TRƯỜNG ĐẠI HỌC VĂN LANG

Đạo đức - Ý chí - Sáng tạo

KHOA CÔNG NGHỆ THÔNG TIN

THỰC HÀNH NHẬP MÔN HỌC MÁY

THỰC HÀNH – DICISION TREE

BỘ MÔN KHOA HỌC DỮ LIỆU

THỰC HÀNH – DICISION TREE

***Decision Tree**

-Cây quyết định -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 tượng n thuộc tính 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.

-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.

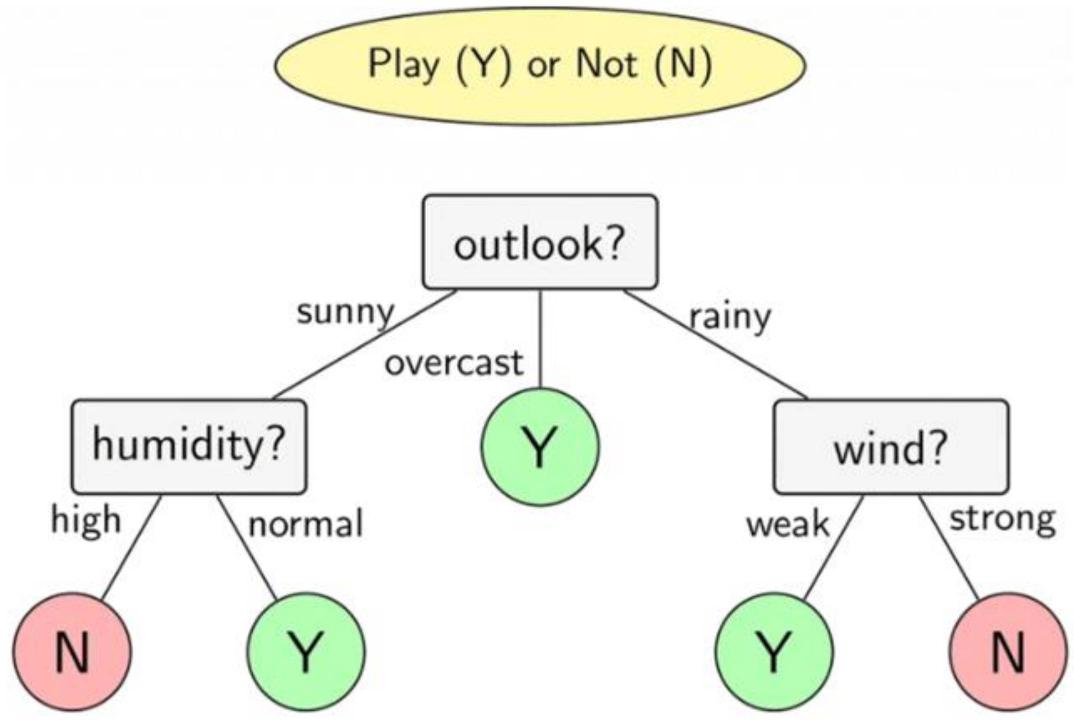
-Xét một ví dụ 1 kinh điển khác về cây quyết định. Giả sử dựa theo thời tiết mà các bạn nam sẽ quyết định đi đá bóng hay không?

Những đặc điểm ban đầu là:

Thời tiết
Độ ẩm
Gió

THỰC HÀNH – DICISION TREE

Dựa vào những thông tin trên, bạn có thể xây dựng được mô hình như sau



Mô hình cây quyết định

THỰC HÀNH – DICISION TREE

-Entropy trong Cây quyết định -Decision Tree

-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 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 như: $x_1, x_2, \ldots, x_n$.

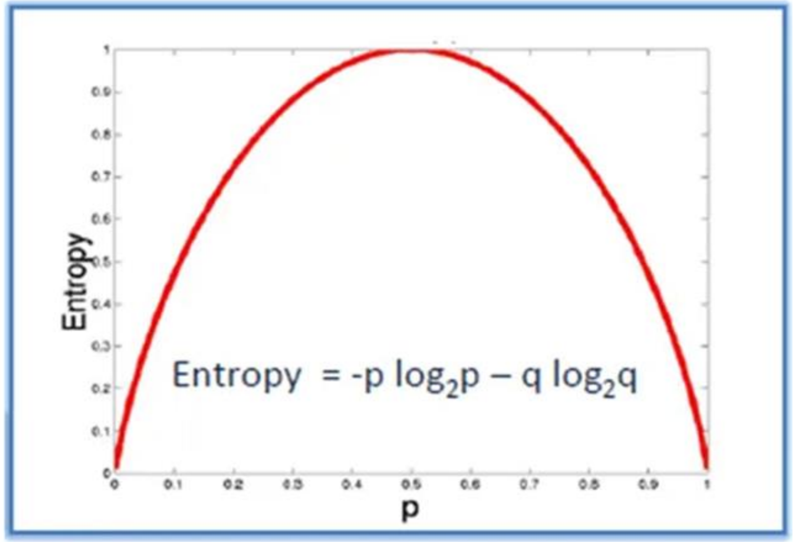
+Giả sử rằng xác suất để X nhận các giá trị tương ứng là $p_i = p(X = x_i)$.

+Ký hiệu $p = (p_1, p_2, \ldots, p_n)$. Entropy của phân phối này được định nghĩa là:

$$H(p) = -\sum_{n=1}^{n} p_i \log(p_i)$$

+Giả sử, tung một đồng xu, entropy sẽ được tính như sau:

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



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

THỰC HÀNH – DICISION TREE

-Biểu diễn sự thay đổi của hàm entropy.
-Hàm entropy đạt tối đa khi xác suất xảy ra của hai lớp bằng nhau.
P tinh khiết: $p_i = 0$ hoặc $p_i = 1$
P vẩn đục: $p_i = 0.5$, khi đó hàm Entropy đạt đỉnh cao nhất

***Information Gain trong Cây quyết định -Decision Tree**

-Information Gain dựa trên sự giảm của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính.
-Xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Infomation gain cao nhất.
-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 của biến mục tiêu S có N phần tử với N_c phần tử thuộc lớp c cho trước:

$$H(S) = -\sum_{c=1}^{c} (N_c/N) \log(N_c/N)$$

Bước 2: Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node $S_1, S_2, \ldots, S_K$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, \ldots, m_K$, ta có:

$$H(x, S) = \sum_{k=1}^{K} (m_k / N) * H(S_k)$$

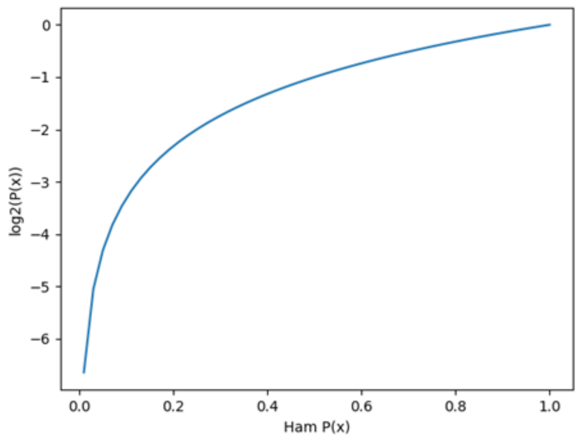
Bước 3: Chỉ số Gain Information được tính bằng:

$$G(x, S) = H(S) - H(x,S)$$

THỰC HÀNH – DICISION TREE

#Thực hành 01

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> fig=plt.figure()
>>> ax=fig.add_subplot(111)
>>> ax.plot(np.linspace(0.01,1),np.log2(np.linspace(0.01,1)))
[<matplotlib.lines.Line2D object at 0x09590230>]
>>> ax.set_xlabel("Ham P(x)")
Text(0.5, 0, 'Ham P(x)')
>>> ax.set_ylabel("log2(P(x))")
      # tuong tu bieu dien ham log(P(x)), log(1-P(x));
Text(0, 0.5, 'log2(P(x))')
>>> plt.show()
```



THỰC HÀNH – DICISION TREE

#Thực hành 02

```
>>> import numpy as np
>>> def gini_index(groups, classes):
     n_instances = float(sum([len(group) for group in groups]))
     gini = 0.0
     for group in groups:
          size = float(len(group))
          if size == 0:
               continue
          score = 0.0
          for class_val in classes:
               p = [row[-1] for row in group].count(class_val) / size
               score += p * p
          gini += (1.0 - score) * (size / n_instances)
     return gini

>>> print(gini_index([[[1, 1], [1, 0]], [[1, 1], [1, 0]]], [0, 1]))
>>> print(gini_index([[[1, 0], [1, 0]], [[1, 1], [1, 1]]], [0, 1]))
```

THỰC HÀNH – DICISION TREE

#Thực hành 03

```
>>> def test_split(index, value, dataset):
	left, right = list(), list()
	for row in dataset:
		if row[index] < value:
			left.append(row)
		else:
			right.append(row)
	return left, right
>>> def gini_index(groups, classes):
	n_instances = float(sum([len(group) for group in groups]))
	gini = 0.0
	for group in groups:
		size = float(len(group))
		if size == 0:
			continue
		score = 0.0
		for class_val in classes:
			p = [row[-1] for row in group].count(class_val) / size
			score += p * p
		gini += (1.0 - score) * (size / n_instances)
	return gini
```

THỰC HÀNH – DICISION TREE

```
>>> def get_split(dataset):
	class_values = list(set(row[-1] for row in dataset))
	b_index, b_value, b_score, b_groups = 999, 999, 999, None
	for index in range(len(dataset[0])-1):
		for row in dataset:
			groups = test_split(index, row[index], dataset)
			gini = gini_index(groups, class_values)
			print('X%d < %.3f Gini=%.3f' % ((index+1), row[index], gini))
			if gini < b_score:
				b_index, b_value, b_score, b_groups = index, row[index], gini, groups
	return {'index':b_index, 'value':b_value, 'groups':b_groups}
```

```
>>> dataset = [[2.771244718,1.784783929,0],
	[1.728571309,1.169761413,0],
	[3.678319846,2.81281357,0],
	[3.961043357,2.61995032,0],
	[2.999208922,2.209014212,0],
	[7.497545867,3.162953546,1],
	[9.00220326,3.339047188,1],
	[7.444542326,0.476683375,1],
	[10.12493903,3.234550982,1],
	[6.642287351,3.319983761,1]]
>>> split = get_split(dataset)
```

THỰC HÀNH – DICISION TREE

#Thực hành 04

```
>>> def predict(node, row):
    if row[node['index']] < node['value']:
        if isinstance(node['left'], dict):
            return predict(node['left'], row)
        else:
            return node['left']
    else:
        if isinstance(node['right'], dict):
            return predict(node['right'], row)
        else:
            return node['right']
```

```
>>> dataset = [[2.771244718,1.784783929,0],
    [1.728571309,1.169761413,0],
    [3.678319846,2.81281357,0],
    [3.961043357,2.61995032,0],
    [2.999208922,2.209014212,0],
    [7.497545867,3.162953546,1],
    [9.00220326,3.339047188,1],
    [7.444542326,0.476683375,1],
    [10.12493903,3.234550982,1],
    [6.642287351,3.319983761,1]]
>>> stump = {'index': 0, 'right': 1, 'value': 6.642287351, 'left': 0}
>>> for row in dataset:
    prediction = predict(stump, row)
    print('Expected=%d, Got=%d' % (row[-1], prediction))
```

TRƯỜNG ĐẠI HỌC
VĂN LANG
Đạo đức - Ý chí - Sáng tạo

KHOA CÔNG NGHỆ THÔNG TIN