Open in Colab

CHỦ ĐỀ CHUẨN BỊ DỮ LIỆU-III

Nội dung

Tiền xử lý dữ liệu

1. □ Giới thiệu
2. □ Nhị phân hóa
3. □ Loại bỏ trung bình
4. □ Điều chỉnh phạm vi dữ liệu
5. □ Chuẩn hóa

1- Tiền xử lý dữ liệu

Giới thiệu

- Trước khi đi xa hơn trong học máy, chúng ta phải tập trung vào một nhiệm vụ quan trọng: **preprocessing data**.
- Trước khi bắt đầu quá trình học (và dự báo) Before starting the learning (and the predicting) cần thiết phải **dữ liệu phải được tiền xử lý** hoặc **biến đổi**.
- Sự khác nhau các quá trình có thể áp dụng vào dữ liệu đó là:
    - Nhị phân hóa
    - Loại bỏ trung bình
    - Điều chỉnh phạm vi
    - Chuẩn hóa

Nhị phân hóa

- Chuyển các giá trị của đặc trưng vào các giá trị **boolean**.
- Cố định một **ngưỡng**. Sau đó **chuyển đổi** các giá trị đặc trưng được tính toán theo **ngưỡng**.
- □ Các giá trị
  > ngưỡng sẽ được chuyển thành **1**
- □ Các giá trị
  <= ngưỡng sẽ được chuyển thành **0**

Ví dụ

```
data= [-0.626009,1.556353,0.45059,0.898856,-0.191403].
Sử dụng DataFrame để xem dữ liệu, sau đó tiền xử lý cho dữ liệu
```

Bước 1: Tải các thư viện

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from pandas import DataFrame as df, Series as S
# thư viện tiền xử lý
from sklearn import preprocessing
```

Bước 2: Chuẩn bị dữ liệu (2.1 tải dữ liệu, 2.2 làm sạch, 2.3 chuẩn hóa, 2.4 tìm hiểu dữ liệu)

Bước 2.1: Tải dữ liệu

```
# Vì dữ liệu chưa rõ các dòng và các đặc trưng nên cần tạo chỉ số dòng và đặc trưng
myData = df([-0.626009,1.556353,0.45059,0.898856,-0.191403],index = ["s1","s2","s3","s4","s5"],columns =
["Feature1"])
# Xem dữ liệu
myData
```

	Feature1
s1	-0.626009
s2	1.556353
s3	0.450590
s4	0.898856
s5	-0.191403

Bước 2.2: Làm sạch dữ liệu

- Loại bỏ dữ liệu sai và thiếu

Bước 2.3: Chuẩn hóa dữ liệu (Chọn 1 trong các cách sau):

Cách 1: Chuyển giá trị đặc trưng theo giá trị ngưỡng

Cách sử dụng:

```
th = 1
# khai báo biến đối tượng Binarizer với ngưỡng khởi tạo là th
myBin = preprocessing.Binarizer(threshold=th)
# chuyển các giá trị đặc trưng theo ngưỡng sử dụng phương thức transform của lớp  Binarizer
myD_B =myBin.transform(myData.Feature1.values.reshape(-1,1))
# Hiển thị các giá trị đặc trưng đã chuyển đổi theo ngưỡng
myD_B
```

```
array([[0.],
       [1.],
       [0.],
       [0.],
       [0.]])
```

Cách 2: loại trung bình

- áp dụng loại trung bình do đó các giá trị đặc trưng theo tâm **0**.

Cách sử dụng:

```
print("trung bình ban đầu==",myData.mean())

# khai báo biến đối tượng myData_mean của lớp scale trong module preprocessing
myData_mean = preprocessing.scale(myData,with_std=False)

# gọi phương thức mean() của đối tượng myData_mean
print("giá trị trung bình mới ==",myData_mean.mean())

# Hiển thị dữ liệu sau loại bỏ trung bình
myData_mean
```

```
trung bình ban đầu == Feature1    0.417677
dtype: float64
giá trị trung bình mới == 2.2204460492503132e-17

array([[-1.0436864],
       [ 1.1386756],
       [ 0.0329126],
       [ 0.4811786],
       [-0.6090804]])
```

Cách 3: Điều chỉnh phạm vi Min Max

- trong trường hợp dữ liệu được biểu diễn nhiều chiều đặc trưng. Các giá trị có thể rất khác nhau giữa đặc trưng này và đặc trưng khác.
- Ví dụ, xác suất các giá trị thay đổi giữa **0** và **1**, và các giá trị chiếm phần lớn vượt qua giá trị này.
- Vậy nên ta áp dụng phép biến đổi **sacling** để tránh một đặc trưng nào đó ảnh hưởng đến kết quả hơn so với một đặc trưng khác, do phạm vi các giá trị của nó.
- phương pháp thường sử dụng là **Min Max scaling**: sử dụng giá trị nhỏ nhất và lớn nhất cho mỗi đặc trưng để đưa các giá trị vào một phạm vi cụ thể áp dụng theo công thức:

$X_{std} = (X - X_{min}) / (X_{max} - X_{min})$

$X_{scaled} = X_{std} * (max - min) + min$

min, max : phạm vi đặc trưng

Cách sử dụng:

```
# thêm một cột mới vào dữ liệu Feature2 và Feature3 myData
myData["Feature2"]=S([100,1000,500,200,80],index = ["s1","s2","s3","s4","s5"])
myData["Feature3"]=S([9,7,-3,1,55],index = ["s1","s2","s3","s4","s5"])
# xem dữ liệu
myData
```

	Feature1	Feature2	Feature3
s1	-0.626009	100	9
s2	1.556353	1000	7
s3	0.450590	500	-3
s4	0.898856	200	1
s5	-0.191403	80	55

Cách sử dụng:

```
# khởi tạo biến đối tượng myScaler của lớp MinMaxScaler trong module preprocessing, tham số phạm vi
feature_range = (0,1)
myScaler=preprocessing.MinMaxScaler(feature_range=(0,1))

# điều chỉnh phạm vi các giá trị đặc trưng trong dữ liệu myData
myData_s = myScaler.fit_transform(myData)

# xem dữ liệu
myData_s


array([[0.        , 0.02173913, 0.20689655],
       [1.        , 1.        , 0.17241379],
       [0.49331825, 0.45652174, 0.        ],
       [0.6987223 , 0.13043478, 0.06896552],
       [0.19914478, 0.        , 1.        ]])
```

Cách 4: Chuẩn hóa

- Một cách khác để tránh thu hẹp phạm vi (giống như các giá trị ngoại lai) ta dùng cách **chuẩn hóa** dữ liệu

$||x||_p = (\sum_i |x_i|p)^{1/p}$

- chuẩn ** L1** lấy độ lệch trị tuyệt đối: chuyển các giá trị đặc trưng theo cách là tổng của các **trị tuyệt đối** trên mỗi dòng mẫu sẽ **bằng với 1**.
- Chuẩn ** L2** (**Least Squares**) : chuyển các giá trị đặc trưng theo cách tổng của căn bậc hai của mỗi dòng mẫu sẽ **bằng với 1**.

Cách sử dụng:

```
# Chuẩn hóa dữ liệu myData sử dụng chuẩn hóa L1
# Khai báo biến đối tượng myData_l1 thuộc lớp preprocessing với phương thức normalize() được truyền vào
tham số là tập dữ liệu và kiểu chuẩn hóa.
myData_l1 = preprocessing.normalize(myData, norm='l1')

# xem dữ liệu
myData_l1


array([[-5.71040582e-03,  9.12192288e-01,  8.20973059e-02],
       [ 1.54314927e-03,  9.91516237e-01,  6.94061366e-03],
       [ 8.95003420e-04,  9.93146120e-01, -5.95887672e-03],
       [ 4.45201136e-03,  9.90595014e-01,  4.95297507e-03],
       [-1.41579269e-03,  5.91753604e-01,  4.06830603e-01]])


# Chuẩn hóa dữ liệu myData sử dụng chuẩn hóa L2
# Khai báo biến đối tượng myData_l2 thuộc lớp preprocessing với phương thức normalize() được truyền vào
tham số là tập dữ liệu và kiểu chuẩn hóa norm='l2'.
myData_l2 = preprocessing.normalize(myData, norm='l2')

# xem dữ liệu
myData_l1


array([[-6.23476844e-03,  9.95955081e-01,  8.96359573e-02],
       [ 1.55631299e-03,  9.99974290e-01,  6.99982003e-03],
       [ 9.01163413e-04,  9.99981594e-01, -5.99988957e-03],
       [ 4.49417844e-03,  9.99977401e-01,  4.99988701e-03],
       [-1.97154737e-03,  8.24040323e-01,  5.66527722e-01]])
```

Thực hành: Thực hiện bước chuẩn bị dữ liệu trên tập dữ liệu: wine.data.csv

Tham khảo

- Joshi Prateek. Artificial intelligence with Python. Packt Publishing, 2017.
- Jake VanderPlas. Python data science handbook: essential tools for working with data. O'Reilly Media, Inc, 2017.

- Scikit-learn.org. scikit-learn, machine learning in python. On-line at scikit-learn.org/stable/. Accessed on 03-11-2018