Tiền xử lý dữ liệu trong lĩnh vực học máy (Phần 1)

GIỚI THIỆU

Tiền xử lý dữ liệu là một bước rất quan trọng trong việc giải quyết bất kỳ vấn đề nào trong lĩnh vực Học Máy. Hầu hết các bộ dữ liệu được sử dụng trong các vấn đề liên quan đến Học Máy cần được xử lý, làm sạch và biến đổi trước khi một thuật toán Học Máy có thể được huấn luyện trên những bộ dữ liệu này. Các kỹ thuật tiền xử lý dữ liệu phổ biến hiện nay bao gồm: xử lý dữ liệu bị khuyết (missing data), mã hóa các biến nhóm (encoding categorical variables), chuẩn hóa dữ liệu (standardizing data), co giãn dữ liệu (scaling data),… Những kỹ thuật này tương đối dễ hiểu nhưng sẽ có nhiều vấn đề phát sinh khi chúng ta áp dụng vào các dữ liệu thực tế. Bởi lẽ các bộ dữ liệu ứng với các bài toán trong thực tế rất khác nhau và mỗi bài toàn thì đối mặt với những thách thức khác nhau về mặt dữ liệu. Trong bài viết này, chúng ta sẽ cùng nhau tìm hiểu về các kỹ thuật tiền xử lý dữ liệu và cách áp dụng chúng trong các bài toán thực tế.

DỮ LIỆU

Trước khi đi vào vấn đề xử lý dữ liệu, điều đầu tiên chúng ta cần biết đó chính là dữ liệu. Trong mục này, chúng ta sẽ cùng nhau tìm hiểu về dữ liệu trong các bài toán Học Máy, các kiểu dữ liệu và những khái niệm cơ bản liên quan đến dữ liệu.

Đặc trưng của dữ liệu

Chắc hẳn chúng ta đều quen với khái niệm bộ dữ liệu, vậy một bộ dữ liệu gồm những thành phần gì? Về cơ bản, một bộ dữ liệu sẽ được tổ chức theo hai thành phần là các mẫu dữ liệu (samples) và các đặc trưng của dữ liệu (features). Các bạn có thể xem hình 2.1 để dễ hình dung hơn.

	Features				
	Fullname	*Age*	*Heigh (cm)*	*Weight (kg)*	*Job*
	Thu Trang	21	Low	47	Designer
Samples	Phan Cuong	23	Medium	54	Developer
	Kieu Linh	25	Medium	58	Marketing

Hình 2.1. Ví dụ về một bộ dữ liệu

- Một mẫu dữ liệu (hoặc một điểm dữ liệu) là một thành phần giúp bạn mô tả dữ liệu theo các đặc điểm định lượng. Chúng được thể hiện bằng các hàng trong bộ dữ liệu của bạn. Nếu bạn đang xây dựng một bộ dữ liệu về hình dạng con người, mỗi mẫu dữ liệu sẽ là các số liệu chi tiết về một người. Khi chúng ta nói rằng chúng ta cần nhiều dữ liệu để xây dựng mô hình, điều đó có nghĩa là chúng ta cần nhiều mẫu dữ liệu.

- Các đặc trưng là những đặc điểm định lượng mô tả các mẫu dữ liệu của bạn. Chúng có thể là số hoặc chữ, ví dụ *Fullname* là một đặc trưng dạng chữ, trong khi *Height* là một đặc trưng dạng số.

Các kiểu đặc trưng

Có rất nhiều các cách gọi tên khác cho đặc trưng, nó phụ phuộc vào người nói là ai và ngữ cảnh là gì. Một số tên gọi khác của đặc trưng là:

- *Thuộc tính (attribute)*: Các đặc trưng là các thuộc tính định lượng của mẫu dữ liệu quan sát được.

- *Trục (axis)*: Các đặc trưng là các trục vuông góc trong không gian đặc trưng nếu chúng độc lập tuyến tính.

- *Cột (column)*: Các đặc trưng được biểu diễn thành các cột trong tập dữ liệu của bạn.

- *Chiều (dimension)*: Tất cả các đặc trưng của một bộ dữ liệu được nhóm lại với nhau có thể được gọi là không gian toạn độ *n* chiều.

- *Đầu vào (input)*: Các giá trị của đặc trưng là các đầu vào của các tiến trình biến đổi dữ liệu hoặc các thuật toán học máy.

- *Biến độc lập (independent variable)*: Các đặc trưng còn được gọi là các biến độc lập trong đại số tuyến tính.

Mặc dù có rất nhiều tên gọi khác nhau, nhưng bất kỳ một đặc trưng nào cũng sẽ thuộc một trong hai loại sau:

- **Đặc trưng liên tục**: Người ta định nghĩa một số phép đo khác nhau giữa các giá trị của đặc trưng liên tục. Các giá trị đặc trưng liên tục thường là một tập con của tập số thực, có thể kể đến một số phép đo các giá trị liên tục như: khoảng cách, thời gian, giá cả, nhiệt độ,…
- **Đặc trưng dạng nhóm**: Với các đặc trưng dạng nhóm, hay còn gọi là đặc trưng rời rạc, các đặc trưng này có tập giá trị cụ thể và giới hạn. Những giá trị này có thể có hoặc không có thứ tự. Nếu chúng có thứ tự, ta gọi chúng là các *đặc trưng nhóm có thứ tự* (oridinal categorical features). Ví dụ cho loại đặc trưng này là đặc trưng *Height* bao gồm ba giá trị là *High-Medium-Low*. Ngược lại, nếu chúng không có tính thứ tự, ta gọi các đặc trưng này là các *đặc trưng định danh* (nominal categorical features). Đặc trưng *Job* trong bộ dữ liệu trên là một ví dụ cho loại đặc trưng này.

Cần chú ý rằng, dữ liệu liên tục hầu hết được biểu diễn dưới dạng đặc trưng số. Nhưng một dữ liệu là dạng số không có nghĩa đó là đặc trưng liên tục. Trong thực tế có những dữ liệu thuộc loại dữ liệu nhóm số (Ví dụ như giá trị của đặc trưng cấp học của một học sinh trong độ tuổi từ 6 tới 18 tuổi là 1, 2 hoặc 3).

TIỀN XỬ LÝ DỮ LIỆU

Tinh chỉnh dữ liệu

Trong thực tế, dữ liệu có thể không được thu thập trực tiếp bởi con người vì các lý do xoay quanh vấn đề về chi phí, cơ sở hạ tầng, con người. Do đó, dữ liệu có thể bị thiếu bởi một sai sót của máy móc, hoặc thực tế nó không tồn tại tại một thời điểm nhất định trong khi thu thập dữ liệu. Qua nghiên cứu, người ta xác định được rằng các dữ liệu bị khuyết bởi các nguyên nhân sau đây:

- *Khuyết ngẫn nhiên (Missing at Random – MAR)*: Khuyết ngẫn nhiên nghĩa là xu hướng giá trị của một đặc trưng bị khuyết không liên quan đến tính chất của đặc trưng đó nhưng liên quan đến một vài đặc trưng không bị khuyết khác. Nói cách khác, sự khuyết dữ liệu của đặc trưng này có điều kiện hoặc phụ thuộc vào một hoặc một vài các đặc trưng khác.
- *Khuyết hoàn toàn ngẫu nhiên (Missing Completely at Random – MCAR)*: Khuyết hoàn toàn ngẫu nhiên nghĩa là xu hướng bị khuyết của một đặc trưng là hoàn toàn ngẫu nhiên. Không có mối quan hệ nào giữa đặc trưng bị khuyết với các giá trị giả định hoặc các ràng buộc trên các đặc trưng khác. Ở đây, tập dữ liệu bị khuyết chỉ là một tập con ngẫu nhiên của bộ dữ liệu.
- *Khuyết không ngẫu nhiên (Missing not at Random – MNAR)*: Khuyết không ngẫu nhiên xảy ra khi một điểm dữ liệu bị khuyết phụ thuộc cả vào các giá trị giả định (ví dụ như những người giàu có thường không tiết lộ mức thu nhập của họ khi bạn khảo sát) và các giá trị của các đặc trưng khác (ví dụ như khi bạn muốn khảo sát tuổi của một người, mà người đó là con gái thì thường là bạn sẽ không nhận được câu trả lời từ họ).

Trong hai trường hợp đầu tiên, xóa đi các điểm dữ liệu bị thiếu dựa vào số lần xuất hiện của chúng là chấp nhận được. Nhưng trong trường hợp thứ ba, việc xóa đi các quan sát bị khuyết giá trị có thể khiến cho mô hình bị ảnh hưởng. Do đó, chúng ta cần rất lưu tâm trước khi xóa đi những điểm dữ liệu này.

Ngoài ra, chúng ta cũng cần hiểu rằng cách bỏ qua (không xử lý gì) những điểm dữ liệu bị khuyết này là cách không nên áp dụng, nó có thể rất tai hại nếu như bạn không xử lý đúng đắn khi phân tích dữ liệu của bạn. Bởi lẽ dữ liệu bị khuyết sẽ khiến bạn đưa ra những kết luận sai lầm về bộ dữ liệu của bạn khi bạn nhìn vào các giá trị sai về tổng, trung bình hoặc phân phối của bộ dữ liệu.

Bộ dữ liệu Pima Indians Diabetes

Bắt đầu từ phần này, chúng ta sẽ cùng nhau đi qua các cơ sở lý thuyết kết hợp với việc xử lý dữ liệu cụ thể trên một bộ dữ liệu mẫu. Tôi chọn bộ dữ liệu **Pima Indians Diabetes** vì nó chứa các giá trị bị khuyết trên một vài thuộc tính của dữ liệu. Bộ dữ liệu Pima Indians Diabetes là bộ dữ liệu thu thập các số liệu về các chỉ số y khoa của những người mắc và không mắc bệnh tiểu đường trong vòng 5 năm tại Pima Indian.

Đây chính xác là một vấn đề phân lớp nhị phân. Số lượng dữ liệu là 768 mẫu với 8 đặc trưng về các chỉ số y khoa và 1 thuộc tính nhãn lớp. Số lượng các quan sát cho các lớp là không đồng đều. Chi tiết về các đặc trưng theo thứ tự từ trái qua phải trong bộ dữ liệu như sau:

- 0: Số lần mang thai
- 1: Nồng độ glucose huyết tương là 2 giờ trong một kiểm tra dung nạp glocose bằng đường uống.
- 2: Huyết áp tâm trương (mm Hg)

- 3: Độ dày nếp gấp da cơ tam đầu (mm)
- 4: 2-giờ insulin huyết thanh (mu U/ml)
- 5: Chỉ số khối lượng cơ thể (Cân nặng theo kg/bình phương chiều cao theo m)
- 6: Chức năng phả hệ tiểu đường
- 7: Tuổi
- 8: Biến phân lớp (có giá trị bằng 1 nếu người đó mắc bệnh tiểu đường và 0 trong trường hợp ngược lại)

Chúng ta cùng nhau quan sát một số mẫu dữ liệu trích ra từ bộ dữ liệu:

```
import pandas as pd

# Set display options
pd.set_option('display.max_columns', 10)

# read data from csv file
dataset = pd.read_csv('pima-indians-diabetes.data.csv', header=0)

# print first 5 rows of data set
print(dataset.head())
```

Chúng ta quan sát thêm mô tả sơ lược của dữ liệu qua các thông số thống kê:

```
# describe dataset with statistic parameters

print(dataset.describe())
```

Theo kết quả quan sát được, bộ dữ liệu có 6 đặc trưng đầu tiên có giá trị nhỏ nhất là 0, điều này đồng nghĩa với việc 6 đặc trưng này có thể đã bị khuyết dữ liệu ở một số mẫu dữ liệu. Tuy nhiên, đặc trưng NoPregnant là đặc trưng về số lần mang thai, một người có thể đã mang thai hoặc chưa từng mang thai. Do đó giá trị 0 của đặc trưng này biểu thị cho những người chưa từng mang thai chứ không phải là bị khuyết dữ liệu. Các đặc trưng còn lại chứa giá trị 0 đang bị khuyết dữ liệu. Chúng ta thử đếm xem mỗi đặc trưng có bao nhiêu điểm dữ liệu bị khuyết:

```
# count missing values for each columns containing missing data

print((dataset[['PlasmaGlucose', 'DiastolicBlood', 'TSThickness', 'SerumInsulin', 'BodyMass']] == 0).sum())
```

Kết quả:

```
PlasmaGlucose       5

DiastolicBlood     35

TSThickness       227

SerumInsulin      374

BodyMass           11
```

Kết quả cho thấy 2 đặc trưng TSThickness và SerumInsulin có nhiều giá trị bị khuyết nhất. Bây giờ chúng ta đã có dữ liệu phù hợp để sẵn sàng tìm hiểu các phần tiếp theo.

Xóa đi dữ liệu bị khuyết

Khi đối mặt với trường hợp trong bộ dữ liệu thu thập được có các dữ liệu bị khuyết, cách đơn giản nhất mà chúng ta nghĩ tới là xóa chúng đi. Điều này có một lợi ích thấy rõ là giúp cho bộ dữ liệu được giảm đi những điểm dữ liệu mập mờ và những điểm dữ liệu nhiễu. Tuy nhiên, cách này có đem lại khuyết điểm gì không? Chúng ta cùng nhau phân tích các chiến lược xóa dữ liệu bị khuyết dưới đây để làm rõ hơn điều này.

Listwise

Phương pháp xóa dữ liệu listwise xóa đi tất cả các điểm dữ liệu có một hoặc một vài giá trị đặc trưng bị khuyết. Phương pháp này thường chỉ được sử dụng khi bạn đang tiến hành một nghiên cứu để so sánh

với một phương pháp xử lý khác. Vì trong thực tế thì phương pháp này đem lại nhiều bất lợi. Bởi lẽ dữ liệu khuyết hoàn toàn ngẫu nhiên MCAR thường hiếm gặp, do đó, phương pháp này nhiều khả năng tạo ra các tham số và ước lượng bị lệch cho mô hình.

Dưới đây là đoạn code xóa dữ liệu theo hướng listwise

```
# create another dataset for listwise deletion

lw = dataset

# delete rows containing any missing value

lw = lw.drop(lw[(lw['PlasmaGlucose'] > 0) | (lw['DiastolicBlood'] > 0) | (lw['TSThickness'] > 0)

                | (lw['SerumInsulin'] > 0) | (lw['BodyMass'] > 0)].index)

# count missing values again
print((lw[['PlasmaGlucose', 'DiastolicBlood', 'TSThickness', 'SerumInsulin', 'BodyMass']] == 0).sum())
```

Kết quả:

```
PlasmaGlucose     0

DiastolicBlood    0

TSThickness       0

SerumInsulin      0

BodyMass          0
```

Kết quả cho thấy không còn bản ghi nào bị thiếu dữ liệu trong bộ dữ liệu.

Pairwise

Phương pháp xóa pairwise cố gắng tối thiểu hóa sự mất mát khi sử dụng phương pháp xóa listwise. Để hiểu đơn giản về phương pháp xóa pairwise, ta hãy liên tưởng đến ma trận tương quan. Mỗi giá trị tương quan thể hiện mức độ liên kết giữa hai biến (đặc trưng). Với mỗi cặp biến mà dữ liệu không bị khuyết, hệ số tương quan sẽ được đưa vào tính toán. Do đó, phương pháp xóa pairwise tối đa hóa các dữ liệu có sẵn bởi một phân tích cơ sở. Một thế mạnh của phương pháp này là nó làm tăng sức mạnh và tính hiệu quả của các phân tích mà bạn muốn thực hiện. Mặc dù phương pháp này được khuyến nghị dùng nhiều hơn là phương pháp xóa listwise, nhưng nó cũng sử dụng giả thiết rằng dữ liệu bị khuyết thuộc dạng khuyết hoàn toàn ngẫu nhiên MCAR.

```
# convert missing data to nan value

print(dataset.info())

dataset['PlasmaGlucose'].replace(0, np.nan, inplace=True)

dataset['DiastolicBlood'].replace(0, np.nan, inplace=True)

dataset['TSThickness'].replace(0, np.nan, inplace=True)

dataset['SerumInsulin'].replace(0, np.nan, inplace=True)

dataset['BodyMass'].replace(0, np.nan, inplace=True)


# create another dataset for pairwise deletion

pw = dataset

# pairwise deletion

pw.dropna()

# Covarian matrix
```

```
print(pw.cov())
```

Xóa bỏ các đặc trưng

Sẽ luôn là tốt hơn nếu chúng ta giữ lại dữ liệu thay vì xóa bỏ nó khỏi dữ liệu thu thập được. Nhưng đôi khi bạn cũng có thể xóa bỏ đi một vài đặc trưng trong bộ dữ liệu nếu chúng bị khuyết nhiều hơn 60% (hoặc một ngưỡng nào đó mà bạn thấy phù hợp) tổng số quan sát và đặc trưng đó không quá quan trọng. Tuy nhiên, tốt hơn hết là chúng ta nên tìm cách xử lý dữ liệu bị khuyết thay vì xóa các đặc trưng của chúng đi.

Trong ví dụ ta đang xét, có hai đặc trưng là *TSThickness* và *SerumInsulin* có rất nhiều dữ liệu thiếu. Do đó, ta có thể xóa bỏ đi hai đặc trưng này.

Để làm điều đó trong Python, ta dùng câu lệnh sau:

```
### Dropping variables ###

# create another dataset for dropping variables

dv = dataset

# drop TSThickness variable

dv.drop('TSThickness', axis=1, inplace=True)

# drop SerumInsulin variable

dv.drop('SerumInsulin', axis=1, inplace=True)

print(dv.info())
```

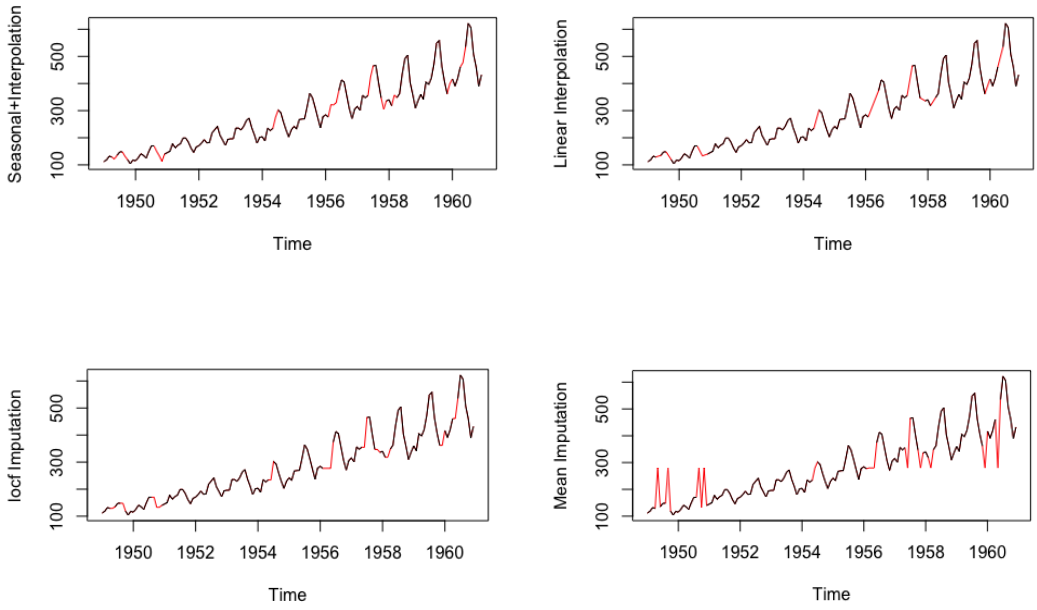
Điền vào dữ liệu bị khuyết

Như đã nói ở trên, các nhà phân tích dữ liệu thường tìm cách điền vào các dữ liệu bị khuyết thay vì xóa chúng khỏi bộ dữ liệu. Đối với từng loại dữ liệu khác nhau và từng bài toán khác nhau, người ta có những chiến lược xử lý khác nhau.

Điền vào dữ liệu bị khuyết dạng Time-Series

Trong thực tế hiện nay, khai thác dữ liệu dạng chuỗi thời gian đang tương đối phát triển với nhiều ứng dụng trong thực tế như dự đoán xu hướng thị trường, dự đoán xu hướng tăng, giảm của chứng khoán, phân tích sóng âm thanh, phân tích suy nghĩ dựa vào tín hiệu thời gian thực của mạng nơ-ron trong não người,… Do vậy, việc xử lý dữ liệu dạng này cũng đang rất được giới khoa học quan tâm. Cùng với thời gian, người ta đã đề xuất ra một số cách xử lý dữ liệu bị khuyết dạng này như sau:

- *Last Observation Carried Forward (LOCF) và Next Observation Carried Backward (NOCB)*: Đây là cách tiếp cận thống kê thông thường mà nó điền vào các giá trị bị khuyết dựa trên các giá trị đã tồn tại ngay trước hoặc ngay sau điểm dữ liệu bị khuyết. Hai phương pháp tiếp cận này đều có thể tạo ra các sai số đến quá trình phân tích dữ liệu, và càng tệ hơn khi dữ liệu có xu hướng tăng giảm rõ ràng.
- *Nội suy tuyến tính (Linear Interpolation):* Phương pháp này hoạt động tốt cho một chuỗi thời gian với một vài khuynh hướng (tăng hoặc giảm một cách tuyến tính) nhưng không phù hợp loại dữ liệu theo mùa (seasonal data – dữ liệu có xu hướng tuần hoàn theo chu kỳ).
- *Kết hợp giữa seasonal adjustment và nội suy tuyến tính*: Phương pháp này hoạt động tốt cho dữ liệu có xu hướng tăng, giảm và dữ liệu theo mùa.



Hình 3.1 So sánh các phương pháp điền vào dữ liệu time-series bị khuyết

Mời bạn đọc tiếp chủ để tiền xử lý dữ liệu trong phần tiếp thep (phần 2).

References:

1. https://statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf
2. https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
3. https://machinelearningmastery.com/handle-missing-data-python
4. https://arxiv.org/pdf/1710.01011.pdf
5. https://www.bu.edu/sph/files/2014/05/Marina-tech-report.pdf
6. *Time-Series Lecture at University of San Francisco by Nathaniel Stevens*