vảy hay không ) |
| 14 | legs | Numeric (bộ giá trị: {0,2,4,5,6,8}) |
| 15 | tail | Boolearn (có đuôi hay không) |
| 16 | domestic | Boolearn ( có trong nước) |
| 17 | catsize | Boolearn |
| 18 | class\_type | Có 7 lớp động vật |

2.2.2 Mô tả bài toán học máy

Bài toán phân loại các lớp động vật trong vườn bách thú

Bài toán sử dụng là phân lớp nhiều lớp với kỹ thuật cây quyết định và phương pháp đánh giá Accuracy score và Confusion matrix.

2.3 Bài toán

2.3.1 Xử lý dữ liệu

- Xử lý các cột không sử dụng : Animal\_name

- Chia X, Y với

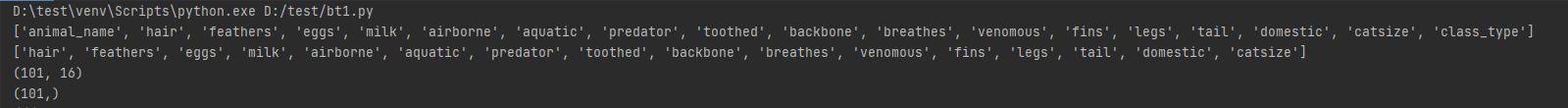
+ X : Là các thuộc tính Hair đến Catsize

+ Y : Là thuộc tính class\_type

Code :

df = pd.read\_csv('zoo.csv')  
df.head()  
features = list(df.columns)  
print(features)  
features.remove('class\_type')  
features.remove('animal\_name')  
  
  
print(features)  
X = df[features].values.astype(np.float32)  
Y = df.class\_type  
  
  
print(X.shape)  
print(Y.shape)  
print("///")

Kết quả :

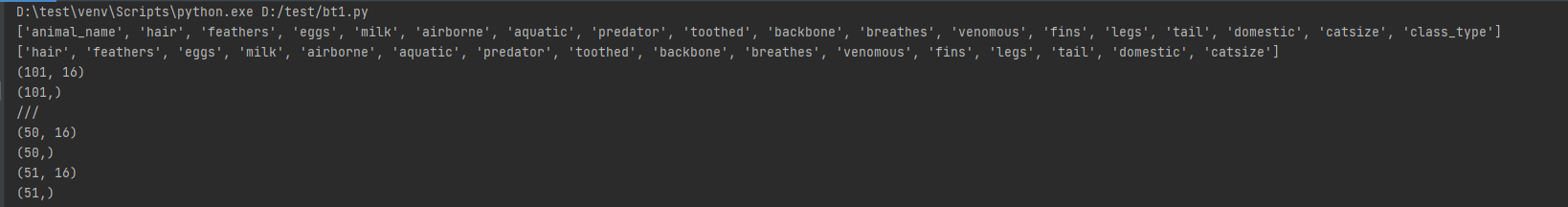


-Chia tập train với tập test : chia tập dữ liệu ra thành 2 phần với tỷ lệ 50:50

Code:

from sklearn.model\_selection import train\_test\_split  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.5, random\_state = 0)  
  
print(X\_train.shape)  
print(Y\_train.shape)  
print(X\_test.shape)  
print(Y\_test.shape)

Kết quả



2.3.2 Kỹ thuật học máy cây quyết định

Cây quyết định ([Decision Tree](https://trituenhantao.io/tu-dien-thuat-ngu/decision-tree/)) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

Cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

Thuật toán ID3 sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) và Information Gain để xây dựng một cây quyết định.

Entropy là thuật ngữ thuộc Nhiệt động lực học, là thước đo của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Năm 1948, Shannon đã mở rộng khái niệm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sang lĩnh vực nghiên cứu, thống kê với công thức như sau:

Với một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,…,xn.

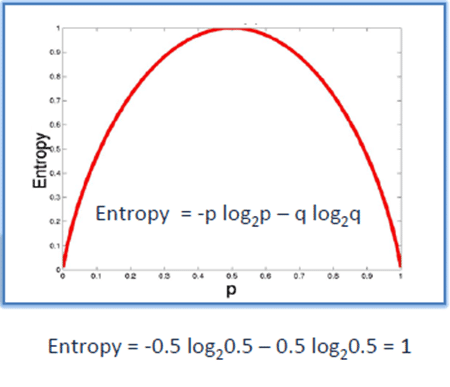
Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi).

Ký hiệu phân phối này là p=(p1 ,p2 ,…,pn). [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của phân phối này được định nghĩa là:

H(p)=  – ∑nn=1 pi log(pi)

Giả sử bạn tung một đồng xu, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sẽ được tính như sau:

H = -[0.5 ln(0.5) + 0.5 ln(0.5)]

Hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)

Hình vẽ trên biểu diễn sự thay đổi của hàm [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/). Ta có thể thấy rằng, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt tối đa khi xác suất xảy ra của hai lớp bằng nhau.

* P tinh khiết: pi = 0 hoặc pi = 1
* P vẩn đục: pi = 0.5, khi đó hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt đỉnh cao nhất

#### Information Gain**trong Cây quyết định (Decision Tree)**

Information Gain dựa trên sự giảm của hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) khi tập dữ liệu được phân chia trên một thuộc tính. Để xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Infomation gain cao nhất.

Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

•**Bước 1**: Tính toán hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của biến mục tiêu S có N phần tử với Nc phần tử thuộc lớp c cho trước:

H(S)=  – ∑cc=1 (Nc/N) log(Nc/N)

•**Bước 2**: Tính hàm số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node S1, S2, …, SK với số điểm trong mỗi child node lần lượt là m1, m2 ,…, mK , ta có:

H(x, S) = ∑Kk=1 (mk / N) \* H(Sk )

**Bước 3**: Chỉ số Gain Information được tính bằng:

**G(x, S) = H(S) – H(x,S)**

Với ví dụ 2 trên, ta tính được hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)Parent= -(0.57\*ln(0.57) + 0.43\*ln(0.43)) = 0.68

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) theo phương pháp chia thứ nhất:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)left= -(.75\*ln(0.75) + 0.25\*ln(0.25))  = 0.56[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)right= -(.33\*ln(0.33) + 0.67\*ln(0.67)) = 0.63

Ta có thể tính hệ số **Information Gain** như sau:

Information Gain = 0.68 – (4\*0.56 + 3\*0.63)/7 = 0.09

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) với phương pháp chia thứ hai như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)left= -(.67\*ln(0.67) + 0.33\*ln(0.33))  = 0.63[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)middle= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)right= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69

Code :

from sklearn.tree import DecisionTreeClassifier  
  
model = DecisionTreeClassifier()  
model.fit(X\_train, Y\_train)  
print("training accuracy :", model.score(X\_train, Y\_train))  
print("testing accuracy :", model.score(X\_test, Y\_test))

Kết quả : 0.96

2.3.3 Đánh giá

-Accuracy score

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.

Các thông số

### Precision

Xem xét trên tập dữ liệu kiểm tra xem có bao nhiêu dữ liệu được mô hình dự đoán đúng. Đây chính là chỉ số **accuracy - độ chính xác** như chúng ta sử dụng bên trên. Một cách toán học thì **precision** được biểu diển như sau:

Precision= Y(true)/Y(tổng)

### Recall

Chỉ số này còn được gọi là **độ bao phủ** tức là xem xét xem mô hình tìm được có khả năng **tổng quát hóa** như thế nào. Từ hai yếu tố **độ chính xác** và **độ bao phủ** phía trên người ta đưa ra một chỉ số khác gọi là **F1-Score**

**Recall= [Y(true) giao Y(tổng)]/Y(true)**

### F1-Score

Đây được gọi là một **trung bình điều hòa**(harmonic mean) của các tiêu chí Precision và Recall. Nó có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị **Precision** và **Recall** và đồng thời nó có giá trị lớn nếu cả 2 giá trị **Precision** và **Recall** đều lớn. Chính vì thế **F1-Score** thể hiện được một cách khách quan hơn **performance** của một mô hình học máy.

F1- Score = 2x [(Precision.Recall)/(Precision+Recall)]

**Chương 3 Đánh giá**

* Với bài toán ta có thể phân chia các loại động vật một cách khá là chính xác, với bài toán tá thể chia các loại động vật thành các lớp động vật với các đặc tính gần giống nhau.
* Ta có thể kết hợp các kỹ thuật học máy lại với nhau để đưa ra kết quả tốt nhất có thể