CHƯƠNG 5

Bắt đầu với Khoa học viễn tưởng kit-học cho Máy Học tập

Giới thiệu về Scikit-learning

Trong Chương 2–4, bạn đã học cách sử dụng Python cùng với các thư viện như NumPy và Pandas để thực hiện bẻ khóa số, trực quan hóa dữ liệu và phân tích. Đối với học máy, bạn cũng có thể sử dụng các thư viện này để xây dựng các mô hình học tập của riêng mình. Tuy nhiên, làm như vậy đòi hỏi bạn phải đánh giá cao nền tảng toán học cho các thuật toán học máy khác nhau — không phải là một vấn đề tầm thường.

Thay vì thực hiện các thuật toán học máy khác nhau theo cách thủ công, may mắn thay, đã có người khác thực hiện công việc khó khăn cho bạn. Giới thiệu*Scikit-learning*, một thư viện Python thực hiện các loại thuật toán học máy khác nhau, chẳng hạn như phân loại, hồi quy, phân cụm, cây quyết định, v.v. Sử dụng Scikit-learning, việc triển khai học máy giờ đây chỉ đơn giản là việc gọi một hàm với dữ liệu thích hợp để bạn có thể điều chỉnh và đào tạo mô hình.

Trong chương này, trước tiên, bạn sẽ tìm hiểu các địa điểm khác nhau nơi bạn có thể lấy bộ dữ liệu mẫu để tìm hiểu cách thực hiện học máy. Sau đó, bạn sẽ học cách sử dụng Scikit-learning để thực hiện hồi quy tuyến tính đơn giản trên một tập dữ liệu đơn giản. Cuối cùng, bạn sẽ học cách thực hiện xóa dữ liệu.

Nhận tập dữ liệu

Thông thường, một trong những thách thức trong học máy là lấy bộ dữ liệu mẫu để thử nghiệm. Trong học máy, khi bạn mới bắt đầu với một thuật toán, sẽ rất hữu ích khi bắt đầu với một tập dữ liệu đơn giản mà bạn có thể tự tạo để kiểm tra xem thuật toán có hoạt động chính xác theo sự hiểu biết của bạn hay không. Khi bạn hoàn thành giai đoạn này, đã đến lúc làm việc với một tập dữ liệu lớn và đối với điều này, bạn cần phải tìm nguồn liên quan để mô hình học máy của bạn có thể giống với thực tế nhất có thể.

Dưới đây là một số nơi bạn có thể lấy tập dữ liệu mẫu để thực hành học máy của mình:

- Bộ dữ liệu tích hợp của Scikit-learning
- Bộ dữ liệu Kaggle
- Kho lưu trữ Máy học UCI (Đại học California, Irvine)

Chúng ta hãy xem xét từng điều này trong các phần sau.

Sử dụng Bộ dữ liệu Scikit-learning

Scikit-learning đi kèm với một số bộ dữ liệu mẫu tiêu chuẩn, giúp việc học máy trở nên dễ dàng. Để tải tập dữ liệu mẫu, hãy nhập mô-đun tập dữ liệu và tải tập dữ liệu mong muốn. Ví dụ: các đoạn mã sau tải *Bộ dữ liệu Iris*:

```
from sklearn import datasets iris =
datasets.load_iris ()                    # dữ liệu thô thuộc loại Bunch
```

TIỀN BOA **Bộ dữ liệu hoa Iris hay bộ dữ liệu Fisher's Iris là một bộ dữ liệu đa biến được giới thiệu bởi nhà thống kê và nhà sinh vật học người Anh Ronald Fisher. Bộ dữ liệu bao gồm 50 mẫu của mỗi loài trong số ba loài Iris (Iris setosa, Iris virginica và Iris versicolor). Bốn đặc điểm được đo từ mỗi mẫu: chiều dài và chiều rộng của các lá đài và cánh hoa tính bằng cm. Dựa trên sự kết hợp của bốn đặc điểm này, Fisher đã phát triển một mô hình phân biệt tuyến tính để phân biệt các loài với nhau.**

Tập dữ liệu đã tải được biểu diễn dưới dạngBóđối tượng, một từ điển Python cung cấp quyền truy cập kiểu thuộc tính. Bạn có thể dùngMÔ TẢthuộc tính để có được mô tả của tập dữ liệu:

```
in (iris.DESCR)
```

Tuy nhiên, quan trọng hơn, bạn có thể có được các tính năng của tập dữ liệu bằng cách sử dụngdữ liệutài sản:

```
in (iris.data)                                    # Đặc trưng
```

Câu lệnh trước in ra như sau:

```
[[5,1 3,5 1,4 0,2] [4,9
                3. 1,4 0,2]
      . . .
  [6,2 3,4 5,4 2.3] [5,9 3. 5,1 1,8]]
```

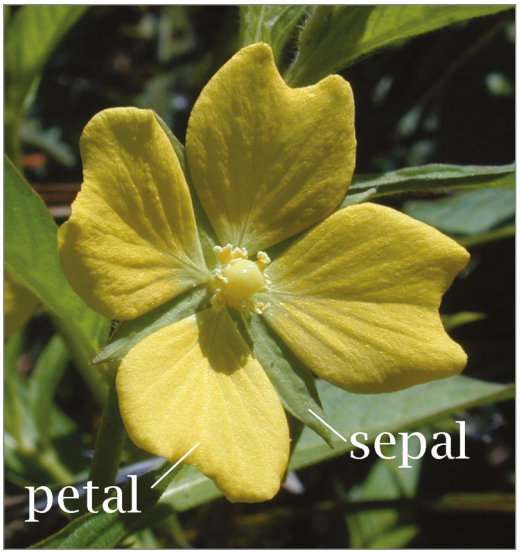
Bạn cũng có thể sử dụngfeature_namesthuộc tính in tên của các đối tượng địa lý:

```
print (iris.feature_names)                    # Tên tính năng
```

Câu lệnh trước in ra như sau:

```
['chiều dài đài hoa (cm)', 'chiều rộng đài hoa (cm)',
  'chiều dài cánh hoa (cm)', 'chiều rộng cánh hoa (cm)']
```

Điều này có nghĩa là tập dữ liệu chứa bốn cột — chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa và chiều rộng cánh hoa. Nếu bạn đang thắc mắc một cánh hoa và đài hoa là gì, thì Hình 5.1 cho thấy hoa tứ bội Ludwigia octovalvis cho thấy những cánh hoa và đài hoa (nguồn:https://en.wikipedia.org/wiki/Sepal).



Hình 5.1:Cánh hoa và đài hoa của hoa

Để in nhãn của tập dữ liệu, hãy sử dụngMục tiêutài sản. Đối với tên nhãn, hãy sử dụngtarget_namestài sản:

```
print (iris.target)                    # Nhãn
print (iris.target_names)              # Tên nhãn
```

Điều này in ra như sau:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 2 2 2 2 2 2
  2 2]
['setosa' 'versicolor' 'virginica']
```

Trong trường hợp này, 0 đại diện cho*setosa*, 1 đại diện*màu sắc*và 2 đại diện*virginica*.

TIỀN BOA **Lưu ý rằng không phải tất cả các bộ dữ liệu mẫu trong Scikit-learning đều hỗ trợ**tính năng_những cái tên **và**Mục tiêu_những cái tên**đặc tính.**

Hình 5.2 tóm tắt tập dữ liệu trông như thế nào.

chiều dài đài hoa	chiều rộng đài hoa	chiều dài cánh hoa	chiều rộng cánh hoa	Mục tiêu
5.1	3.5	1,4	0,2	*0*
4,9	3.0	1,4	0,2	*0*
…	…	…	…	…
5.9	3.0	5.1	1,8	*2*

0 đại diện cho setosa, 1 đại diện cho màu sắc, 2 đại diện cho virginica

Hình 5.2:Các trường trong tập dữ liệu Iris và mục tiêu của nó

Thông thường, rất hữu ích khi chuyển đổi dữ liệu sang khung dữ liệu Pandas để bạn có thể thao tác dễ dàng:

```
nhập gấu trúc dưới dạng pd
df = pd.DataFrame (iris.data)          # chuyển đổi tính năng
                                       # sang khung dữ liệu trong Pandas
print (df.head ())
```

Các câu lệnh này in ra như sau:

```
          0        1        2        3
0 5,1 3,5 1,4 0,2 1 4,9 3,0 1,4
0,2 2
      4,7 3,2 1,3 0,2
3     4,6 3,1 1,5 0,2
4 5,0 3,6 1,4 0,2
```

Bên cạnh tập dữ liệu Iris, bạn cũng có thể tải một số tập dữ liệu thú vị trong Scikitlearn, chẳng hạn như sau:

```
# dữ liệu về ung thư vú
vú_cancer = datasets.load_breast_cancer ()

# dữ liệu về bệnh tiểu đường
bệnh tiểu đường = datasets.load_diabetes ()

# tập dữ liệu gồm 1797 hình ảnh 8x8 gồm các chữ số viết tay =
datasets.load_digits ()
```

Để biết thêm thông tin về bộ dữ liệu Scikit-learning, hãy xem tài liệu tion athttp://scikit-learn.org/stable/datasets/index.html.

Sử dụng tập dữ liệu Kaggle

*Kaggle*là cộng đồng các nhà khoa học dữ liệu và máy học lớn nhất thế giới. Khởi đầu là một nền tảng cung cấp các cuộc thi học máy, Kaggle hiện cũng cung cấp một nền tảng dữ liệu công cộng, cũng như một bàn làm việc dựa trên đám mây cho các nhà khoa học dữ liệu. Google mua lại Kaggle vào tháng 3 năm 2017.

Đối với những người học máy học, bạn có thể sử dụng bộ dữ liệu mẫu do Kaggle cung cấp tạihttps://www.kaggle.com/datasets/.Một số bộ dữ liệu thú vị bao gồm:

- **Giá giày nữ:**Danh sách 10.000 đôi giày nữ và giá tại mà họ được bán (https://www.kaggle.com/datafiniti/womensshoes-prices)
- **Dữ liệu phát hiện mùa thu từ Trung Quốc:**Hoạt động của bệnh nhân cao tuổi cùng với thông tin y tế của họ (https://www.kaggle.com/pitasr/falldata)
- **Bán bất động sản NYC:**Giá trị của một năm bất động sản được bán trên NYC real thị trường bất động sản (https://www.kaggle.com/new-york-city/nyc-propertysales#nyc-rolling-sales.csv)
- **Chuyến bay Hoa Kỳ bị hoãn:**Các chuyến bay bị hoãn trong năm 2016 (https://www.kaggle.com/niranjan0272 / us-Flight-delay)

Sử dụng Kho lưu trữ Máy học UCI (Đại học California, Irvine)

Kho lưu trữ Máy học UCI (https://archive.ics.uci.edu/ml/ datasets.html)là tập hợp cơ sở dữ liệu, lý thuyết miền và trình tạo dữ liệu được cộng đồng học máy sử dụng để phân tích thực nghiệm

của các thuật toán học máy. Dưới đây là một số cái thú vị từ tập dữ liệu khổng lồ mà nó chứa:

- **Tập dữ liệu MPG tự động:**Tập hợp dữ liệu về hiệu suất nhiên liệu của different loại ô tô (https://archive.ics.uci.edu/ml/datasets/Auto+MPG)
- **Tập dữ liệu về Hiệu suất của Học sinh:**Dự đoán kết quả học tập của học sinh ở trường trung học giáo dục (trung học) (https://archive.ics.uci.edu/ml/datasets/ Sinh viên + Hiệu suất)
- **Tập dữ liệu thu nhập điều tra dân số:**Dự đoán liệu thu nhập có vượt quá $ 50K / năm hay không. dựa trên trên dữ liệu điều tra dân số (https://archive.ics.uci.edu/ml/datasets/census+income)

Tạo tập dữ liệu của riêng bạn

Nếu bạn không thể tìm thấy một tập dữ liệu phù hợp để thử nghiệm, tại sao không tạo một tập dữ liệu chính bạn? Cácsklearn.datasets.samples_generatormô-đun từ Scikit-thư viện học chứa một số chức năng để cho phép bạn tạo các loại tập dữ liệu khác nhau cho các loại vấn đề khác nhau. Bạn có thể sử dụng nó để tạo tập dữ liệu của các bản phân phối khác nhau, chẳng hạn như sau:

- Bộ dữ liệu được phân phối tuyến tính
- Bộ dữ liệu được phân nhóm
- Tập dữ liệu được phân nhóm được phân phối theo kiểu vòng tròn

Tập dữ liệu được phân phối tuyến tính

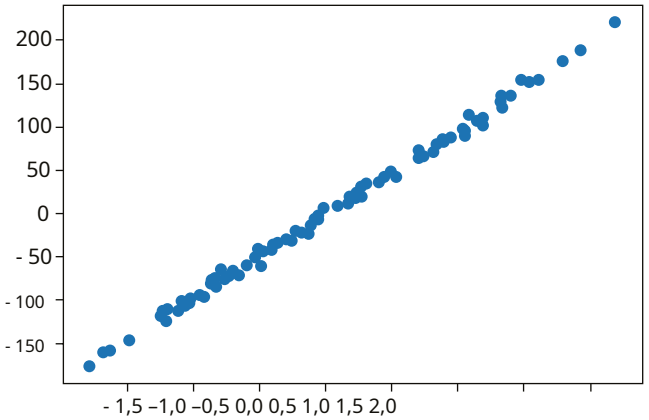
Cácmake_regression ()hàm tạo ra dữ liệu được phân phối tuyến tính. Bạn có thể chỉ định số lượng tính năng mà bạn muốn, cũng như độ lệch chuẩn của tiếng ồn Gaussian được áp dụng cho đầu ra:

```
% matplotlib inline
từ matplotlib nhập pyplot dưới dạng plt
từ sklearn.datasets.samples_generator nhập make_regression

X, y = make_regression (n_samples = 100, n_features = 1, noise = 5.4) plt.scatter (X, y)
```

Hình 5.3 cho thấy biểu đồ phân tán của tập dữ liệu được tạo.

Tập dữ liệu theo cụm

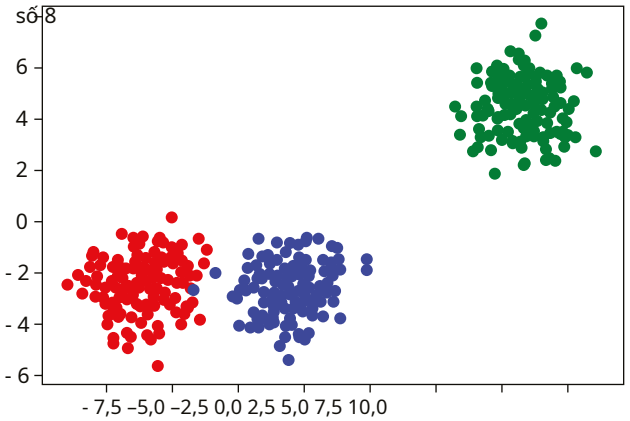
Cácmake_blobs ()chức năng tạo raNsố lượng các cụm dữ liệu ngẫu nhiên. Điều này rất hữu ích khi thực hiện phân cụm trong học tập không có giám sát (Chương 9, “Học tập có giám sát — Phân loại sử dụng K Nearest Neighbors (KNN)”):



Hình 5.3: Biểu đồ phân tán hiển thị các điểm dữ liệu được phân phối tuyến tính

```
% matplotlib inline
import matplotlib.pyplot as plt import
numpy as np
từ sklearn.datasets nhập make_blobs

X, y = make_blobs (500, center = 3)          # Tạo Gaussian đẳng hướng
                                             # đốm màu để phân nhóm

rgb = np.array (['r', 'g', 'b'])

# vẽ các đốm màu bằng cách sử dụng biểu đồ phân tán và sử dụng plt.scatter
mã hóa màu (X [:, 0], X [:, 1], color = rgb [y])
```

Hình 5.4 cho thấy biểu đồ phân tán của tập dữ liệu ngẫu nhiên được tạo.



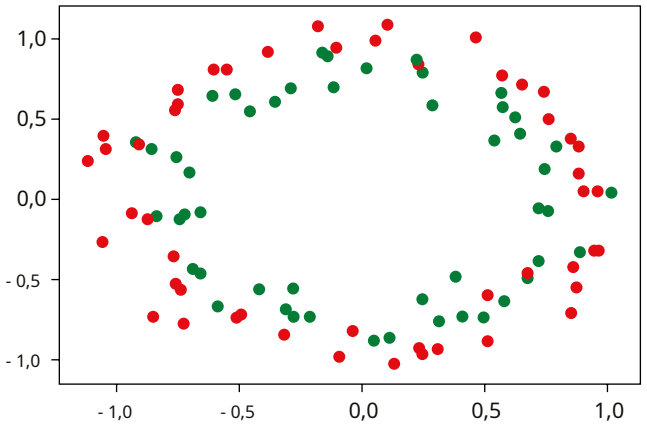
Hình 5.4: Biểu đồ phân tán hiển thị ba cụm điểm dữ liệu được tạo

Tập dữ liệu theo cụm được phân phối theo kiểu tròn

Cácmake_circles ()hàm tạo một tập dữ liệu ngẫu nhiên có chứa một vòng tròn lớn nhúng một vòng tròn nhỏ hơn theo hai chiều. Điều này rất hữu ích khi thực hiện phân loại, sử dụng các thuật toán như SVM (Hỗ trợ Máy vectơ). SVM sẽ được đề cập trong Chương 8, “Học tập có giám sát — Phân loại sử dụng SVM.”

```
% matplotlib inline
import matplotlib.pyplot as plt import
numpy as np
từ sklearn.datasets nhập make_circles

X, y = make_circles (n_samples = 100, noise = 0.09)

rgb = np.array (['r', 'g', 'b']) plt.scatter (X [:, 0], X [:, 1], color
= rgb [y])
```

Hình 5.5 cho thấy biểu đồ phân tán của tập dữ liệu ngẫu nhiên được tạo.



Hình 5.5: Biểu đồ phân tán hiển thị hai cụm điểm dữ liệu được phân phối theo kiểu hình tròn

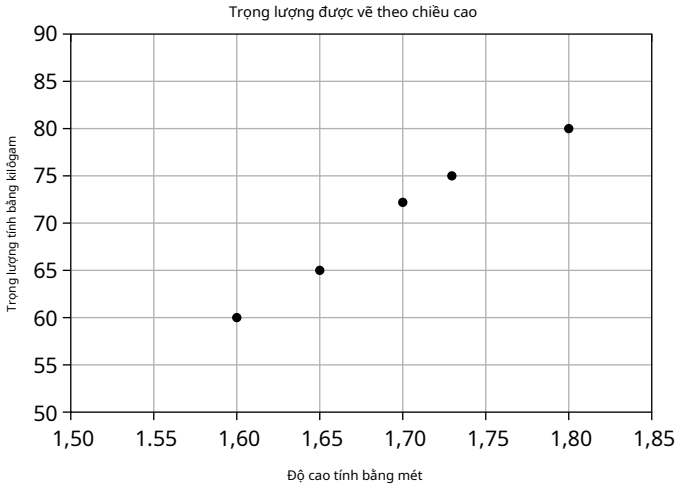
Bắt đầu với Scikit-learning

Cách dễ nhất để bắt đầu học máy với Scikit-learning là bắt đầu với hồi quy tuyến tính.*Hồi quy tuyến tính*là một cách tiếp cận tuyến tính để mô hình hóa mối quan hệ giữa một biến phụ thuộc vô hướng*y*và một hoặc nhiều biến giải thích (hoặc biến độc lập). Ví dụ: hãy tưởng tượng rằng bạn có một tập hợp dữ liệu bao gồm chiều cao (tính bằng mét) của một nhóm người và trọng lượng tương ứng của họ (tính bằng kg):

```
% matplotlib inline
nhập matplotlib.pyplot dưới dạng plt
```

```
# đại diện cho chiều cao của một nhóm người theo mét chiều cao = [[1.6],
[1.65], [1.7], [1.73], [1.8]]

# đại diện cho trọng lượng của một nhóm người theo trọng lượng kg =
[[60], [65], [72,3], [75], [80]]

plt.title ('Trọng lượng được vẽ theo chiều cao') plt.xlabel
('Chiều cao tính bằng mét')
plt.ylabel ('Trọng lượng tính bằng kilogam')

plt.plot (chiều cao, trọng lượng, 'k.')

# dải trục cho x và y plt.axis ([1,5, 1,85,
50, 90]) plt.grid (Đúng)
```

Khi bạn vẽ biểu đồ trọng lượng so với chiều cao, bạn sẽ thấy biểu đồ như trong Hình 5.6.



Hình 5.6:Vẽ biểu đồ trọng lượng so với chiều cao cho một nhóm người

Từ biểu đồ, bạn có thể thấy rằng có mối tương quan thuận giữa cân nặng và chiều cao của nhóm người này. Bạn có thể vẽ một đường thẳng qua các điểm và sử dụng nó để dự đoán cân nặng của một người khác dựa trên chiều cao của họ.

Sử dụng lớp tuyến tính để lắp mô hình

Vậy làm thế nào để vẽ đường thẳng cắt mặc dù tất cả các điểm? Nó chỉ ra rằng thư viện Scikit-learning cóTuyến tínhlớp học giúp bạn làm điều đó. Tất cả những gì bạn cần làm là tạo một phiên bản của lớp này và sử dụng

các***chiều cao***và***trọng lượng***danh sách để tạo mô hình hồi quy tuyến tính bằng cách sử dụngPhù hợp() chức năng, như thế này:

```
từ sklearn.linear_model import LinearRegression

# Tạo và điều chỉnh mô hình model =
LinearRegression () model.fit (X = heights,
y = weights)
```

TIỀN BOA **Quan sát rằng*chiều cao*và*trọng lượng*cả hai đều được đại diện là danh sách hai chiều. Điều này là bởi vì**Phù hợp()**chức năng yêu cầu cả hai*X*và*y* đối số là hai chiều (thuộc loại**danh sách**hoặc**ndarray**).**

Đưa ra dự đoán

Khi bạn đã trang bị (được đào tạo) mô hình, bạn có thể bắt đầu đưa ra dự đoán bằng cách sử dụngdự đoán()chức năng, như thế này:

```
# Đưa ra dự đoán
weight = model.p Dự đoán ([[1.75]]) [0] [0] print
(round (weight, 2))                    # 76.04
```

Trong ví dụ trước, bạn muốn dự đoán cân nặng của một người Cao 1,75m. Dựa trên mô hình, trọng lượng được dự đoán là 76,04kg.

TIỀN BOA **Trong Scikit-learning, bạn thường sử dụng**Phù hợp()**chức năng đào tạo một mô hình. Sau khi mô hình được đào tạo, bạn sử dụng**dự đoán()**chức năng để đưa ra một dự đoán.**

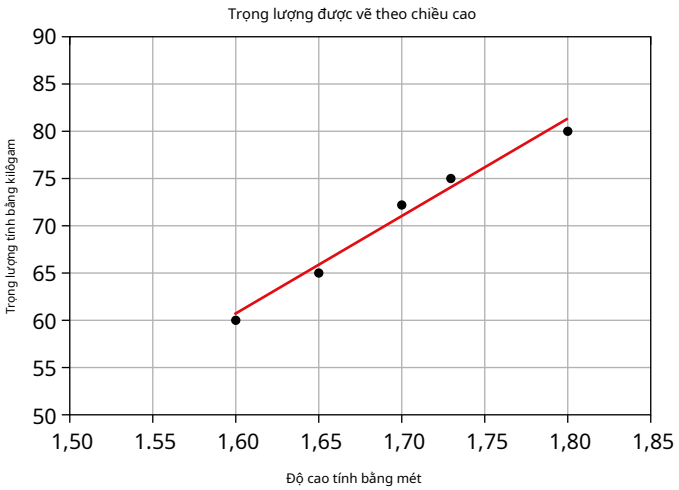
Vẽ đường hồi quy tuyến tính

Sẽ rất hữu ích khi hình dung đường hồi quy tuyến tính đã được tạo bởiTuyến tínhlớp. Hãy làm điều này trước tiên bằng cách vẽ biểu đồ các điểm dữ liệu ban đầu và sau đó gửi***chiều cao***danh sách vào mô hình để dự đoán trọng lượng. Sau đó, chúng tôi vẽ biểu đồ chuỗi các trọng số dự báo để thu được đường. Đoạn mã sau đây cho biết cách thực hiện điều này:

```
nhập matplotlib.pyplot dưới dạng plt

chiều cao = [[1.6], [1.65], [1.7], [1.73], [1.8]] cân nặng = [[60], [65],
[72.3], [75], [80]]

plt.title ('Trọng lượng được vẽ theo chiều cao') plt.xlabel
('Chiều cao tính bằng mét')
plt.ylabel ('Trọng lượng tính bằng kilogam')
```

```
plt.plot (chiều cao, trọng lượng, 'k.')

plt.axis ([1,5, 1,85, 50, 90]) plt.grid
(Đúng)

# vẽ đường hồi quy
plt.plot (chiều cao, mô hình.p dự đoán (chiều cao), color = 'r')
```

Hình 5.7 cho thấy đường hồi quy tuyến tính.



Hình 5.7:Vẽ đường hồi quy tuyến tính

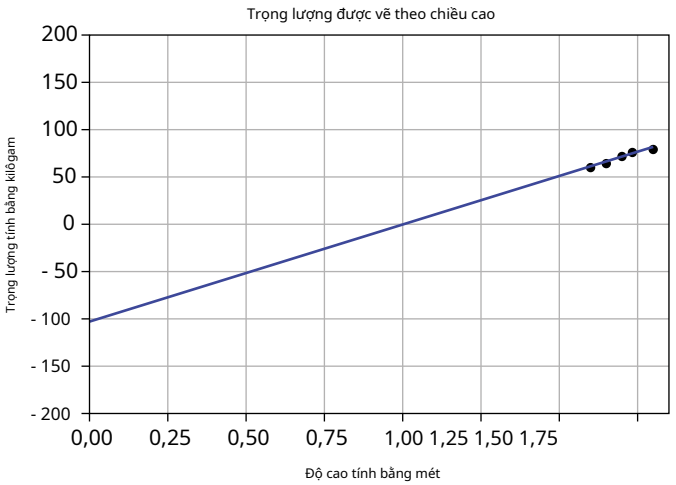
Lấy Gradient và Intercept của đường hồi quy tuyến tính

Từ Hình 5.7, không rõ giá trị nào của đường hồi quy tuyến tính chặn trục y. Điều này là do chúng tôi đã điều chỉnh trục x để bắt đầu vẽ đồ thị ở mức 1,5. Cách tốt hơn để hình dung điều này là đặt trục x bắt đầu từ 0 và phóng to phạm vi của trục y. Sau đó, bạn vẽ đường thẳng bằng cách cung cấp hai giá trị cực đại của chiều cao: 0 và 1,8. Đoạn mã sau vẽ lại các điểm và đường hồi quy tuyến tính:

```
plt.title ('Trọng lượng được vẽ theo chiều cao') plt.xlabel
('Chiều cao tính bằng mét')
plt.ylabel ('Trọng lượng tính bằng kilogam')

plt.plot (chiều cao, trọng lượng, 'k.')

plt.axis ([0, 1.85, -200, 200]) plt.grid
(Đúng)
```

```
# vẽ biểu đồ của dòng hồi quy
Extreme_heights = [[0], [1.8]]
plt.plot (Extreme_heights, model.p Dự đoán (Extreme_heights), color = 'b')
```

Hình 5.8 bây giờ cho thấy điểm mà đường thẳng cắt trục y.



Hình 5.8:Đường hồi quy tuyến tính

Trong khi bạn có thể nhận được điểm chặn y bằng cách dự đoán cân nặng nếu chiều cao là 0:

```
vòng (mô hình.p dự đoán ([[0]]) [0] [0], 2)              # - 104,75
```

các***người mẫu***đối tượng cung cấp câu trả lời trực tiếp thông quachặn_tài sản:

```
print (round (model.intercept_ [0], 2))              # - 104,75
```

Sử dụng***người mẫu***đối tượng, bạn cũng có thể nhận được gradient của đường hồi quy tuyến tính thông quacoef_tài sản:

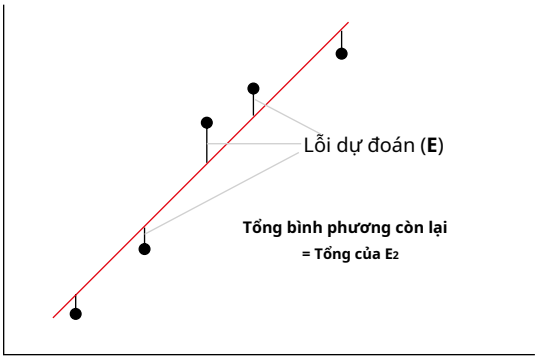
```
print (round (model.coef_ [0] [0], 2))                  # 103.31
```

Kiểm tra hiệu suất của mô hình bằng cách tính tổng dư của hình vuông

Để biết liệu đường hồi quy tuyến tính của bạn có phù hợp với tất cả các điểm dữ liệu hay không, chúng tôi sử dụng*Tổng bình phương còn lại (RSS)*phương pháp. Hình 5.9 cho thấy cách tính RSS.

Đoạn mã sau đây cho thấy cách tính RSS bằng Python:

```
nhập numpy dưới dạng np

print ('Tổng số dư của các ô vuông:% .2f'%
          np.sum ((cân nặng - mô hình.p dự đoán (chiều cao)) ** 2))
# Tổng dư của bình phương: 5,34
```



Hình 5.9:Tính tổng dư của bình phương cho hồi quy tuyến tính

RSS phải càng nhỏ càng tốt, với 0 cho thấy rằng đường hồi quy khớp chính xác với các điểm (hiếm khi có thể đạt được trong thế giới thực).

Đánh giá mô hình bằng tập dữ liệu thử nghiệm

Bây giờ mô hình của chúng tôi đã được đào tạo với dữ liệu đào tạo của chúng tôi, chúng tôi có thể đưa nó vào thử nghiệm. Giả sử rằng chúng ta có tập dữ liệu thử nghiệm sau:

```
# dữ liệu thử nghiệm
heights_test = [[1.58], [1.62], [1.69], [1.76], [1.82]] weights_test = [[58], [63],
[72], [73], [85]]
```

chúng tôi có thể đo lường mức độ chặt chẽ của dữ liệu thử nghiệm phù hợp với đường hồi quy bằng cách sử dụng *Phương pháp bình phương R*. Phương pháp R-Squared còn được gọi là*hệ số xác định*, hoặc là*hệ số của nhiều phép xác định cho nhiều phép hồi quy*.

Công thức tính R-Squared được thể hiện trong Hình 5.10.

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$TSS = \sum_{tôi=1}^{N} (y_{tôi} - \bar{y})^2$$

$$RSS = \sum_{tôi=1}^{N} (y_{tôi} - f(x_{tôi}))^2$$

Hình 5.10:Công thức tính R-Squared

Sử dụng công thức được hiển thị cho R-Squared, hãy lưu ý những điều sau:

- R^2là R bình phương
- TSS là Tổng hình vuông
- RSS là Tổng bình phương còn lại

Bây giờ bạn có thể tính toán nó bằng Python bằng cách sử dụng đoạn mã sau:

```
# Tổng bình phương (TSS)
weights_test_mean = np.mean (np.ravel (weights_test)) TSS = np.sum
((np.ravel (weights_test) -
                    weights_test_mean) ** 2)
print ("TSS:% .2f"% TSS)

# Tổng bình phương còn lại (RSS) RSS = np.sum
((np.ravel (weights_test) -
                    np.ravel (model.p Dự đoán (heights_test)))
                        * * 2)
print ("RSS:% .2f"% RSS)

# R_squared
R_squared = 1 - (RSS / TSS) print ("R-squared:%
.2f"% R_squared)
```

TIỀN BOA **Các**ravel ()**hàm chuyển đổi danh sách hai chiều thành liền kề mảng phẳng (một chiều).**

Đoạn mã trước đó mang lại kết quả sau:

```
TSS: 430,80
RSS: 24,62
Bình phương R: 0,94
```

May mắn thay, bạn không phải tự mình tính toán Bình phương R bằng tay — Scikit-learning cóghi bàn()để tính toán R-Squared tự động cho bạn:

```
# using scikit-learning để tính toán r bình phương in ('R bình phương:
% .4f'% model.score (heights_test,
                                                    weights_test))

# R bình phương: 0,9429
```

Giá trị R-Squared là 0,9429 (94,29%) cho thấy mức độ phù hợp khá tốt cho dữ liệu thử nghiệm của bạn.

Kiên trì mô hình

Khi bạn đã đào tạo một mô hình, bạn có thể lưu nó để sử dụng sau này thường rất hữu ích. Thay vì đào tạo lại mô hình mỗi khi bạn có dữ liệu mới để kiểm tra, mô hình đã lưu cho phép bạn tải mô hình đã đào tạo và đưa ra dự đoán ngay lập tức mà không cần đào tạo lại mô hình.

Có hai cách để lưu mô hình được đào tạo của bạn bằng Python:

- Sử dụng tiêu chuẩndưa muốimô-đun bằng Python để tuần tự hóa và giải mã hóa các đối tượng
- Sử dụngjoblibmô-đun trong Scikit-learning được tối ưu hóa để lưu và tải các đối tượng Python xử lý dữ liệu NumPy

Ví dụ đầu tiên bạn sẽ thấy là lưu mô hình bằng cách sử dụngdưa muốimô-đun:

```
nhập khẩu dưa chua

# lưu mô hình vào đĩa
filename = 'HeightsAndWeights_model.sav'
# ghi vào tệp bằng ghi và chế độ nhị phân pickle.dump (model,
open (tên tệp, 'wb'))
```

Trong đoạn mã trước đó, lần đầu tiên bạn mở một tệp trong "wb "cách thức ("w "để viết và "b "đối với hệ nhị phân). Sau đó, bạn sử dụngbãi rác()chức năng từdưa muối để lưu mô hình vào tệp.

Để tải mô hình từ tệp, hãy sử dụngtrọng tải()hàm số:

```
# tải mô hình từ đĩa
Load_model = pickle.load (mở (tên tệp, 'rb'))

Bây giờ bạn có thể sử dụng mô hình như bình thường:

result = loading_model.score (heights_test,
                                        weights_test)
```

Sử dụngjoblibmô-đun rất giống với việc sử dụngdưa muốimô-đun:

```
từ sklearn.externals import joblib

# lưu mô hình vào đĩa
filename = 'HeightsAndWeights_model2.sav'
joblib.dump (model, filename)

# tải mô hình từ disk loading_model = joblib.load
(filename) result = loading_model.score (heights_test,

                                        weights_test)
in (kết quả)
```

Dọn dẹp dữ liệu

Trong học máy, một trong những tác vụ đầu tiên bạn cần thực hiện là*dọn dẹp dữ liệu*. Rất hiếm khi bạn có một tập dữ liệu mà bạn có thể sử dụng ngay lập tức để đào tạo mô hình của mình. Thay vào đó, bạn phải kiểm tra dữ liệu cẩn thận xem có bất kỳ

các giá trị bị thiếu và xóa chúng hoặc thay thế chúng bằng một số giá trị hợp lệ hoặc bạn phải chuẩn hóa chúng nếu có các cột có giá trị cực kỳ khác nhau. Các phần sau đây trình bày một số tác vụ phổ biến bạn cần thực hiện khi làm sạch dữ liệu của mình.

Làm sạch hàng bằng NaN

Hãy xem xét một tệp CSV có tênNaNDataset.csvvới nội dung sau:

```
A, B, C
1,2,3
4, 6
7, 9
10,11,12
13,14,15
16,17,18
```

Bằng mắt thường, bạn có thể nhận ra rằng có một vài hàng có các trường trống. Cụ thể, hàng thứ hai và thứ ba bị thiếu giá trị cho cột thứ hai. Đối với các tập dữ liệu nhỏ, điều này rất dễ nhận ra. Nhưng nếu bạn có một tập dữ liệu lớn, nó gần như không thể bị phát hiện. Một cách hiệu quả để phát hiện các hàng trống là tải tập dữ liệu vào khung dữ liệu Pandas và sau đó sử dụngisnull ()hàm để kiểm tra các giá trị null trong khung dữ liệu:

```
nhập gấu trúc dưới dạng pd
df = pd.read_csv ('NaNDataset.csv') df.isnull ().
sum ()
```

Đoạn mã này sẽ tạo ra kết quả sau:

```
Một   0
B     2
C     0
dtype: int64
```

Bạn có thể thấy rằng cột B có hai giá trị rỗng. Khi Pandas tải một tập dữ liệu có chứa các giá trị trống, nó sẽ sử dụngNaNđể đại diện cho các trường trống đó. Sau đây là kết quả đầu ra của khung dữ liệu khi bạn in nó ra:

```
     Một    B     C
0    1     2,0 3
1    4     NaN 6
2    7     NaN 9
3 10 11,0 12
4 13 14,0 15
5 16 17,0 18
```

Thay thế NaN bằng Giá trị trung bình của cột

Một trong những cách để đối phó vớiNaNs trong tập dữ liệu của bạn là để thay thế chúng bằng giá trị trung bình của các cột mà chúng nằm trong đó. Đoạn mã sau thay thế tất cảNaNs trong cột B với giá trị trung bình của cột B:

```
# thay thế tất cả các NaN trong cột B bằng giá trị trung bình của cột B df.B = df.B.fillna
(df.B.mean ())
in (df)
```

Khung dữ liệu bây giờ trông như thế này:

```
      Một     B     C
0     1     2,0 3
1     4    11.0     6
2     7    11.0     9
3 10 11,0 12
4 13 14,0 15
5 16 17,0 18
```

Xóa hàng

Một cách khác để đối phó vớiNaNs trong tập dữ liệu của bạn chỉ đơn giản là loại bỏ các hàng chứa chúng. Bạn có thể làm như vậy bằng cách sử dụngdropna ()chức năng, như thế này:

```
df = pd.read_csv ('NaNDataset.csv') df =
df.dropna ()                                    # thả tất cả các hàng với NaN
in (df)
```

Đoạn mã này sẽ tạo ra kết quả sau:

```
      Một     B     C
0     1     2,0 3
3 10 11,0 12
4 13 14,0 15
5 16 17,0 18
```

Quan sát rằng sau khi loại bỏ các hàng có chứaNaN,chỉ mục không còn theo thứ tự tuần tự. Nếu bạn cần đặt lại chỉ mục, hãy sử dụngreset_index ()hàm số:

```
df = df.reset_index (drop = True) print (df)                # đặt lại chỉ mục
```

Khung dữ liệu với chỉ mục đặt lại bây giờ sẽ giống như sau:

```
      Một     B     C
0     1     2,0 3
1 10 11,0 12
2 13 14,0 15
3 16 17,0 18
```

Loại bỏ các hàng trùng lặp

Hãy xem xét một tệp CSV có tênDuplicateRows.csvvới nội dung sau:

```
A, B, C
1,2,3
4,5,6
4,5,6
7,8,9
7,18,9
10,11,12
10,11,12
13,14,15
16,17,18
```

Để tìm tất cả các hàng trùng lặp, trước tiên hãy tải tập dữ liệu vào khung dữ liệu và sau đó sử dụngnhân đôi ()chức năng, như thế này:

```
nhập gấu trúc dưới dạng pd
df = pd.read_csv ('DuplicateRows.csv') print
(df.duplicated (keep = False))
```

Điều này sẽ tạo ra kết quả sau:

```
0      Sai
1       ĐÚNG VẬY
2       ĐÚNG VẬY
3      Sai
4      Sai
5       ĐÚNG VẬY
6       ĐÚNG VẬY
7      Sai
số 8   Sai
dtype: bool
```

Nó cho thấy những hàng nào bị trùng lặp. Trong ví dụ này, các hàng có chỉ mục 1, 2, 5 và 6 là các hàng trùng lặp. Cácgiữ chođối số cho phép bạn chỉ định cách chỉ ra các bản sao:

- Mặc định là 'Đầu tiên':Tất cả các bản sao được đánh dấu làĐÚNG VẬYngoại trừ lần xuất hiện đầu tiên
- 'Cuối cùng':Tất cả các bản sao được đánh dấu làĐÚNG VẬYngoại trừ lần xuất hiện cuối cùng
- Sai:Tất cả các bản sao được đánh dấu làĐÚNG VẬY

Vì vậy, nếu bạn đặtgiữ chođến 'Đầu tiên',bạn sẽ thấy kết quả sau:

```
0      Sai
1      Sai
2       ĐÚNG VẬY
```

```
3       Sai
4       Sai
5       Sai
6       ĐÚNG VẬY
7       Sai
số 8    Sai
dtype: bool
```

Do đó, nếu bạn muốn xem tất cả các hàng trùng lặp, bạn có thể đặtgiữ chođếnSaivà sử dụng kết quả củanhân đôi ()hoạt động như một chỉ mục trong khung dữ liệu:

```
print (df [df.duplicated (keep = False)])
```

Câu lệnh trước sẽ in tất cả các hàng trùng lặp:

```
      Một  B     C
1     4     5     6
2     4     5     6
5 10 11 12
6 10 11 12
```

Để loại bỏ các hàng trùng lặp, bạn có thể sử dụngdrop_duplicates ()chức năng, như thế này:

```
df.drop_duplicates (keep = 'first', inplace = True) # loại bỏ các bản sao và giữ
lại bản đầu tiên
in (df)
```

TIỀN BOA **Theo mặc định,**làm rơi_bản sao ()**chức năng sẽ không sửa đổi bản gốc dataframe và sẽ trả về dataframe chứa các hàng bị loại bỏ. Nếu bạn muốn sửa đổi khung dữ liệu ban đầu, hãy đặt**tại chỗ**tham số cho**ĐÚNG VẬY**, như được hiển thị trong đoạn mã trước.**

Các câu lệnh trước sẽ in ra như sau:

```
      Một  B     C
0     1     2     3
1     4     5     6
3     7     số 8  9
4     7 18        9
5 10 11 12
7 13 14 15
8 16 17 18
```

TIỀN BOA **Để xóa tất cả các bản sao, hãy đặt**giữ cho**tham số cho**Sai**. Để giữ lần xuất hiện cuối cùng của các hàng trùng lặp, hãy đặt**giữ cho**tham số cho**'Cuối cùng'**.**

Đôi khi, bạn chỉ muốn loại bỏ các bản sao được tìm thấy trong các cột nhất định trong tập dữ liệu. Ví dụ: nếu bạn nhìn vào tập dữ liệu mà chúng tôi có

đang sử dụng, hãy quan sát rằng đối với hàng 3 và hàng 4, các giá trị của cột A và C giống hệt nhau:

```
      Một  B      C
3     7      số 8  9
4     7 18         9
```

Bạn có thể loại bỏ các bản sao trong các cột nhất định bằng cách chỉ địnhtập hợp con tham số:

```
df.drop_duplicates (subset = ['A', 'C'], keep = 'last',
                                          inplace = True)          # loại bỏ tất cả các bản sao trong
                                                                   # cột A và C và giữ nguyên
                                                                   # cuối cùng
in (df)
```

Câu lệnh này sẽ mang lại kết quả như sau:

```
      Một  B      C
0     1      2      3
1     4      5      6
4     7 18         9
5 10 11 12
7 13 14 15
8 16 17 18
```

Chuẩn hóa các cột

Chuẩn hóa là một kỹ thuật thường được áp dụng trong quá trình làm sạch dữ liệu. Mục đích của*bình thường hóa*là thay đổi giá trị của các cột số trong tập dữ liệu để sử dụng một thang đo chung, mà không sửa đổi sự khác biệt trong phạm vi giá trị.

Chuẩn hóa là rất quan trọng đối với một số thuật toán để mô hình hóa dữ liệu một cách chính xác. Ví dụ: một trong các cột trong tập dữ liệu của bạn có thể chứa các giá trị từ 0 đến 1, trong khi một cột khác có các giá trị từ 400.000 đến 500.000. Sự chênh lệch lớn về quy mô của các con số có thể gây ra các vấn đề khi bạn sử dụng hai cột để đào tạo mô hình của mình. Sử dụng chuẩn hóa, bạn có thể duy trì tỷ lệ của các giá trị trong hai cột trong khi vẫn giữ chúng ở một phạm vi giới hạn. Trong Pandas, bạn có thể sử dụngMinMaxScalerlớp để chia tỷ lệ mỗi cột thành một phạm vi giá trị cụ thể.

Hãy xem xét một tệp CSV có tênNormalizeColumns.csvvới nội dung sau:

```
A, B, C
1000,2,3
400,5,6
700,6,9
100,11,12
1300,14,15
1600,17,18
```

Đoạn mã sau sẽ chia tỷ lệ tất cả các giá trị của cột thành phạm vi (0,1):

```
nhập gấu trúc dưới dạng pd
từ quá trình tiền xử lý nhập khẩu sklearn

df = pd.read_csv ('NormalizeColumns.csv') x =
df.values.astype (float)

min_max_scaler = preprocessing.MinMaxScaler () x_scaled =
min_max_scaler.fit_transform (x) df = pd.DataFrame (x_scaled,
column = df.columns) print (df)
```

Bạn sẽ thấy kết quả sau:

```
        Một              B        C
0 0,6 0,000000 0,0 1 0,2
0,200000 0,2
2 0,4 0,266667 0,4
3 0,0 0,600000 0,6
4 0,8 0,800000 0,8
5 1,0 1,000000 1,0
```

Loại bỏ các yếu tố ngoại lai

Trong thống kê, một*ngoại lệ*là một điểm nằm xa các điểm quan sát khác. Ví dụ: đã cho một bộ giá trị — 234, 267, 1, 200, 245, 300, 199, 250, 8999 và 245 — rõ ràng là 1 và 8999 là các giá trị ngoại lệ. Chúng nổi bật rõ ràng so với phần còn lại của các giá trị và chúng "nằm ngoài" hầu hết các giá trị khác trong tập dữ liệu; do đó từ*ngoại lệ*. Các lỗi ngoại lệ xảy ra chủ yếu do lỗi trong quá trình ghi hoặc lỗi thử nghiệm và trong học máy, điều quan trọng là phải loại bỏ chúng trước khi đào tạo mô hình của bạn vì nó có thể có khả năng làm sai lệch mô hình của bạn nếu bạn không thực hiện.

Có một số kỹ thuật để loại bỏ các ngoại lệ và trong chương này, chúng ta thảo luận về hai trong số chúng:

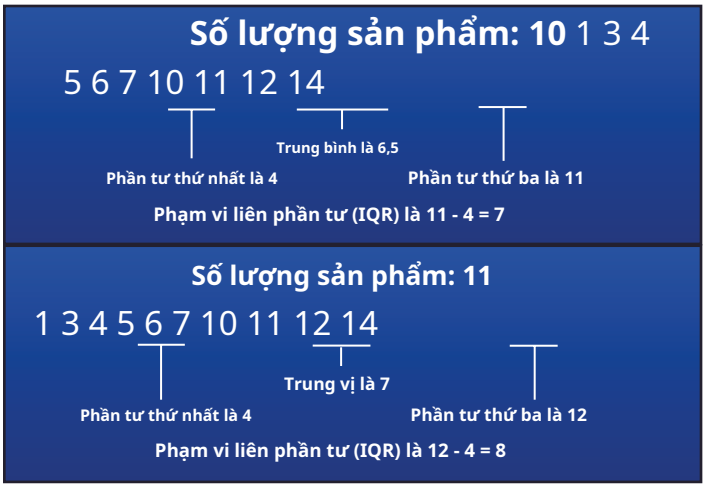
- Hàng rào Tukey
- Điểm Z

Hàng rào Tukey

Tukey Fences dựa trên*Dải phân vị (IQR)*. IQR là sự khác biệt giữa phần tư thứ nhất và phần tư thứ ba của một bộ giá trị. Phần tư đầu tiên, được ký hiệu là Q1, là giá trị trong tập dữ liệu chứa 25% giá trị bên dưới nó. Phần tư thứ ba,

được ký hiệu là Q3, là giá trị trong tập dữ liệu chứa 25% giá trị trên nó. Do đó, theo định nghĩa, IQR = Q3 - Q1.

Hình 5.11 cho thấy một ví dụ về cách thu được IQR đối với các tập dữ liệu có số giá trị chẵn và lẻ.



Hình 5.11:Ví dụ về việc tìm Dải phân vị (IQR)

Trong Tukey Fences, các giá trị ngoại lệ là các giá trị như sau:

- Dưới Q1 - (1,5 × IQR), hoặc
- Hơn Q3 + (1,5 × IQR)

Đoạn mã sau cho thấy việc triển khai Tukey Fences bằng Python:

```
nhập numpy dưới dạng np

def outlier_iqr (dữ liệu):
    q1, q3 = np.percentile (dữ liệu, [25, 75]) iqr = q3 - q1

    low_bound = q1 - (iqr * 1.5) upper_bound
    = q3 + (iqr * 1.5)
    return np.where ((data> upper_bound) | (data <Lower_bound))
```

TIỀN BOA **Các**np.where ()**hàm trả về vị trí của các mục thỏa mãn các điều kiện.**

Cáoutlier_iqr ()hàm trả về một bộ giá trị trong đó phần tử đầu tiên là một mảng chỉ số của những hàng có giá trị lớn hơn.

Để kiểm tra Hàng rào Tukey, hãy sử dụng tập dữ liệu Galton nổi tiếng về chiều cao của cha mẹ và con cái của họ. Bộ dữ liệu chứa dữ liệu dựa trên nghiên cứu nổi tiếng năm 1885 của Francis Galton khám phá mối quan hệ giữa chiều cao của trẻ em trưởng thành và chiều cao của cha mẹ chúng. Mỗi trường hợp là một đứa trẻ trưởng thành và các biến như sau:

Gia đình: Gia đình mà trẻ thuộc về, được gắn nhãn bởi các số từ 1 đến 204 và 136A **Bố:**Chiều cao của người cha, tính bằng inch **Mẹ:**Chiều cao của người mẹ, tính bằng inch **Giới tính:**Giới tính của trẻ, nam (M) hay nữ (F) **Chiều cao:**Chiều cao của đứa trẻ, tính bằng inch **Trẻ em:**Số trẻ em trong gia đình của trẻ em

Bộ dữ liệu có 898 trường hợp.
Đầu tiên, nhập dữ liệu:

```
nhập gấu trúc dưới dạng pd
df = pd.read_csv ("http://www.mosaic-web.org/go/datasets/galton.csv") print (df.head ())
```

Bạn sẽ thấy những điều sau:

	gia đình	sex cha me		Chiều cao	nkids
0	1	78,5 67,0	M	73,2	4
1	1	78,5 67,0	F	69,2	4
2	1	78,5 67,0	F	69.0	4
3	1	78,5 67,0	F	69.0	4
4	2	75,5 66,5 triệu		73,5	4

Nếu bạn muốn tìm những điểm khác biệt trongChiều cao, bạn có thể gọi outlier_iqr ()chức năng như sau:

```
print ("Các giá trị ngoại lệ sử dụng outlier_iqr ()")
print ("==============================") cho tôi
trong outlier_iqr (df .height) [0]:
    print (df [i: i + 1])
```

Bạn sẽ thấy kết quả sau:

```
Các giá trị ngoại lai sử dụng outlier_iqr
() ============================
     gia đình  cha       sex mẹ          Chiều cao  nkids
288        72    70.0       65.0 triệu       79.0        7
```

Sử dụng phương pháp Tukey Fences, bạn có thể thấy rằngChiều caocột có một ngoại lệ duy nhất.

Điểm Z

Phương pháp thứ hai để xác định các ngoại lệ là sử dụng*Điểm Z*phương pháp. Điểm Z cho biết điểm dữ liệu có bao nhiêu độ lệch chuẩn so với giá trị trung bình. Điểm Z có công thức sau:

$$= \left(\mathbf{x}_{\text{tôi}} - \mu\right) / \sigma$$

nơi $x_{tôi}$là điểm dữ liệu, μ là giá trị trung bình của tập dữ liệu và σ là độ lệch chuẩn.

Đây là cách bạn diễn giải điểm Z:

- Điểm Z âm cho biết điểm dữ liệu nhỏ hơn giá trị trung bình và điểm Z dương cho biết điểm dữ liệu được đề cập lớn hơn điểm trung bình
- Điểm Z là 0 cho bạn biết rằng điểm dữ liệu nằm ngay giữa (trung bình) và điểm Z là 1 cho bạn biết rằng điểm dữ liệu của bạn cao hơn 1 độ lệch chuẩn so với mức trung bình, v.v.
- Bất kỳ điểm Z nào lớn hơn 3 hoặc nhỏ hơn –3 được coi là ngoại lệ

Đoạn mã sau cho thấy việc triển khai Z-score bằng Python:

```
def outlier_z_score (dữ liệu):
    ngưỡng = 3
    mean = np.mean (dữ liệu)
    std = np.std (dữ liệu)
    z_scores = [(y - mean) / std cho y trong dữ liệu] return
    np.where (np.abs (z_scores)> ngưỡng)
```

Sử dụng cùng một tập dữ liệu Galton mà bạn đã sử dụng trước đó, bây giờ bạn có thể tìm thấy các ngoại lệ choChiều caocột bằng cách sử dụngoutlier_z_score ()hàm số:

```
print ("Các giá trị ngoại lệ sử dụng outlier_z_score ()")
print ("==================================") cho tôi
trong outlier_z_score (df.height) [0]:
    print (df [i: i + 1])
in()
```

Bạn sẽ thấy kết quả sau:

```
Các giá trị ngoại lai sử dụng outlier_z_score ()
==================================
     gia đình  cha   sex mẹ          Chiều cao  nkids
125        35    71.0     69.0 triệu       78.0       5
     gia đình  cha   sex mẹ          Chiều cao  nkids
288        72    70.0     65.0 triệu       79.0       7
     gia đình  cha   sex mẹ          Chiều cao  nkids
672       155    68.0     60.0     F       56.0       7
```

Sử dụng phương pháp điểm Z, bạn có thể thấy rằngChiều caocột có ba giá trị ngoại lệ.

Bản tóm tắt

Trong chương này, bạn đã biết cách bắt đầu với thư viện Scikit-learning để giải quyết vấn đề hồi quy tuyến tính. Ngoài ra, bạn cũng đã học được cách lấy bộ dữ liệu mẫu, tạo bộ dữ liệu của riêng bạn, thực hiện làm sạch dữ liệu, cũng như hai kỹ thuật mà bạn có thể sử dụng để loại bỏ các ngoại lệ khỏi bộ dữ liệu của mình.

Trong các chương tiếp theo, bạn sẽ tìm hiểu thêm về các thuật toán học máy khác nhau và cách sử dụng chúng để giải quyết các vấn đề trong cuộc sống thực.