CHƯƠNG 6

Học tập có giám sát - Tuyến tính Lại gression

Các loại hồi quy tuyến tính

Trong chương trước, bạn đã học cách bắt đầu với học máy bằng cách sử dụng hồi quy tuyến tính đơn giản, đầu tiên sử dụng Python, sau đó sử dụng thư viện Scikit-learning. Trong chương này, chúng ta sẽ xem xét chi tiết hơn về hồi quy tuyến tính và thảo luận về một biến thể khác của hồi quy tuyến tính được gọi là *hồi quy đa thức*.

Tóm lại, Hình 6.1 cho thấy tập dữ liệu Iris được sử dụng trong Chương 5, "Bắt đầu với Scikit-learning cho Máy học". Bốn cột đầu tiên được gọi là *Tính năng, đặc điểm*, hoặc cũng thường được gọi là *biến độc lập*. Cột cuối cùng được gọi là *nhãn mác*, hoặc thường được gọi là *biến phụ thuộc* (hoặc *biến phụ thuộc* nếu có nhiều hơn một nhãn).

Đặc trưng (Biến độc lập)				Nhãn mác (Biến phụ thuộc)
chiều dài đài hoa	chiều rộng đài hoa	chiều dài cánh hoa	chiều rộng cánh hoa	Mục tiêu

Hình 6.1: Một số thuật ngữ cho các tính năng và nhãn

Học máy Python®, Ấn bản đầu tiên. Wei-Meng Lee.

TIỀN BOA **Các tính năng đôi khi cũng được gọi là *biến giải thích*, trong khi nhãn đôi khi cũng được gọi là *mục tiêu*.**

Trong hồi quy tuyến tính đơn giản, chúng ta đã nói về mối quan hệ tuyến tính giữa một biến độc lập và một biến phụ thuộc. Trong chương này, bên cạnh hồi quy tuyến tính đơn giản, chúng ta cũng sẽ thảo luận về những điều sau:

Nhiều hồi quyMối quan hệ tuyến tính giữa hai hoặc nhiều độc lập biến và một biến phụ thuộc.

Hồi quy đa thứcMô hình hóa mối quan hệ giữa một bên độc lập biến và một biến phụ thuộc bằng cách sử dụng$N_{\text{thứ tự}}$hàm đa thức bậc.

Hồi quy bội đa thứcMô hình hóa mối quan hệ giữa hai hoặc nhiều biến độc lập và một biến phụ thuộc sử dụng$N_{\text{thứ tự}}$hàm đa thức bậc.

TIỀN BOA **Có một dạng hồi quy tuyến tính khác, được gọi là *regres tuyến tính đa biến-sion*, trong đó có nhiều hơn một biến phụ thuộc tương quan trong mối quan hệ. Hồi quy tuyến tính đa biến nằm ngoài phạm vi của cuốn sách này.**

Hồi quy tuyến tính

Trong học máy,*hồi quy tuyến tính*là một trong những thuật toán đơn giản nhất mà bạn có thể áp dụng cho tập dữ liệu để mô hình hóa mối quan hệ giữa các đối tượng địa lý và nhãn. Trong Chương 5, chúng ta đã bắt đầu bằng cách khám phá hồi quy tuyến tính đơn giản, nơi chúng ta có thể giải thích mối quan hệ giữa một đối tượng địa lý và nhãn bằng cách sử dụng một đường thẳng. Trong phần sau, bạn sẽ tìm hiểu về một biến thể của hồi quy tuyến tính đơn giản, được gọi là*hồi quy nhiều tuyến tính*, bằng cách dự đoán giá nhà dựa trên nhiều tính năng.

Sử dụng Tập dữ liệu Boston

Đối với ví dụ này, chúng tôi sẽ sử dụng tập dữ liệu Boston, tập dữ liệu này chứa dữ liệu về nhà ở và dữ liệu giá cả ở khu vực Boston. Bộ dữ liệu này được lấy từ thư viện StatLib, được duy trì tại Đại học Carnegie Mellon. Nó thường được sử dụng trong học máy và nó là một ứng cử viên tốt để tìm hiểu về các vấn đề hồi quy. Tập dữ liệu Boston có sẵn từ một số nguồn, nhưng hiện có sẵn trực tiếp từ`sklearn.datasets`bưu kiện. Điều này có nghĩa là bạn có thể tải nó trực tiếp trong Scikit-learning mà không cần tải xuống một cách rõ ràng.

Đầu tiên, hãy nhập các thư viện cần thiết và sau đó tải tập dữ liệu bằng cách sử dụng load_boston ()hàm số:

```
import matplotlib.pyplot as plt import
pandas as pd
nhập numpy dưới dạng np

từ sklearn.datasets import load_boston dataset =
load_boston ()
```

Việc kiểm tra dữ liệu trước khi bạn làm việc với nó luôn luôn tốt. Cácdữ liệu thuộc tính chứa dữ liệu cho các cột khác nhau của tập dữ liệu:

```
print (dataset.data)
```

Bạn sẽ thấy những điều sau:

```
[[     6.32000000e-03      1.80000000e + 01      2.31000000e + 00 ...,      1.53000000e + 01
       3.96900000e + 02       4,98000000e + 00]
 [     2.73100000e-02      0,00000000e + 00      7,07000000e + 00 ...,      1.78000000e + 01
       3.96900000e + 02       9.14000000e + 00]
 [     2.72900000e-02      0,00000000e + 00      7,07000000e + 00 ...,      1.78000000e + 01
       3,92830000e + 02       4.03000000e + 00]
 . . . ,
 [     6.07600000e-02      0,00000000e + 00      1,19300000e + 01 ...,      2.10000000e + 01
       3.96900000e + 02       5.64000000e + 00]
 [     1.09590000e-01      0,00000000e + 00      1,19300000e + 01 ...,      2.10000000e + 01
       3,93450000e + 02       6.48000000e + 00]
 [     4,74100000e-02      0,00000000e + 00      1,19300000e + 01 ...,      2.10000000e + 01
       3.96900000e + 02       7.88000000e + 00]]
```

Dữ liệu là một mảng hai chiều. Để biết tên của từng cột (tính năng), hãy sử dụngfeature_namestài sản:

```
print (dataset.feature_names)
```

Bạn sẽ thấy những điều sau:

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'TUỔI' 'DIS' 'RAD' 'THUẾ' 'PTRATIO'
  'B' 'LSTAT']
```

Để biết mô tả của từng tính năng, bạn có thể sử dụngMÔ TẢtài sản:

```
in (tập dữ liệu.DESCR)
```

Câu lệnh trước sẽ in ra như sau:

```
Bộ dữ liệu Giá nhà ở Boston
==========================

Ghi chú
------
```

Đặc điểm của Tập dữ liệu:

: Số phiên bản: 506

: Số thuộc tính: 13 dự đoán số / phân loại

: Giá trị Trung bình (thuộc tính 14) thường là mục tiêu

: Thông tin thuộc tính (theo thứ tự):

- TỘI LỖI tỷ lệ tội phạm bình quân đầu người theo thị trấn
- ZN tỷ lệ đất ở được phân lô trên 25.000 sq.ft.
- INDUS tỷ lệ mẫu đất kinh doanh không bán lẻ trên mỗi thị trấn Biến giả
- CHAS sông Charles (= 1 nếu đường giới hạn con sông; 0 nếu không)
- NOX nồng độ nitric oxit (phần trên 10 triệu) số phòng trung bình trên
- RM một ngôi nhà
- TUỔI TÁC tỷ lệ các đơn vị có chủ sở hữu được xây dựng trước năm 1940 có trọng số
- DIS khoảng cách đến năm trung tâm việc làm Boston chỉ số về khả năng tiếp
- RAD cận các đường cao tốc xuyên tâm
- THUẾ thuế suất tài sản-giá trị đầy đủ cho mỗi tỷ lệ học sinh-
- PTRATIO giáo viên $ 10.000 theo thị trấn
- B 1000 (Bk - 0,63) ^ 2 trong đó Bk là tỷ lệ của người da đen ở thị trấn
- THƯỜNG THỨC % tình trạng thấp hơn của dân số
- MEDV Giá trị trung bình của những ngôi nhà có chủ sở hữu tính bằng $ 1000

: Thiếu giá trị thuộc tính: Không có

: Người tạo: Harrison, D. và Rubinfeld, DL

Đây là bản sao của tập dữ liệu nhà ở UCI ML: http://archive.ics.uci.edu/ml / datasets / Nhà ở

Bộ dữ liệu này được lấy từ thư viện StatLib được duy trì tại Đại học Carnegie Mellon.

Dữ liệu về giá nhà ở Boston của Harrison, D. và Rubinfeld, DL 'Giá Hedonic và nhu cầu về không khí sạch', J. Environ. Kinh tế & Quản lý, tập 5, 81-102, 1978.
Được sử dụng ở Belsley, Kuh & Welsch, 'Regression chẩn đoán
. . . ', Wiley, 1980. NB Các phép biến đổi khác nhau được sử dụng trong bảng trên trang 244-261 của phần sau.

Dữ liệu giá nhà ở Boston đã được sử dụng trong nhiều bài báo về máy học để giải quyết hồi quy
các vấn đề.

* * Người giới thiệu**

- Belsley, Kuh & Welsch, 'Chẩn đoán hồi quy: Xác định
Dữ liệu có ảnh hưởng và các nguồn thông tin cộng gộp ', Wiley, 1980. 244-261.
- Quinlan, R. (1993). Kết hợp dựa trên phiên bản và dựa trên mô hình
Học tập. Trong Kỷ yếu Hội nghị Quốc tế lần thứ mười về Học máy, 236-243, Đại học
Massachusetts, Amherst. Morgan Kaufmann.

- nhiều hơn nữa! (xem http://archive.ics.uci.edu/ml/datasets/
Nhà ở)

Giá nhà là thông tin mà chúng tôi đang tìm kiếm và có thể được truy cập thông quaMục tiêutài sản:

```
print (dataset.target)
```

Bạn sẽ thấy như sau:

```
[24.        21,6 34,7 33,4 36,2 28,7 22,9 27,1 16,5 18,9 15. 18,9
   21,7 20,4 18,2 19,9 23,1 17,5 20,2 18,2 13,6 19,6 15,2 14,5
   15,6 13,9 16,6 14,8 18,4 21. 12,7 14,5 13,2 13,1 13,5 18,9
   20. 21. 24,7 30,8 34,9 26,6 25,3 24,7 21,2 19,3 20. 16,6
   14,4 19,4 19,7 20,5 25. 23,4 18,9 35,4 24,7 31,6 23,3 19,6
   18,7 16. 22,2 25. 33. 23,5 19,4 22. 17,4 20,9 24,2 21,7
   22,8 23,4 24,1 21,4 20. 20,8 21,2 20,3 28. 23,9 24,8 22,9
   23,9 26,6 22,5 22,2 23,6 28,7 22,6 22. 22,9 25. 20,6 28,4
   21,4 38,7 43,8 33,2 27,5 26,5 18,6 19,3 20,1 19,5 19,5 20,4
   19,8 19,4 21,7 22,8 18,8 18,7 18,5 18,3 21,2 19,2 20,4 19,3
   22. 20,3 20,5 17,3 18,8 21,4 15,7 16,2 18.      14,3 19,2 19,6
   23. 18,4 15,6 18,1 17,4 17,1 13,3 17,8 14.      14,4 13,4 15,6
   11,8 13,8 15,6 14,6 17,8 15,4 21,5 19,6 15,3 19,4 17.      15,6
   13,1 41,3 24,3 23,3 27. 50. 50. 50. 22,7 25. 50. 23,8
   23,8 22,3 17,4 19,1 23,1 23,6 22,6 29,4 23,2 24,6 29,9 37,2
   39,8 36,2 37,9 32,5 26,4 29,6 50. 32. 29,8 34,9 37. 30,5
   36,4 31,1 29,1 50. 33,3 30,3 34,6 34,9 32,9 24,1 42,3 48,5
   50. 22,6 24,4 22,5 24,4 20. 21,7 19,3 22,4 28,1 23,7 25.
   23,3 28,7 21,5 23. 26,7 21,7 27,5 30,1 44,8 50. 37,6 31,6
   46,7 31,5 24,3 31,7 41,7 48,3 29. 24. 25,1 31,5 23,7 23,3
   22. 20,1 22,2 23,7 17,6 18,5 24,3 20,5 24,5 26,2 24,4 24,8
   29,6 42,8 21,9 20,9 44. 50. 36. 30,1 33,8 43,1 48,8 31.
   36,5 22,8 30,7 50. 43,5 20,7 21,1 25,2 24,4 35,2 32,4 32.
   33,2 33,1 29,1 35,1 45,4 35,4 46. 50. 32,2 22. 20,1 23,2
   22,3 24,8 28,5 37,3 27,9 23,9 21,7 28,6 27,1 20,3 22,5 29.
   24,8 22. 26,4 33,1 36,1 28,4 33,4 28,2 22,8 20,3 16,1 22,1
   19,4 21,6 23,8 16,2 17,8 19,8 23,1 21. 23,8 23,1 20,4 18,5
   25. 24,6 23. 22,2 19,3 22,6 19,8 17,1 19,4 22,2 20,7 21,1
   19,5 18,5 20,6 19. 18,7 32,7 16,5 23,9 31,2 17,5 17,2 23,1
   24,5 26,6 22,9 24,1 18,6 30,1 18,2 20,6 17,8 21,7 22,7 22,6
   25. 19,9 20,8 16,8 21,9 27,5 21,9 23,1 50. 50. 50. 50.
   50. 13,8 13,8 15. 13,9 13,3 13,1 10,2 10,4 10,9 11,3 12,3
    8,8 7,2 10,5 7,4 10,2 11,5 15,1 23,2 9,7 13,8 12,7 13,1
   12,5 8,5 5. 6,3 5,6 7,2 12,1 8,3 8,5 5. 11,9 27,9
   17,2 27,5 15. 17,2 17,9 16,3 7. 7,2 7,5 10,4 8,8 8,4
```

```
16,7 14,2 20,8 13,4 11,7 8,3 10,2 10,9 11.                    9,5 14,5 14,1
16,1 14,3 11,7 13,4 9,6 8,7 8,4 12,8 10,5 17,1 18,4 15,4
10,8 11,8 14,9 12,6 14,1 13. 13,4 15,2 16,1 17,8 14,9 14,1
12,7 13,5 14,9 20. 16,4 17,7 19,5 20,2 21,4 19,9 19. 19,1
19,1 20,1 19,9 19,6 23,2 29,8 13,8 13,3 16,7 12. 14,6 21,4
23. 23,7 25. 21,8 20,6 21,2 19,1 20,6 15,2 7. 8,1 13,6
20,1 21,8 24,5 23,1 19,7 18,3 21,2 17,5 16,8 22,4 20,6 23,9
22. 11,9]
```

Bây giờ hãy tải dữ liệu vào Pandas DataFrame:

```
df = pd.DataFrame (dataset.data, column = dataset.feature_names) df.head ()
```

DataFrame sẽ giống như trong hình 6.2.

	TỘI LỖI	ZN	INDUS	CHAS	NOX	RM	TUỔI	DIS	RAD	THUẾ	PTRATIO	B	LSTAT
0	0,00632	18,0	2,31	0,0	0,538	6,575	65,2	4,0900	1,0	296,0	15,3	396,90	4,98
1	0,02731	0,0	7,07	0,0	0,469	6,421	78,9	4,9671	2.0	242.0	17,8	396,90	9.14
2	0,02729	0,0	7,07	0,0	0,469	7,185	61,1	4,9671	2,0	242,0	17,8	392,83	4,03
3	0,03237	0,0	2,18	0,0	0,458	6,998	45,8	6,0622	3.0	222.0	18,7	394,63	2,94
4	0,06905	0,0	2,18	0,0	0,458	7,147	54,2	6,0622	3,0	222,0	18,7	396,90	5,33

Hình 6.2: DataFrame chứa tất cả các tính năng

Bạn cũng muốn thêm giá của các ngôi nhà vào DataFrame, vì vậy hãy thêm một cột mới vào DataFrame và gọi nó là MEDV:

```
df ['MEDV'] = dataset.target df.head
()
```

Hình 6.3 cho thấy DataFrame hoàn chỉnh với các tính năng và nhãn.

	TỘI LỖI	ZN	INDUS	CHAS	NOX	RM	TUỔI	DIS	RAD	THUẾ	PTRATIO	B	LSTAT	MEDV
0	0,00632	18,0	2,31	0,0	0,538	6,575	65,2	4,0900	1,0	296,0	15,3	396,90	4,98	24,0
1	0,02731	0,0	7,07	0,0	0,469	6,421	78,9	4,9671	2.0	242.0	17,8	396,90	9,14	21,6
2	0,02729	0,0	7,07	0,0	0,469	7,185	61,1	4,9671	2,0	242,0	17,8	392,83	4,03	34,7
3	0,03237	0,0	2,18	0,0	0,458	6,998	45,8	6,0622	3.0	222.0	18,7	394,63	2,94	33,4
4	0,06905	0,0	2,18	0,0	0,458	7,147	54,2	6,0622	3,0	222,0	18,7	396,90	5,33	36,2

Hình 6.3: DataFrame chứa tất cả các tính năng và nhãn

Dọn dẹp dữ liệu

Bước tiếp theo sẽ là làm sạch dữ liệu và thực hiện bất kỳ chuyển đổi nào nếu cần. Đầu tiên, sử dụngthông tin()chức năng kiểm tra kiểu dữ liệu của từng trường:

```
df.info ()
```

Bạn sẽ thấy những điều sau:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 mục nhập, 0 đến 505 cột Dữ liệu
(tổng số 14 cột): CRIM
                506 float64 không null
ZN              506 float64 không null
INDUS           506 float64 không null
CHAS            506 float64 không null
NOX             506 float64 không null
RM              506 float64 không null
TUỔI TÁC        506 float64 không null
DIS             506 float64 không null
RAD             506 float64 không null
THUẾ            506 float64 không null
PTRATIO         506 float64 không null
B               506 float64 không null
LSTAT           506 float64 không null
MEDV            506 float64 không null
các loại: float64 (14)
sử dụng bộ nhớ: 55,4 KB
```

Vì Scikit-learning chỉ hoạt động với các trường là số, bạn cần mã hóa các giá trị chuỗi thành các giá trị số. May mắn thay, tập dữ liệu chứa tất cả các giá trị số và vì vậy không cần mã hóa.

Tiếp theo, chúng ta cần kiểm tra xem có thiếu giá trị nào không. Để làm như vậy, hãy sử dụngisnull ()hàm số:

```
print (df.isnull (). sum ())
```

Một lần nữa, tập dữ liệu tốt, vì nó không có bất kỳ giá trị nào bị thiếu:

```
TỘI LỖI         0
ZN              0
INDUS           0
CHAS            0
NOX             0
RM              0
TUỔI TÁC        0
DIS             0
RAD             0
THUẾ            0
PTRATIO         0
B               0
```

```
LSTAT    0
MEDV     0
dtype: int64
```

Lựa chọn tính năng

Bây giờ dữ liệu đã tốt, chúng tôi đã sẵn sàng chuyển sang bước tiếp theo của quy trình. Vì có 13 tính năng trong tập dữ liệu, chúng tôi không muốn sử dụng tất cả các tính năng này để đào tạo mô hình của mình, bởi vì không phải tất cả chúng đều phù hợp. Thay vào đó, chúng tôi muốn chọn những đặc điểm ảnh hưởng trực tiếp đến kết quả (tức là giá nhà) để đào tạo mô hình. Đối với điều này, chúng ta có thể sử dụng`corr ()`hàm số. Các`corr ()`hàm tính toán mối tương quan theo cặp của các cột:

```
corr = df.corr ()
in (corr)
```

Bạn sẽ thấy như sau:

```
TỘI LỖI      ZN       INDUS      CHAS       NOX        RM      TUỔI TÁC \
TỘI LỖI   1.000000 -0.199458  0,404471 -0,055295  0,417521 -0,219940
0,350784
ZN       - 0,199458  1.000000 -0.533828 -0.042697 -0.516604  0,311991
- 0,569537
INDUS     0,404471 -0,533828  1,000000  0,062938  0,763651 -0,391676
0,644779
CHAS     - 0,055295 -0,042697  0,062938  1,000000  0,091203  0,091251
0,086518
NOX       0,417521 -0,516604  0,763651  0,091203  1,000000 -0,302188
0,731470
RM       - 0,219940  0,311991 -0,391676  0,091251 -0,302188  1.000000
- 0,240265
TUỔI TÁC  0,350784 -0,569537  0,644779  0,086518  0,731470 -0,240265
1.000000
DIS      - 0,377904  0,664408 -0,708027 -0,099176 -0,769230  0,205246
- 0,747881
RAD       0,622029 -0,311948  0,595129 -0,007368  0,611441 -0,209847
0,456022
THUẾ      0,579564 -0,314563  0,720760 -0,035587  0,668023 -0,292048
0,506456
PTRATIO   0,288250 -0,391679  0,383248 -0,121515  0,188933 -0,355501
0,261515
B        - 0,377365  0,175520 -0,356977  0,048788 -0,380051  0,128069
- 0,273534
LSTAT     0,452220 -0,412995  0,603800 -0,053929  0,590879 -0,613808
0,602339
MEDV     - 0,385832  0,360445 -0,483725  0,175260 -0,427321  0,695360
- 0,376955
```

	DIS	RAD	THUẾ	PTRATIO	B	LSTAT	MEDV
TỘI LỖI	- 0,377904	0,622029	0,579564	0,288250	-0,377365	0,452220	- 0,385832
ZN	0,664408	-0,311948	-0,314563	-0,391679	0,175520	-0,412995	0,360445
INDUS	- 0,708027	0,595129	0,720760	0,383248	-0,356977	0,603800	- 0,483725
CHAS	- 0,099176	-0,007368	-0,035587	-0,121515	0,048788	-0,053929	0,175260
NOX	- 0,769230	0,611441	0,668023	0,188933	-0,380051	0,590879	- 0,427321
RM	0,205246	-0,209847	-0,292048	-0,355501	0,128069	-0,613808	0,695360
TUỔI TÁC	- 0,747881	0,456022	0,506456	0,261515	-0,273534	0,602339	- 0,376955
DIS	1,000000	-0,494588	-0,534432	-0,232471	0,291512	-0,496996	0,249929
RAD	- 0,494588	1,000000	0,910228	0,464741	-0,444413	0,488676	- 0,381626
THUẾ	- 0,534432	0,910228	1,000000	0,460853	-0,441808	0,543993	- 0,468536
PTRATIO	-0,232471	0,464741	0,460853	1,000000	-0,177383	0,374044	- 0,507787
B	0,291512	-0,444413	-0,441808	-0,177383	1,000000	-0,366087	0,333461
LSTAT	- 0,496996	0,488676	0,543993	0,374044	-0,366087	1,000000	- 0,737663
MEDV	0,249929	-0,381626	-0,468536	-0,507787	0,333461	-0,737663	1.000000

Một*tương quan tích cực*là mối quan hệ giữa hai biến trong đó cả hai biến chuyển động song song với nhau. Mối tương quan thuận tồn tại khi một biến giảm khi biến kia giảm hoặc một biến tăng trong khi biến kia tăng. Tương tự, một*tương quan tiêu cực*là mối quan hệ giữa hai biến, trong đó một biến tăng khi biến kia giảm. Mối tương quan âm hoàn hảo được biểu thị bằng giá trị –1,00: 0,00 cho biết không có tương quan và +1,00 cho biết mối tương quan dương hoàn hảo.

TừMEDVtrong đầu ra, bạn có thể thấy rằngRMvàLSTATcác đối tượng địa lý có các yếu tố tương quan cao (tương quan tích cực và tiêu cực) vớiMEDV:

	MEDV
TỘI LỖI	- 0,385832
ZN	0,360445
INDUS	- 0,483725
CHAS	0,175260
NOX	- 0,427321
RM	**0,695360**
TUỔI TÁC	- 0,376955

```
DIS          0,249929
RAD        - 0,381626
THUẾ       - 0,468536
PTRATIO -0,507787
B            0,333461
LSTAT      - 0,737663
MEDV         1.000000
```

Điều này có nghĩa làLSTAT(“%của địa vị thấp hơn của dân số ”) tăng lên, giá nhà đi xuống. Khi nàoLSTATgiảm, giá đi lên. Tương tự, như RM (“số lượng phòng trung bình trên mỗi ngôi nhà ”) tăng, giá cũng vậy. Và khiRMgiảm, giá cũng giảm.

Thay vì tìm kiếm trực quan hai tính năng hàng đầu có các yếu tố tương quan cao nhất, chúng tôi có thể thực hiện theo chương trình như sau:

```
# - - - lấy 3 tính năng hàng đầu có mối tương quan cao nhất --- print (df.corr (). abs ().
nlargest (3, 'MEDV'). index)

# - - - in 3 giá trị tương quan hàng đầu --- print (df.corr (). abs (). nlargest (3,
'MEDV'). giá trị [:, 13])
```

Kết quả xác nhận những phát hiện của chúng tôi:

```
Chỉ mục (['MEDV', 'LSTAT', 'RM'], dtype = 'object') [1.
                        0,73766273 0,69535995]
```

TIỀN BOA **Chúng tôi sẽ bỏ qua kết quả đầu tiên, vì**MEDV**chắc chắn có một mối tương quan hoàn hảo với chính nó!**

TừRMvàLSTATcó giá trị tương quan cao, chúng tôi sẽ sử dụng hai tính năng này để đào tạo mô hình của mình.

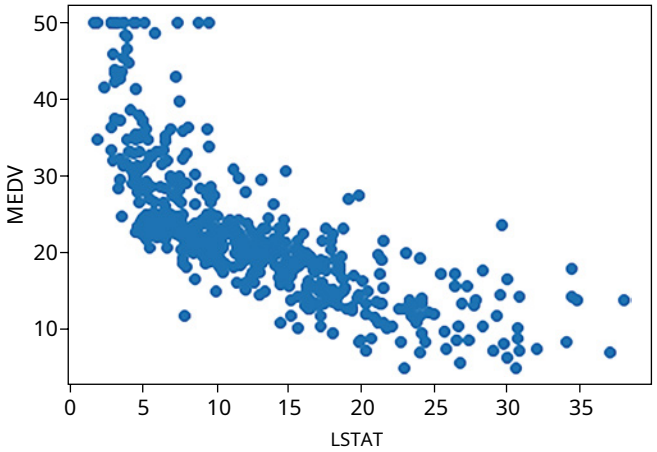
Nhiều hồi quy

Trong chương trước, bạn đã biết cách thực hiện một hồi quy tuyến tính đơn giản bằng cách sử dụng một đối tượng địa lý và một nhãn. Thông thường, bạn có thể muốn đào tạo mô hình của mình bằng cách sử dụng nhiều hơn một biến độc lập và một nhãn. Điều này được gọi là*hồi quy nhiều lần*. Trong hồi quy bội, hai hoặc nhiều biến độc lập được sử dụng để dự đoán giá trị của một biến phụ thuộc (nhãn).

Bây giờ, hãy vẽ một biểu đồ phân tán cho thấy mối quan hệ giữaLSTATtính năng vàMEDVnhãn mác:

```
plt.scatter (df ['LSTAT'], df ['MEDV'], marker = 'o') plt.xlabel
('LSTAT')
plt.ylabel ('MEDV')
```

Hình 6.4 cho thấy biểu đồ phân tán. Dường như có một mối tương quan tuyến tính giữa hai yếu tố này.

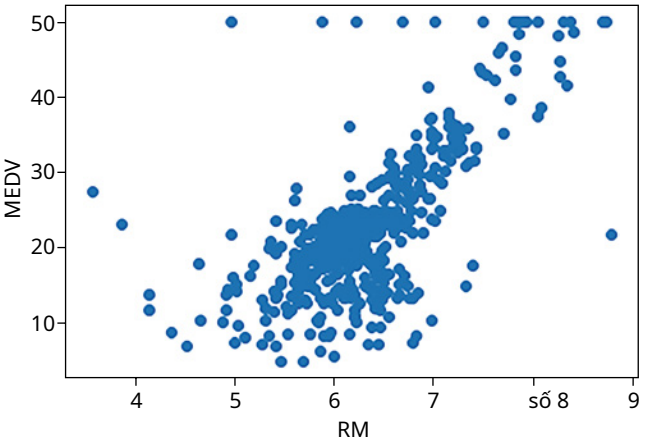


Hình 6.4: Biểu đồ phân tán cho thấy mối quan hệ giữa LSTAT và MEDV

Hãy cũng vẽ một biểu đồ phân tán cho thấy mối quan hệ giữaRMtính năng và MEDVnhãn mác:

```
plt.scatter (df ['RM'], df ['MEDV'], marker = 'o') plt.xlabel ('RM')

plt.ylabel ('MEDV')
```

Hình 6.5 cho thấy biểu đồ phân tán. Một lần nữa, dường như có một mối tương quan tuyến tính giữa hai yếu tố này, mặc dù có một số ngoại lệ.

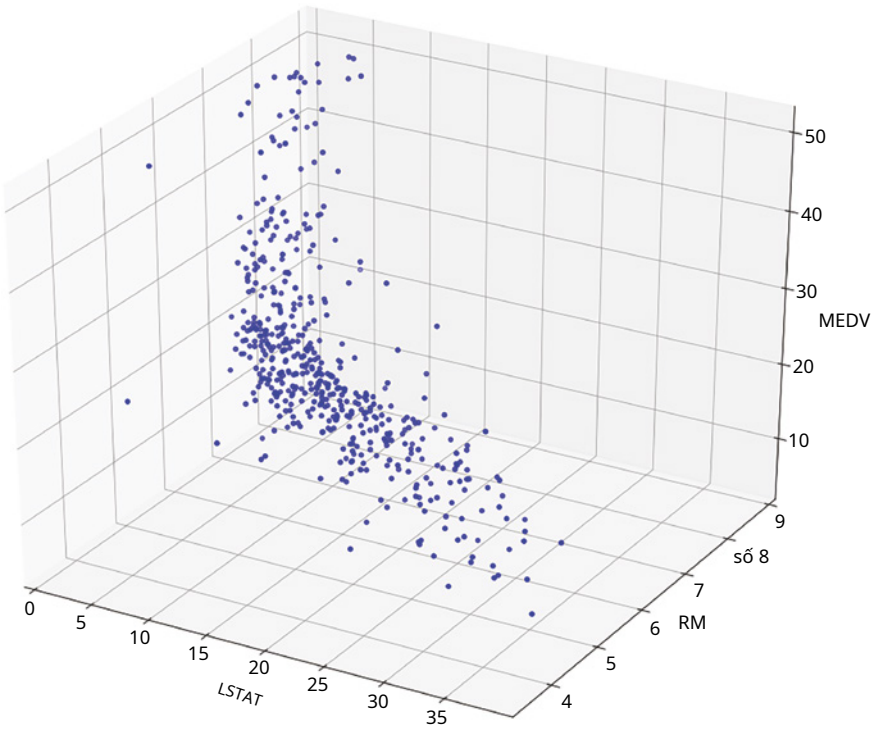


Hình 6.5: Biểu đồ phân tán cho thấy mối quan hệ giữa RM và MEDV

Vẫn tốt hơn, hãy vẽ biểu đồ của hai tính năng và nhãn trên biểu đồ 3D:

```
từ mpl_toolkits.mplot3d nhập Axes3D

fig = plt.figure (figsize = (18,15))
ax = fig.add_subplot (111, projector = '3d')

ax.scatter (df ['LSTAT'],
                df ['RM'],
                df ['MEDV'],
                c = 'b')

ax.set_xlabel ("LSTAT")
ax.set_ylabel ("RM")
ax.set_zlabel ("MEDV")
plt.show ()
```

Hình 6.6 cho thấy biểu đồ 3D củaLSTATvàRMÂm mưu chống lạiMEDV.



Hình 6.6: Biểu đồ phân tán 3D cho thấy mối quan hệ giữa LSTAT, RM và MEDV

Đào tạo người mẫu

Bây giờ chúng tôi có thể đào tạo mô hình. Đầu tiên, tạo hai DataFrames:xvàY.Cácx DataFrame sẽ chứa sự kết hợp củaLSTATvàRMcác tính năng, trong khiY DataFrame sẽ chứaMEDVnhãn mác:

```
x = pd.DataFrame (np.c_ [df ['LSTAT'], df ['RM']], cột = ['LSTAT', 'RM']) Y = df ['MEDV']
```

Chúng tôi sẽ chia tập dữ liệu thành 70 phần trăm để đào tạo và 30 phần trăm để thử nghiệm:

```
từ sklearn.model_selection nhập train_test_split
x_train, x_test, Y_train, Y_test = train_test_split (x, Y, test_size = 0,3,
                                                     random_state = 5)
```

TIỀN BOA **Chương 7, "Học có giám sát — Phân loại sử dụng hồi quy logistic," sẽ thảo luận thêm về**xe lửa_kiểm tra_tách ra()**hàm số.**

Sau khi tách, chúng ta hãy in ra hình dạng của các bộ huấn luyện:

```
print (x_train.shape)
print (Y_train.shape)
```

Bạn sẽ thấy như sau:

```
(354, 2)
(354,)
```

Điều này có nghĩa làxtập huấn luyện hiện có 354 hàng và 2 cột, trong khiY tập huấn luyện (chứa nhãn) có 354 hàng và 1 cột.

Hãy cũng in ra bộ thử nghiệm:

```
print (x_test.shape)
print (Y_test.shape)
```

Lần này, tập hợp thử nghiệm có 152 hàng:

```
(152, 2)
(152,)
```

Bây giờ chúng tôi đã sẵn sàng để bắt đầu đào tạo. Như bạn đã học từ chương trước, bạn có thể sử dụngTuyến tínhlớp để thực hiện hồi quy tuyến tính. Trong trường hợp này, chúng tôi sẽ sử dụng nó để đào tạo mô hình của chúng tôi:

```
từ sklearn.linear_model import LinearRegression

model = LinearRegression ()
model.fit (x_train, Y_train)
```

Sau khi mô hình được đào tạo, chúng tôi sẽ sử dụng bộ thử nghiệm để thực hiện một số dự đoán:

```
price_pred = model.p Dự đoán (x_test)
```

Để tìm hiểu mô hình của chúng tôi hoạt động tốt như thế nào, chúng tôi sử dụng phương pháp R-Squared mà bạn đã học trong chương trước. Phương pháp R-Squared cho phép bạn biết mức độ gần của dữ liệu thử nghiệm phù hợp với đường hồi quy. Giá trị 1,0 có nghĩa là một sự phù hợp hoàn hảo. Vì vậy, bạn nhắm đến giá trị của R-Squared gần bằng 1:

```
print ('R-Bình phương:% .4f'% model.score (x_test,
                                                    Y_test))
```

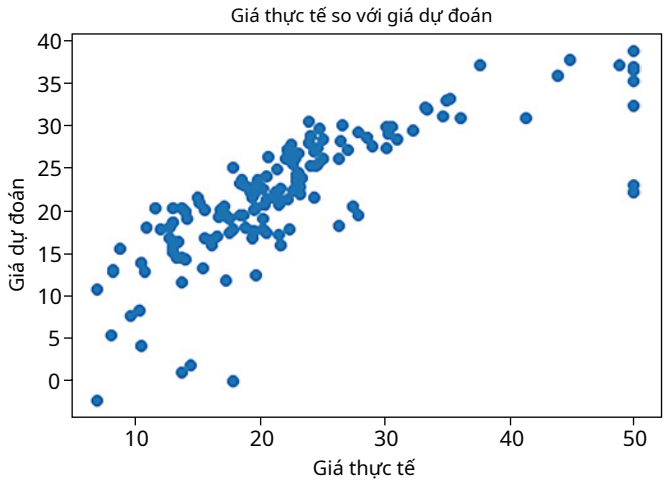
Đối với mô hình của chúng tôi, nó trả về giá trị R-Squared như sau:

```
Bình phương R: 0,6162
```

Chúng tôi cũng sẽ vẽ một biểu đồ phân tán hiển thị giá thực tế so với giá dự đoán:

```
từ sklearn.metrics nhập mean_squared_error

mse = mean_squared_error (Y_test, price_pred) print (mse)


plt.scatter (Y_test, price_pred) plt.xlabel ("Giá thực tế")
plt.ylabel ("Giá dự đoán") plt.title ("Giá thực tế so với giá dự
đoán")
```

Hình 6.7 cho thấy âm mưu. Lý tưởng nhất, nó nên là một đường thẳng, nhưng bây giờ nó là đủ tốt.



Hình 6.7: Biểu đồ phân tán hiển thị giá dự đoán so với giá thực tế

Nhận điểm đánh chặn và hệ số

Công thức cho hồi quy bội như sau:

$$= \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

trong đó Y là biến phụ thuộc, $\beta_0$là điểm chặn, và $\beta_1$và $\beta_2$là hệ số của hai đối tượng $x_1$và x_2, tương ứng.

Với mô hình được đào tạo, chúng ta có thể nhận được hệ số chặn cũng như hệ số của các tính năng:

```
print (model.intercept_) print
(model.coef_)
```

Bạn sẽ thấy những điều sau:

```
0,3843793678034899
[-0,65957972 4,83197581]
```

Chúng ta có thể sử dụng mô hình để đưa ra dự đoán giá nhà khiLSTAT 30 tuổi vàRMlà 5:

```
print (model.p dự đoán ([[30,5]]))
```

Bạn sẽ thấy những điều sau:

```
[4,75686695]
```

Bạn có thể xác minh giá trị dự đoán bằng cách sử dụng công thức được đưa ra trước đó:

$$= \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$= 0,3843793678034899 + 30\ 0(-65957972) + 5(4\ 83197581)$$

$$= 4.7568$$

Vẽ siêu phẳng 3D

Hãy vẽ một siêu phẳng hồi quy 3D hiển thị các dự đoán:

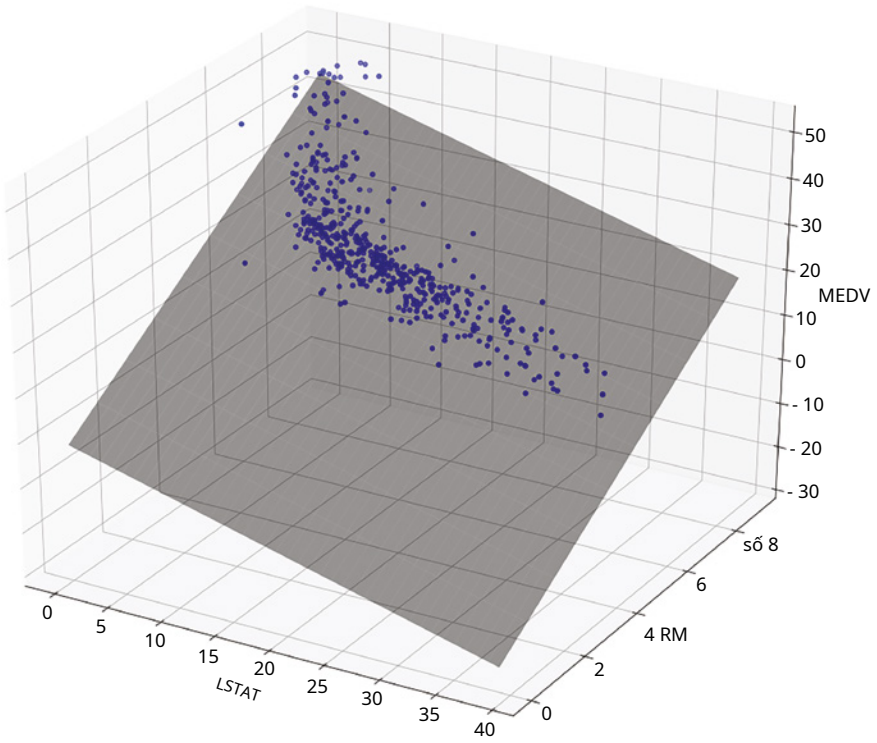
```
import matplotlib.pyplot as plt import
pandas as pd
nhập numpy dưới dạng np
từ mpl_toolkits.mplot3d nhập Axes3D

từ sklearn.datasets import load_boston dataset =
load_boston ()

df = pd.DataFrame (dataset.data, column = dataset.feature_names) df ['MEDV'] =
dataset.target
```

```
x = pd.DataFrame (np.c_ [df ['LSTAT'], df ['RM']], cột = ['LSTAT', 'RM']) Y = df ['MEDV']


fig = plt.figure (figsize = (18,15))
ax = fig.add_subplot (111, projector = '3d')

ax.scatter (x ['LSTAT'],
                    x ['RM'],
                    Y,
                    c = 'b')

ax.set_xlabel ("LSTAT")
ax.set_ylabel ("RM")
ax.set_zlabel ("MEDV")

# - - - tạo lưới lưới của tất cả các giá trị cho LSTAT và RM --- x_surf = np.arange (0,
40, 1)                                          # - - - cho LSTAT ---
y_surf = np.arange (0, 10, 1)                   # - - - cho RM ---
x_surf, y_surf = np.meshgrid (x_surf, y_surf)

from sklearn.linear_model import LinearRegression model =
LinearRegression ()
model.fit (x, Y)

# - - - tính z (MEDC) dựa trên mô hình ---
z = lambda x, y: (model.intercept_ + model.coef_ [0] * x + model.coef_ [1] * y)

ax.plot_surface (x_surf, y_surf, z (x_surf, y_surf),
                            rstride = 1,
                            cstride = 1,
                            color = 'Không có',
                            alpha = 0,4)

plt.show ()
```

Ở đây, chúng tôi đang đào tạo mô hình bằng cách sử dụng toàn bộ tập dữ liệu. Sau đó, chúng tôi đưa ra dự đoán bằng cách chuyển một tổ hợp các giá trị choLSTAT (x _ lướt)vàRM (y_surf) và tính toán các giá trị dự đoán bằng cách sử dụng hệ số và hệ số chặn của mô hình. Siêu phẳng sau đó được vẽ bằng cách sử dụngplot_surface ()hàm số. Kết quả cuối cùng được thể hiện trong Hình 6.8.

Vì biểu đồ được hiển thị trong Máy tính xách tay Jupyter là tĩnh, hãy lưu đoạn mã trước đó trong tệp có tênboston.pyvà chạy nó trong Terminal, như thế này:

```
$ python boston.py
```



Hình 6.8: Siêu phẳng hiển thị các dự đoán cho hai tính năng — LSTAT và RM

Bây giờ bạn sẽ có thể xoay biểu đồ và di chuyển nó xung quanh để có góc nhìn tốt hơn, như trong Hình 6.9.

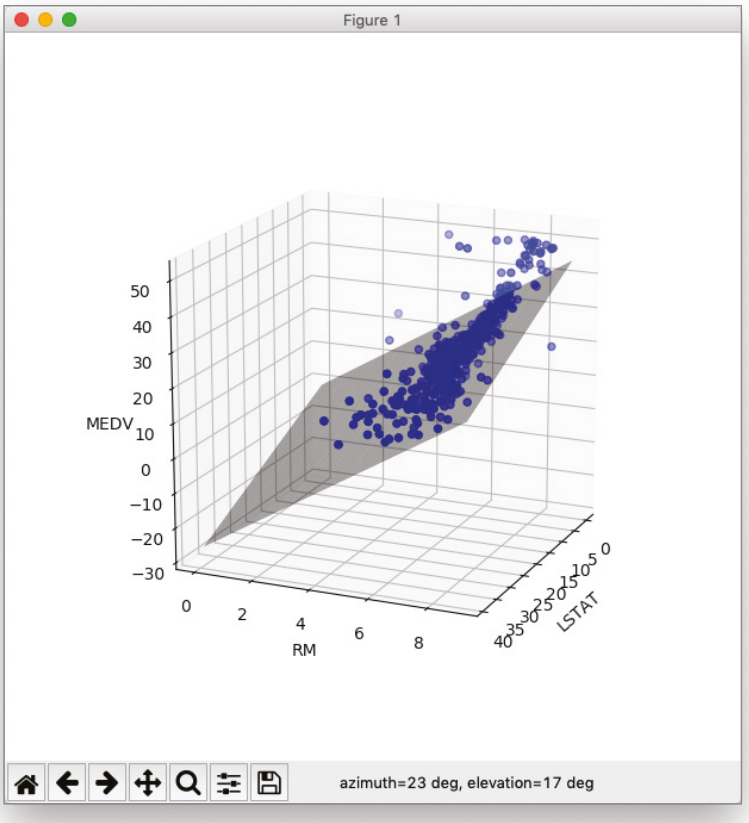
Hồi quy đa thức

Trong phần trước, bạn đã biết cách áp dụng hồi quy tuyến tính để dự đoán giá nhà ở khu vực Boston. Mặc dù kết quả có thể chấp nhận được, nhưng nó không chính xác lắm. Điều này là do đôi khi một đường hồi quy tuyến tính có thể không phải là giải pháp tốt nhất để nắm bắt các mối quan hệ giữa các đối tượng địa lý và nhãn một cách chính xác. Trong một số trường hợp, một đường cong có thể làm tốt hơn.

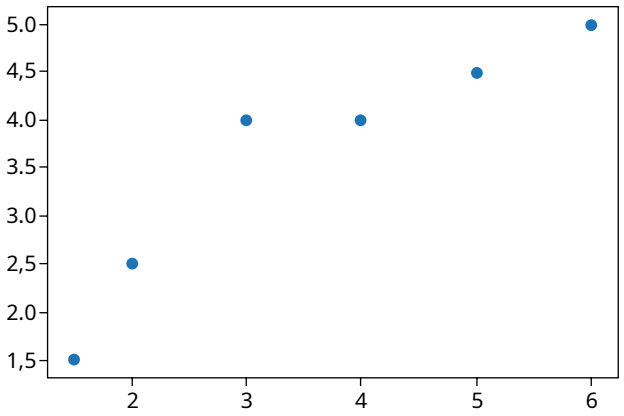
Hãy xem xét một loạt các điểm được thể hiện trong hình 6.10.

Chuỗi điểm được lưu trữ trong một tệp có tênpolynomial.csv:

```
x, y
1,5,1,5
2,2,5
3,4
4,4
5,4,5
6,5
```



Hình 6.9: Xoay biểu đồ để có cái nhìn rõ hơn về siêu phẳng



Hình 6.10: Biểu đồ phân tán của các điểm

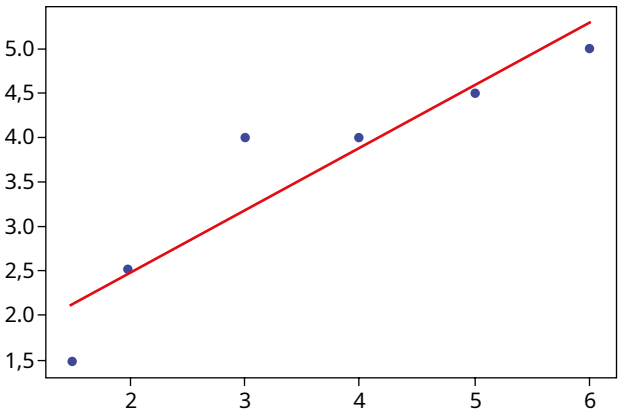
Và được vẽ bằng biểu đồ phân tán:

```
df = pd.read_csv ('polynomial.csv') plt.scatter
(df.x, df.y)
```

Sử dụng hồi quy tuyến tính, bạn có thể cố gắng vẽ một đường thẳng cắt qua hầu hết các điểm:

```
model = LinearRegression ()

x = df.x [0: 6, np.newaxis] # --- chuyển đổi sang mảng 2D --- y = df.y [0:
6, np.newaxis] # --- chuyển đổi sang mảng 2D ---

model.fit (x, y)

# - - - thực hiện dự đoán --- y_pred
= model.p Dự đoán (x)

# - - - vẽ biểu đồ điểm rèn luyện --- plt.scatter
(x, y, s = 10, color = 'b')

# - - - vẽ đường thẳng --- plt.plot (x,
y_pred, color = 'r') plt.show ()


# - - - tính R-Squared ---
print ('R-Bình phương cho tập huấn luyện:% .4f'% model.score (x, y))
```

Bạn sẽ thấy đường hồi quy thẳng, như trong Hình 6.11.



Hình 6.11:Đường hồi quy phù hợp với các điểm

Giá trị R-Squared cho tập huấn luyện là:

```
R-Squared cho tập huấn luyện: 0,8658
```

Chúng tôi muốn xem có cách nào chính xác hơn để phù hợp với các điểm không. Ví dụ, thay vì một đường thẳng, chúng tôi muốn điều tra khả năng của một đường cong. Đây là đâu*hồi quy đa thức*vào đi.

Công thức hồi quy đa thức

*Hồi quy đa thức*là một nỗ lực để tạo ra một hàm đa thức phù hợp với một tập hợp các điểm dữ liệu.

Một hàm đa thức bậc 1 có dạng như sau:

$$= \beta_0 + \beta_1 x$$

Đây là hồi quy tuyến tính đơn giản mà chúng ta đã thấy trong chương trước. *Hồi quy bậc hai*là một đa thức bậc 2:

$$= \beta_0 + \beta_1 x x_2 \beta_2$$

Đối với đa thức bậc 3, công thức như sau:

$$= \beta_0 + \beta_1 x x_2 \beta_2 \quad + \beta_3 x^3$$

Nói chung, một đa thức bậc*N*có công thức là:

$$= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_N x^N$$

Ý tưởng đằng sau hồi quy đa thức rất đơn giản — tìm các hệ số của hàm đa thức phù hợp nhất với dữ liệu.

Hồi quy đa thức trong Scikit-learning

Thư viện Scikit-learning chứa một số lớp và hàm để giải quyết hồi quy đa thức. CácĐa thứclớp nhận vào một số xác định mức độ của các đối tượng đa thức. Trong đoạn mã sau, chúng ta đang tạo một phương trình bậc hai (hàm đa thức bậc 2):

```
from sklearn.preprocessing import PolynomialFeatures level = 2

polynomial_features = Tính năng đa thức (độ = độ)
```

Sử dụng cái nàyĐa thức, bạn có thể tạo một ma trận đối tượng mới bao gồm tất cả các tổ hợp đa thức của các đối tượng có mức độ nhỏ hơn hoặc bằng mức độ đã chỉ định:

```
x_poly = polynomial_features.fit_transform (x) print (x_poly)
```

Bạn sẽ thấy những điều sau:

```
[[1.    1,5 2,25]
 [1.    2. 4.]
 [1.    3. 9.]
 [1.    4. 16.]
 [1.    5. 25.]
 [1.    6. 36.]]
```

Ma trận mà bạn thấy được tạo như sau:

- Cột đầu tiên luôn là 1.
- Cột thứ hai là giá trị của x.
- Cột thứ ba là giá trị của x^2.

Điều này có thể được xác minh bằng cách sử dụngget_feature_names ()hàm số:

```
print (polynomial_features.get_feature_names ('x'))
```

Nó in ra những thứ sau:

```
['1', 'x', 'x ^ 2']
```

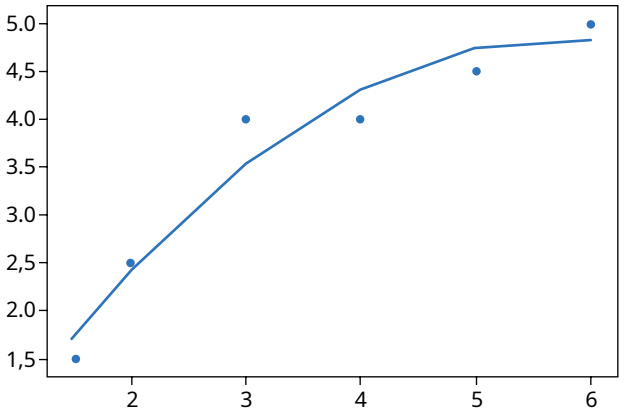
TIỂN BOA **Toán học đằng sau việc tìm kiếm các hệ số của một hàm đa thức nằm ngoài phạm vi của cuốn sách này. Tuy nhiên, đối với những ai quan tâm, hãy xem liên kết sau về phép toán đằng sau hồi quy đa thức:**http: // đa thức. drque.net/math.html**.**

Bây giờ bạn sẽ sử dụng ma trận được tạo này vớiTuyến tínhlớp đào tạo mô hình của bạn:

```
model = LinearRegression ()
model.fit (x_poly, y)
y_poly_pred = model.p Dự đoán (x_poly)

# - - - vẽ biểu đồ các điểm ---
plt.scatter (x, y, s = 10)

# - - - vẽ biểu đồ đường hồi quy --- plt.plot
(x, y_poly_pred)
plt.show ()
```

Hình 6.12 bây giờ cho thấy đường hồi quy, một đường cong đẹp đang cố gắng khớp các điểm. Bạn có thể in ra hệ số chặn và hệ số của hàm đa thức:

```
print (model.intercept_) print
(model.coef_)
```



Hình 6.12:Một đường cong cố gắng vừa với các điểm

Bạn sẽ thấy những điều sau:

```
[-0,87153912]
[[0.                1.98293207 -0.17239897]]
```

Bằng cách cắm những con số $= -0.87153912 + 1.98293207\ x\ + (-0.17239897)x_2$ này vào công thức $= \beta_0 + \beta_1 x + \beta_2 x_2$, bạn có thể đưa ra dự đoán bằng cách sử dụng công thức nhượng bộ.

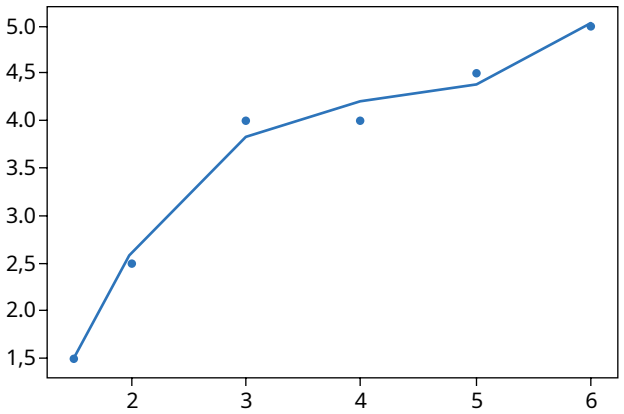
Nếu bạn đánh giá hồi quy bằng cách in giá trị Bình phương R của nó,

```
print ('R-Bình phương cho tập huấn luyện:% .4f'% model.score (x_poly, y))
```

bạn sẽ nhận được giá trị là 0,9474:

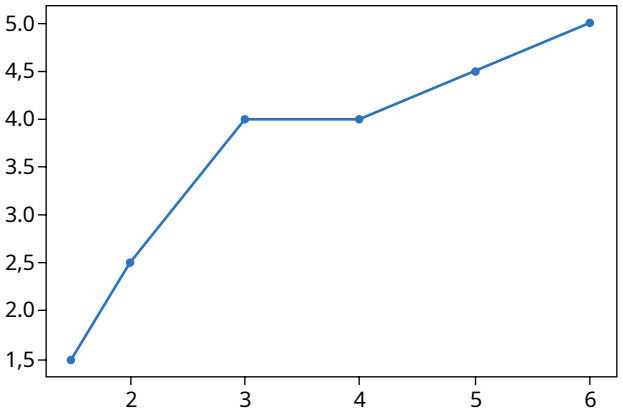
```
R-Squared cho tập huấn luyện: 0.9474
```

Giá trị R-Squared có thể được cải thiện không? Hãy thử một đa thức bậc 3. Sử dụng cùng một mã và thay đổibằng cấpđến3,bạn sẽ nhận được đường cong được hiển thị trong Hình 6.13 và giá trị 0,9889 cho R-Squared.



Hình 6.13:Một đường cong cố gắng vừa với hầu hết các điểm

Bây giờ bạn thấy một đường cong phù hợp hơn với các điểm và giá trị R-Squared được cải thiện nhiều. Hơn nữa, vì việc nâng bậc đa thức lên 1 sẽ cải thiện giá trị R-Squared, bạn có thể muốn tăng nó hơn nữa. Trên thực tế, Hình 6.14 cho thấy đường cong khibằng cấp được đặt thành4.Nó phù hợp với tất cả các điểm một cách hoàn hảo.



Hình 6.14:Dòng bây giờ hoàn toàn phù hợp với các điểm

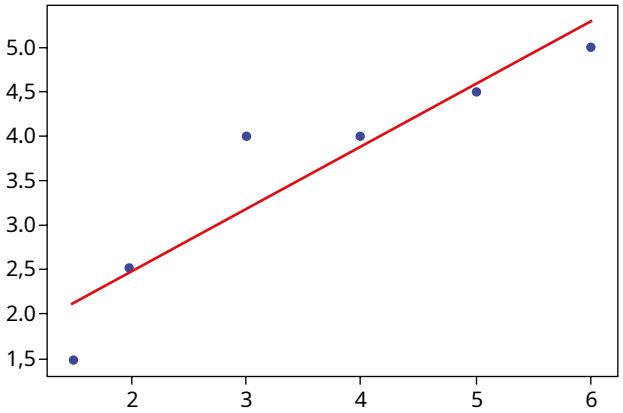
Và đoán xem? Bạn nhận được giá trị R-Squared là 1! Tuy nhiên, trước khi ăn mừng thành công của mình trong việc tìm ra thuật toán hoàn hảo trong dự đoán của mình, bạn cần nhận ra rằng mặc dù thuật toán của bạn có thể phù hợp hoàn hảo với dữ liệu đào tạo, nhưng nó không chắc hoạt động tốt với dữ liệu mới. Đây được gọi là*quá sức*, và phần tiếp theo sẽ thảo luận chi tiết hơn về chủ đề này.

Hiểu được độ chệch và phương sai

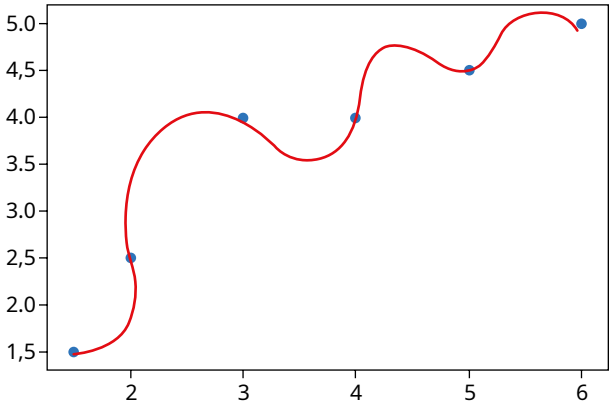
Thuật toán học máy không thể nắm bắt được mối quan hệ thực sự giữa các biến và kết quả được gọi là*Thiên kiến*. Hình 6.15 cho thấy một đường thẳng cố gắng nối tất cả các điểm. Bởi vì nó không cắt qua tất cả các điểm, nó có độ chệch cao.

Tuy nhiên, đường cong trong Hình 6.16 có thể phù hợp với tất cả các điểm và do đó có độ chệch thấp.

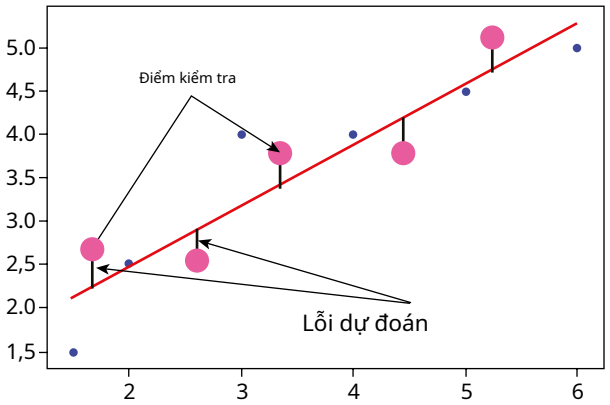
Mặc dù đường thẳng không thể đi qua tất cả các điểm và có độ chệch cao, nhưng khi áp dụng các quan sát không nhìn thấy, nó sẽ đưa ra một ước tính khá tốt. Hình 6.17 cho thấy các điểm kiểm tra (màu hồng). RSS (Tổng bình phương còn lại), là tổng các sai số của dự đoán, khá thấp so với đường cong khi sử dụng cùng các điểm kiểm tra (xem Hình 6.18).



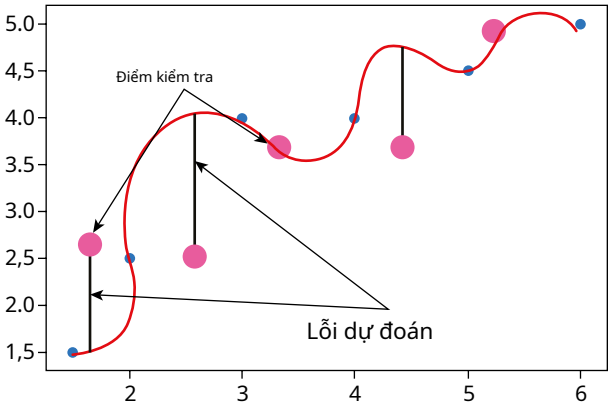
Hình 6.15:Đường thẳng không thể phù hợp với tất cả các điểm, do đó độ lệch cao



Hình 6.16:Đường cong phù hợp với tất cả các điểm nên độ lệch thấp



Hình 6.17:Đường thẳng hoạt động tốt với dữ liệu không nhìn thấy và kết quả của nó không thay đổi nhiều với các bộ dữ liệu khác nhau. Do đó, nó có phương sai thấp.



Hình 6.18:Đường cong không hoạt động tốt với dữ liệu không nhìn thấy và kết quả của nó thay đổi theo các bộ dữ liệu khác nhau. Do đó, nó có phương sai cao.

Trong học máy, sự phù hợp giữa các tập dữ liệu được gọi là*phương sai*. Trong ví dụ này, đường cong có*phương sai cao*bởi vì nó sẽ dẫn đến RSS rất khác nhau cho các bộ dữ liệu khác nhau. Có nghĩa là, bạn thực sự không thể dự đoán nó sẽ hoạt động tốt như thế nào với các bộ dữ liệu trong tương lai — đôi khi nó sẽ hoạt động tốt với một số bộ dữ liệu nhất định và những lúc khác nó có thể bị lỗi nặng. Mặt khác, đường thẳng có *phương sai thấp*, vì RSS tương tự đối với các bộ dữ liệu khác nhau.

TIỀN BOA **Trong học máy, khi chúng ta cố gắng tìm một đường cong cố gắng khớp với tất cả các điểm một cách hoàn hảo, nó được gọi là*quá sức*. Mặt khác, nếu chúng ta có một đường không phù hợp với hầu hết các điểm, nó được gọi là*thiếu trang bị*.**

Tốt nhất, chúng ta nên tìm một đường thể hiện chính xác mối quan hệ giữa các biến độc lập và kết quả. Được thể hiện dưới dạng độ chệch và phương sai, thuật toán lý tưởng phải có những điều sau đây:

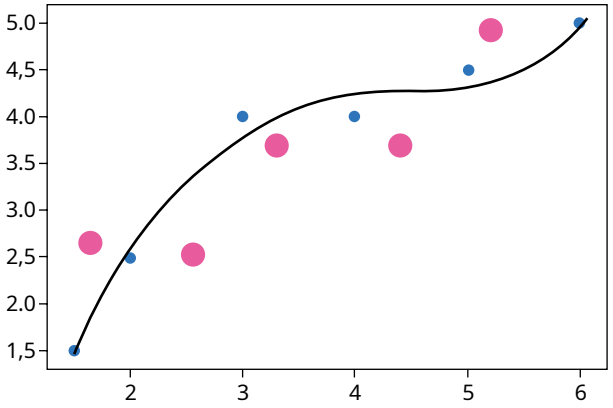
Thiên vị cao, với đường ôm càng nhiều điểm càng tốt

Phương sai thấp, với dòng dẫn đến các dự đoán nhất quán bằng cách sử dụng differ- bộ dữ liệu ent

Hình 6.19 cho thấy một đường cong lý tưởng như vậy - độ chệch cao và phương sai thấp.

Để cân bằng giữa việc tìm kiếm một mô hình đơn giản và một mô hình phức tạp, bạn có thể sử dụng các kỹ thuật như*Chính quy*,*Đóng bao*, và*Thúc đẩy*:

- *Chính quy*là một kỹ thuật tự động xử phạt các tính năng bổ sung mà bạn đã sử dụng trong mô hình của mình.
- *Đóng bao*(hoặc*tổng hợp bootstrap*) là một loại quy trình học máy cụ thể sử dụng *học tập chung*để phát triển các mô hình học máy. Việc đóng gói sử dụng một tập hợp con của dữ liệu và mỗi mẫu đào tạo một người học yếu hơn. Những người học yếu sau đó có thể được kết hợp (thông qua trung bình hoặc bình chọn tối đa) để tạo ra một người học mạnh có thể đưa ra dự đoán chính xác.



Hình 6.19:Bạn nên nhắm đến một dòng có độ chệch cao và phương sai thấp

- *Thúc đẩy*cũng tương tự như Bagging, ngoại trừ việc nó sử dụng tất cả dữ liệu để đào tạo từng người học, nhưng các điểm dữ liệu đã bị phân loại sai bởi những người học trước sẽ có trọng lượng hơn để những người học tiếp theo sẽ tập trung hơn cho họ trong quá trình đào tạo.

TIỀN BOA ***Học tập theo nhóm*là một kỹ thuật trong đó bạn sử dụng một số mô hình làm việc cùng nhau trên một tập dữ liệu duy nhất và sau đó kết hợp kết quả của nó.**

Sử dụng hồi quy bội đa thức trên tập dữ liệu Boston

Trước đó trong chương này, bạn đã sử dụng hồi quy nhiều tuyến tính và đào tạo một mô hình dựa trên tập dữ liệu Boston. Sau khi tìm hiểu về hồi quy đa thức trong phần trước, bây giờ chúng ta hãy thử áp dụng nó vào tập dữ liệu Boston và xem liệu chúng ta có thể cải thiện mô hình hay không.

Như thường lệ, hãy tải dữ liệu và chia tập dữ liệu thành các tập huấn luyện và thử nghiệm:

```
import matplotlib.pyplot as plt import
pandas as pd
nhập numpy dưới dạng np

from sklearn.preprocessing import Đa thức Tính năng từ
sklearn.linear_model import LinearRegression from sklearn.datasets
import load_boston

dataset = load_boston ()

df = pd.DataFrame (dataset.data, column = dataset.feature_names) df ['MEDV'] =
dataset.target

x = pd.DataFrame (np.c_ [df ['LSTAT'], df ['RM']], cột = ['LSTAT', 'RM']) Y = df ['MEDV']
```

```
từ sklearn.model_selection nhập train_test_split
x_train, x_test, Y_train, Y_test = train_test_split (x, Y, test_size = 0,3,
                                                     random_state = 5)
```

Sau đó, bạn sử dụng hàm đa thức với bậc 2:

```
# - - - sử dụng hàm đa thức bậc 2 --- bậc = 2

polynomial_features = Đa thức Tính năng (độ = độ) x_train_poly =
polynomial_features.fit_transform (x_train)
```

Khi sử dụng hàm đa thức bậc 2 trên hai biến độc lập $x_1$và x_2, công thức trở thành:

$$= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 + \beta_4 x x_2 + \beta_5 x_2^2$$

trong đó Y là biến phụ thuộc, $\beta_0$là điểm chặn, và β_1, β_2, β_3, và $\beta_4$là hệ số của các kết hợp khác nhau của hai đối tượng $x_1$và x_2, tương ứng.

Bạn có thể xác minh điều này bằng cách in ra các tên tính năng:

```
# - - - in ra công thức --- print (polynomial_features.get_feature_names (['x',
'y']))
```

Bạn sẽ thấy thông tin sau, trùng với công thức:

```
# ['1', 'x', 'y', 'x ^ 2', 'x y', 'y ^ 2']
```

TIỀN BOA **Biết công thức hàm đa thức rất hữu ích khi vẽ siêu phẳng 3D, bạn sẽ thực hiện ngay sau đây.**

Sau đó, bạn có thể đào tạo mô hình của mình bằng cách sử dụngTuyến tínhlớp:

```
model = LinearRegression () model.fit
(x_train_poly, Y_train)
```

Bây giờ chúng ta hãy đánh giá mô hình bằng cách sử dụng bộ thử nghiệm:

```
x_test_poly = polynomial_features.fit_transform (x_test) print ('R-Squared:%
.4f'% model.score (x_test_poly,
                                                    Y_test))
```

Bạn sẽ thấy kết quả như sau:

```
Bình phương R: 0,7340
```

Bạn cũng có thể in hệ số chặn và hệ số:

```
print (model.intercept_) print
(model.coef_)
```

Bạn sẽ thấy những điều sau:

```
26,9334305238
[0,00000000e + 00          1,47424550e + 00      - 6,70204730e + 00        7.93570743e-04
   - 3.66578385e-01        1,17188007e + 00]
```

Với những giá trị này, công thức bây giờ trở thành:

$$= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1 x_2 + \beta_5 x_2^2$$

$$= 26.9334305238 + 1.47424550e\,00\, x_1 + (-6.70204730e + 00)\, x_2 + 7.93570743e - 04\, x_1^2 + (-3.66578385e - 01)\, x_1 x_2 + 1.17188007e + 00\, x_2^2$$

Vẽ siêu phẳng 3D

Vì bạn biết hệ số chặn và hệ số của hàm hồi quy bội đa thức, bạn có thể vẽ siêu phẳng 3D của hàm một cách dễ dàng. Lưu đoạn mã sau dưới dạng tệp có tênboston2.py:

```
import matplotlib.pyplot as plt import
pandas as pd
nhập numpy dưới dạng np

từ mpl_toolkits.mplot3d nhập Axes3D
from sklearn.preprocessing import Đa thức Tính năng từ
sklearn.linear_model import LinearRegression from sklearn.datasets
import load_boston

dataset = load_boston ()

df = pd.DataFrame (dataset.data, column = dataset.feature_names) df ['MEDV'] =
dataset.target

x = pd.DataFrame (np.c_ [df ['LSTAT'], df ['RM']], cột = ['LSTAT', 'RM']) Y = df ['MEDV']


fig = plt.figure (figsize = (18,15))
ax = fig.add_subplot (111, projector = '3d')

ax.scatter (x ['LSTAT'],
                    x ['RM'],
                    Y,
                    c = 'b')
```

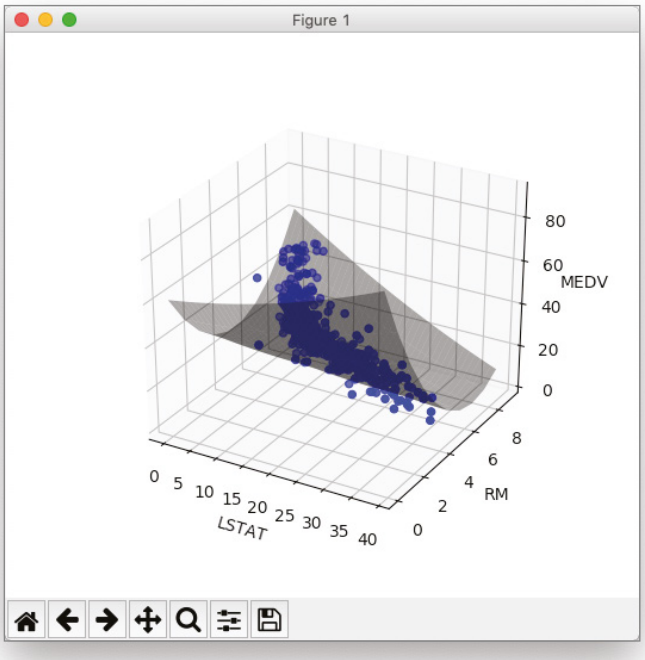
```
ax.set_xlabel ("LSTAT")
ax.set_ylabel ("RM")
ax.set_zlabel ("MEDV")

# - - - tạo lưới lưới của tất cả các giá trị cho LSTAT và RM --- x_surf = np.arange (0,
40, 1)                                  # - - - cho LSTAT ---
y_surf = np.arange (0, 10, 1)           # - - - cho RM ---
x_surf, y_surf = np.meshgrid (x_surf, y_surf)

# - - - sử dụng hàm đa thức bậc 2 --- bậc = 2

polynomial_features = PolynomialFeatures (Deg = Deg) x_poly =
polynomial_features.fit_transform (x) print
(polynomial_features.get_feature_names (['x', 'y']))

# - - - áp dụng hồi quy tuyến tính ---
model = LinearRegression () model.fit
(x_poly, Y)

# - - - tính z (MEDC) dựa trên mô hình --- z = lambda x, y:
(model.intercept_ +
                        (model.coef_ [1] * x) +
                        (model.coef_ [2] * y) +
                        (model.coef_ [3] * x ** 2) +
                        (model.coef_ [4] * x * y) +
                        ( model.coef_ [5] * y ** 2))

ax.plot_surface (x_surf, y_surf, z (x_surf, y_surf),
                        rstride = 1,
                        cstride = 1,
                        color = 'Không có',
                        alpha = 0,4)

plt.show ()
```

Để chạy mã, hãy nhập nội dung sau vào Terminal:

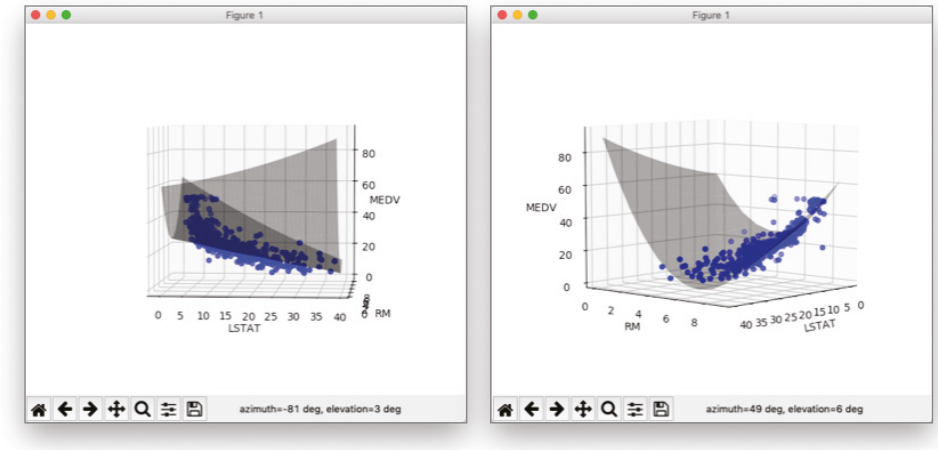
```
$ python boston2.py
```

Bạn sẽ thấy biểu đồ 3D, như trong Hình 6.20.

Bạn có thể kéo để xoay biểu đồ. Hình 6.21 cho thấy các phối cảnh khác nhau của siêu phẳng.



Hình 6.20: Siêu phẳng trong hồi quy bội đa thức



Hình 6.21: Xoay biểu đồ để xem các góc nhìn khác nhau của siêu phẳng

Bản tóm tắt

Trong chương này, bạn đã học về các loại hồi quy tuyến tính khác nhau. Đặc biệt, bạn đã học về những điều sau:

Nhiều hồi quyMối quan hệ tuyến tính giữa hai hoặc nhiều độc lập biến và một biến phụ thuộc.

Hồi quy đa thứcMô hình hóa mối quan hệ giữa một bên độc lập biến và một biến phụ thuộc bằng cách sử dụng$N_{\text{thứ tự}}$hàm đa thức bậc.

Hồi quy bội đa thứcMô hình hóa mối quan hệ giữa hai hoặc nhiều biến độc lập và một biến phụ thuộc sử dụng$N_{\text{thứ tự}}$hàm đa thức bậc.

Bạn cũng đã học cách vẽ biểu đồ siêu phẳng hiển thị mối quan hệ giữa hai biến độc lập và nhãn.