

Data preprocessing

Nguyễn Sơn

sonnhfit@gmail.com

Nội Dung

- **Data preprocessing (R/PYTHON)**
- Xử lý dữ liệu bị thiếu (Missing value)
- Đánh danh mục dữ liệu (Encode categorical data)
- Feature scaling

Tiền xử lý dữ liệu

- Đọc vào tập dữ liệu
- Xử lý dữ liệu bị thiếu
- Mã hóa dữ liệu phân loại
- Tách dữ liệu thành các tập dữ liệu đào tạo và kiểm tra
- Chuẩn hóa dữ liệu(Feature scaling)

```
dataset <- read.csv("Data.csv")
dataset
```

```
##   Country Age Salary Purchased
## 1 France  44 72000      No
## 2 Spain   27 48000     Yes
## 3 Germany 30 54000      No
## 4 Spain   38 61000      No
## 5 Germany 40    NA     Yes
## 6 France  35 58000     Yes
## 7 Spain   NA 52000      No
## 8 France  48 79000     Yes
## 9 Germany 50 83000      No
## 10 France 37 67000    Yes
```

Data.csv

Đọc vào tập dữ liệu

Filename.csv cùng thư mục với file code

```
# Khai báo thư viện nè
import pandas as pd
# Đọc dữ liệu từ file 'filename.csv'
# (in the same directory that your python process is based)
# Control delimiters, rows, column names with read_csv (see later)
data = pd.read_csv("filename.csv")
# Preview the first 5 lines of the loaded data
data.head()
```

Xử lý dữ liệu bị thiếu

1. Thay NA thành giá trị trung bình.
2. Xoá
3. Đambiền giá trị cótần suất xuất hiện nhiều nhất
- ...

```
dataset <- read.csv("Data.csv")
dataset
```

```
##   Country Age Salary Purchased
## 1 France  44 72000     No
## 2 Spain   27 48000    Yes
## 3 Germany 30 54000     No
## 4 Spain   38 61000     No
## 5 Germany 40 NA          NA
## 6 France  35 58000    Yes
## 7 Spain   NA 52000     No
## 8 France  48 79000    Yes
## 9 Germany 50 83000     No
## 10 France 37 67000   Yes
```

#Delete rows with missing values

```
df <- na.omit(dataset)
```

Xử lý dữ liệu bị thiếu: Trung bình(R)

Điền giá trị trung bình vào những ô bị thiếu

```
dataset$Age <- ifelse(is.na(dataset$Age),  
                      ave(dataset$Age, FUN = function(x)  
                           mean(x, na.rm = TRUE)), dataset$Age)
```

```
dataset$Salary <- ifelse(is.na(dataset$Salary),  
                         ave(dataset$Salary, FUN = function(x)  
                            mean(x, na.rm = TRUE)), dataset$Salary)
```

```
##   Country Age Salary Purchased  
## 1  France  44  72000      No  
## 2   Spain  27  48000     Yes  
## 3 Germany  30  54000      No  
## 4   Spain  38  61000      No  
## 5 Germany  40      NA     Yes  
## 6  France  35  58000     Yes  
## 7   Spain  NA  52000      No  
## 8  France  48  79000     Yes  
## 9 Germany  50  83000      No  
## 10 France  37  67000     Yes
```



```
##   Country     Age    Salary Purchased  
## 1  France 44.00000 72000.00      No  
## 2   Spain 27.00000 48000.00     Yes  
## 3 Germany 30.00000 54000.00      No  
## 4   Spain 38.00000 61000.00      No  
## 5 Germany 40.00000 63777.78     Yes  
## 6  France 35.00000 58000.00     Yes  
## 7   Spain 38.77778 52000.00      No  
## 8  France 48.00000 79000.00     Yes  
## 9 Germany 50.00000 83000.00      No  
## 10 France 37.00000 67000.00     Yes
```

Xử lý dữ liệu bị thiếu: Trung bình(Python pandas)

```
import pandas as pd  
import numpy as np  
dff = pd.DataFrame(np.random.randn(10, 3), columns=list('ABC'))
```

In [53]: dff

Out[53]:

	A	B	C
0	0.271860	-0.424972	0.567020
1	0.276232	-1.087401	-0.673690
2	0.113648	-1.478427	0.524988
3	NaN	0.577046	-1.715002
4	NaN	NaN	-1.157892
5	-1.344312	NaN	NaN
6	-0.109050	1.643563	NaN
7	0.357021	-0.674600	NaN
8	-0.968914	-1.294524	0.413738
9	0.276662	-0.472035	-0.013960



In [54]: dff.fillna(dff.mean())

Out[54]:

	A	B	C
0	0.271860	-0.424972	0.567020
1	0.276232	-1.087401	-0.673690
2	0.113648	-1.478427	0.524988
3	-0.140857	0.577046	-1.715002
4	-0.140857	-0.401419	-1.157892
5	-1.344312	-0.401419	-0.293543
6	-0.109050	1.643563	-0.293543
7	0.357021	-0.674600	-0.293543
8	-0.968914	-1.294524	0.413738
9	0.276662	-0.472035	-0.013960

Xử lý dữ liệu bị thiếu: Trung bình

x = NA 2, 3

```
#defining x = 1 2 3
x <- 1:3
#introducing missing value
x[1] <- NA
# mean = NA
mean(x)
```

```
## [1] NA
```

```
# mean = mean excluding the NA value
mean(x, na.rm = T)
```

```
## [1] 2.5
```

x = 2.5, 2, 3

ENCODE CATEGORICAL DATA (R)

```
# Encoding categorical data
dataset$Country = factor(dataset$Country,
                         levels = c('France', 'Spain', 'Germany'),
                         labels = c(1, 2, 3))

dataset$Purchased = factor(dataset$Purchased,
                           levels = c('No', 'Yes'), labels = c(0, 1))
```

```
##   Country     Age   Salary Purchased
## 1   France 44.00000 72000.00      No
## 2    Spain 27.00000 48000.00     Yes
## 3 Germany 30.00000 54000.00      No
## 4    Spain 38.00000 61000.00      No
## 5 Germany 40.00000 63777.78     Yes
## 6   France 35.00000 58000.00     Yes
## 7    Spain 38.77778 52000.00      No
## 8   France 48.00000 79000.00     Yes
## 9 Germany 50.00000 83000.00      No
## 10  France 37.00000 67000.00     Yes
```



	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	0
2	2	27.00000	48000.00	1
3	3	30.00000	54000.00	0
4	2	38.00000	61000.00	0
5	3	40.00000	63777.78	1
6	1	35.00000	58000.00	1
7	2	38.77778	52000.00	0
8	1	48.00000	79000.00	1
9	3	50.00000	83000.00	0
10	1	37.00000	67000.00	1

ENCODE CATEGORICAL DATA (python)

```
In [67]: s = pd.Series(["a", "b", "c", "a"], dtype="category")

In [68]: s
Out[68]:
0    a
1    b
2    c
3    a
dtype: category
Categories (3, object): ['a', 'b', 'c']

In [69]: s.cat.categories = ["Group %s" % g for g in s.cat.categories]

In [70]: s
Out[70]:
0    Group a
1    Group b
2    Group c
3    Group a
dtype: category
Categories (3, object): ['Group a', 'Group b', 'Group c']

In [71]: s = s.cat.rename_categories([1, 2, 3])

In [72]: s
Out[72]:
0    1
1    2
2    3
3    1
dtype: category
Categories (3, int64): [1, 2, 3]
```

FEATURE SCALING (NORMALIZATION)

- Cần feature scaling khi các đặc trưng khác nhau có phạm vi khác nhau, ví dụ: **Tuổi** và **lương** phạm vi chênh lệch nhau lớn.
- Chúng có các khoảng dữ liệu rất khác nhau nhưng khi đào tạo một mô hình, về cơ bản đang cố gắng tìm ra đường thẳng khớp với dữ liệu (trong hồi quy tuyến tính) và cố gắng giảm thiểu lỗi
- Để giảm thiểu lỗi, khoảng cách Euclid được giảm thiểu bằng cách sử dụng một số thuật toán (gradient descent)
- Nếu không áp dụng feature scaling thì việc huấn luyện sẽ có độ chệch cao với đối tượng có giá trị lớn vì khoảng cách Euclid sẽ lớn ở đó.

```
#feature scaling  
dataset[,2:3] = scale(dataset[,2:3])
```

	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	0
2	2	27.00000	48000.00	1
3	3	30.00000	54000.00	0
4	2	38.00000	61000.00	0
5	3	40.00000	63777.78	1
6	1	35.00000	58000.00	1
7	2	38.77778	52000.00	0
8	1	48.00000	79000.00	1
9	3	50.00000	83000.00	0
10	1	37.00000	67000.00	1



	Country	Age	Salary	Purchased
1	1	0.7199314	0.7110128	0
2	2	-1.6236751	-1.3643758	1
3	3	-1.2100975	-0.8455287	0
4	2	-0.1072238	-0.2402070	0
5	3	0.1684946	0.0000000	1
6	1	-0.5208015	-0.4996306	1
7	2	0.0000000	-1.0184777	0
8	1	1.2713683	1.3163344	1
9	3	1.5470867	1.6622325	0
10	1	-0.2450830	0.2786401	1

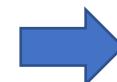
FEATURE SCALING (NORMALIZATION) python

```
import pandas as pd
import numpy as np
from sklearn import preprocessing

# Create a DataFrame
d = {'Score':[62,-47,-55,74,31,77,85,63,42,67,89,81,56]}

df = pd.DataFrame(d,columns=['Score'])
float_array = df['Score'].values.astype(float)
min_max_scaler = preprocessing.MinMaxScaler()
scaled_array = min_max_scaler.fit_transform(float_array)
df_normalized = pd.DataFrame(scaled_array)
print(df_normalized) #ket qua
```

	Score
0	62
1	-47
2	-55
3	74
4	31
5	77
6	85
7	63
8	42
9	67
10	89
11	81
12	56



0
0 0.812500
1 0.055556
2 0.000000
3 0.895833
4 0.597222
5 0.916667
6 0.972222
7 0.819444
8 0.673611
9 0.847222
10 1.000000
11 0.944444
12 0.770833

Chia tập dữ liệu thành tập đào tạo và tập test(R)

```
#install.packages("caTools") #if not present
library(caTools) #adding caTools to the library
set.seed(123) # this is to ensure same output as split is done
randomly, you can exclude in real time
split = sample.split(dataset$Purchased, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

- SplitRatio là tỷ lệ của tập dữ liệu test và tập training, nó thường đặt 80:20. 80% cho training và 20% cho test.
- Phương thức Subset () nhận tập dữ liệu và tập con theo điều kiện

Chia tập dữ liệu train và test

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape(5, 2), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
```

```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

```
>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```