CHƯƠNG 10

Lê không giám sát arning— Sử dụng Clustering K-Means

Học không giám sát là gì?

Cho đến nay, tất cả các thuật toán học máy mà bạn đã thấy đều là học có giám sát. Có nghĩa là, tất cả các bộ dữ liệu đã được dán nhãn, phân loại hoặc phân loại. Tập dữ liệu đã được gắn nhãn được gọi là *dữ liệu được gắn nhãn*, trong khi các tập dữ liệu chưa được gắn nhãn được gọi là *dữ liệu không được gắn nhãn*. Hình 10.1 cho thấy một ví dụ về dữ liệu được gắn nhãn.

Đặc trưng		*Nhãn mác*
Kích thước của Căn nhà	Năm Được xây dựng	Giá bán Đã bán

Hình 10.1: Dữ liệu được gắn nhãn

Dựa vào kích thước của ngôi nhà và năm xây dựng, bạn sẽ có giá bán ngôi nhà. Giá bán của ngôi nhà là *nhãn mác* và mô hình học máy của bạn có thể được đào tạo để đưa ra giá trị ước tính của ngôi nhà dựa trên kích thước và năm xây dựng nó.

*Dữ liệu không được gắn nhãn*mặt khác, là dữ liệu không có (các) nhãn. Ví dụ, Hình 10.2 cho thấy một tập dữ liệu chứa một nhóm các chu vi vòng eo của mọi người và chiều dài chân tương ứng. Với bộ dữ liệu này, bạn có thể cố gắng tập hợp chúng thành các nhóm dựa trên chu vi vòng eo và chiều dài chân và từ đó bạn có thể tìm ra kích thước trung bình trong mỗi nhóm. Điều này sẽ hữu ích cho các nhà sản xuất quần áo để điều chỉnh các kích cỡ quần áo khác nhau để phù hợp với khách hàng của mình.

Hình 10.2:Dữ liệu không được gắn nhãn

Học không giám sát bằng K-Means

Vì không có nhãn trong dữ liệu chưa được gắn nhãn, do đó chúng tôi quan tâm đến việc chúng tôi có thể tìm thấy các mẫu trong dữ liệu chưa được gắn nhãn đó. Kỹ thuật tìm kiếm các mẫu trong dữ liệu không được gắn nhãn này được gọi là*nhóm lại*. Mục đích chính của phân nhóm là tách các nhóm có các đặc điểm giống nhau và phân chúng thành các nhóm (thường được gọi là *cụm*).

Một trong những thuật toán phổ biến được sử dụng để phân cụm là thuật toán K-Means. K-Means clustering là một loại học tập không giám sát:

- Được sử dụng khi bạn có dữ liệu chưa được gắn nhãn
- Mục tiêu là tìm các nhóm trong dữ liệu, với số lượng nhóm được đại diện bởi K

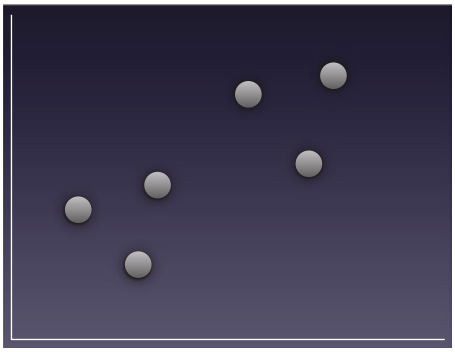
Mục tiêu của phân cụm K-Means là đạt được những điều sau:

- K centroid đại diện cho trung tâm của các cụm
- Nhãn cho dữ liệu đào tạo

Trong phần tiếp theo, bạn sẽ tìm hiểu cách hoạt động của phân cụm bằng K-Means.

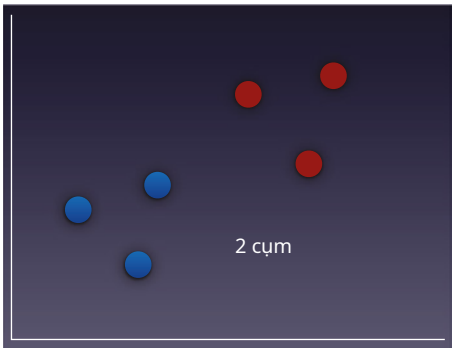
Cách Clustering trong K-Means hoạt động

Hãy xem qua một ví dụ đơn giản để bạn có thể thấy cách phân cụm sử dụng K-Means hoạt động như thế nào. Giả sử bạn có một loạt các điểm không được gắn nhãn, như trong Hình 10.3.



Hình 10.3:Tập hợp các điểm dữ liệu chưa được gắn nhãn

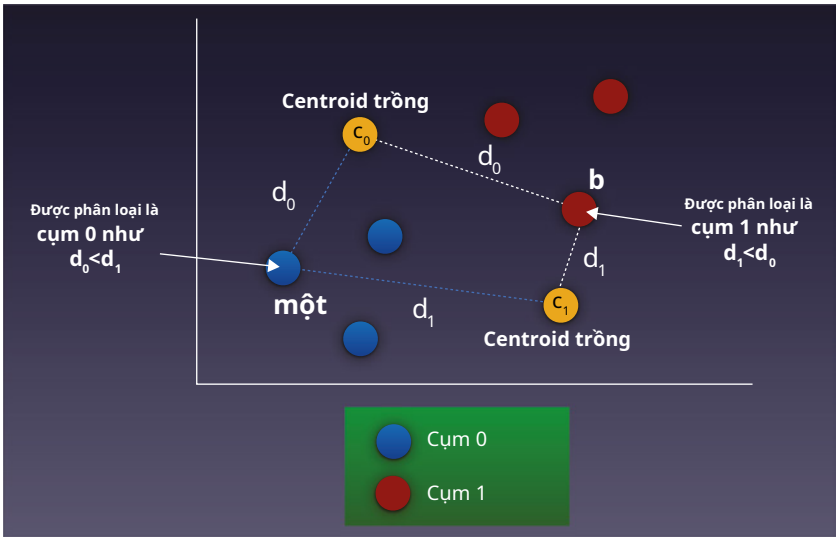
Công việc của bạn là tập hợp tất cả các điểm này thành các nhóm riêng biệt để bạn có thể khám phá ra một hình mẫu trong số đó. Giả sử bạn muốn tách chúng thành hai nhóm (nghĩa là$K = 2$).Kết quả cuối cùng sẽ giống như Hình 10.4.



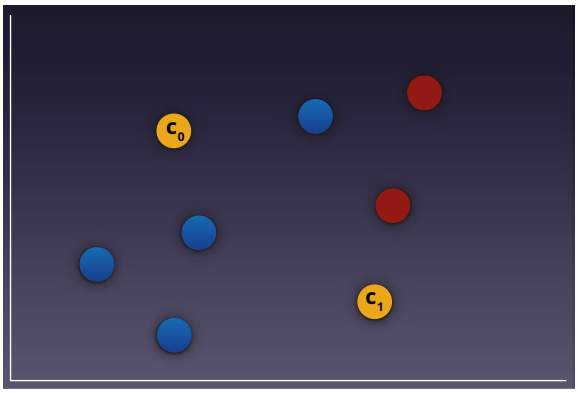
Hình 10.4: Nhóm các điểm thành hai cụm riêng biệt

Đầu tiên, bạn sẽ đặt ngẫu nhiên K số centroid trên đồ thị. Trong hình 10.5, vì K bằng 2, chúng ta sẽ đặt ngẫu nhiên hai trọng tâm trên đồ thị: C_0và C_1. Đối với mỗi điểm trên biểu đồ, đo khoảng cách giữa chính nó và mỗi tâm. Như thể hiện trong hình, khoảng cách (được biểu thị bằngd_0) giữamộtvà C_0ngắn hơn khoảng cách (được biểu thị bằngd_1) giữamộtvàC_1. Kể từ đây,mộtthiện được phân loại là cụm 0. Tương tự như vậy, đối với điểmb,khoảng cách giữa chính nó và C_1ngắn hơn khoảng cách giữa chính nó vàC_0. Do đó, điểmbđược phân loại là cụm 1. Bạn lặp lại quá trình này cho tất cả các điểm trong biểu đồ.

Sau vòng đầu tiên, các điểm sẽ được nhóm lại, như thể hiện trong Hình 10.6.



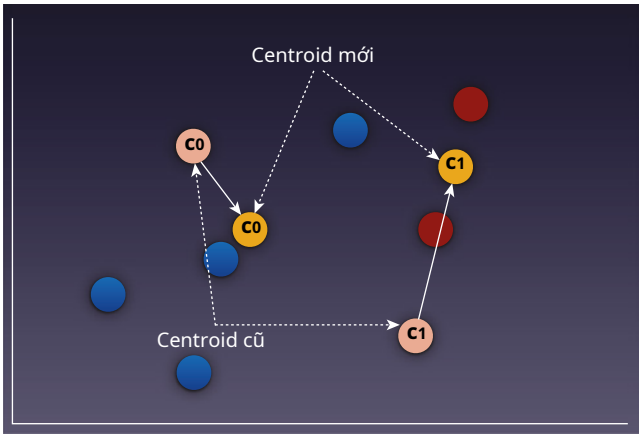
Hình 10.5:Đo khoảng cách của mỗi điểm đối với mỗi centroid và tìm khoảng cách ngắn nhất



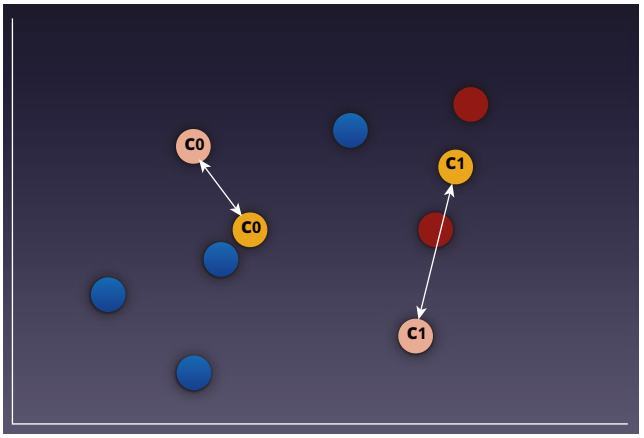
Hình 10.6: Nhóm các điểm sau vòng đầu tiên của nhóm

Bây giờ lấy trung bình của tất cả các điểm trong mỗi cụm và định vị lại các trọng tâm bằng cách sử dụng giá trị trung bình mới được tính toán. Hình 10.7 cho thấy vị trí mới của hai trung tâm.

Bây giờ bạn đo khoảng cách giữa mỗi centroid cũ và centroid mới (xem Hình 10.8). Nếu khoảng cách là 0, điều đó có nghĩa là tâm không thay đổi vị trí và do đó tâm được tìm thấy. Bạn lặp lại toàn bộ quá trình cho đến khi tất cả các trọng tâm không thay đổi vị trí nữa.



Hình 10.7:Định vị lại các trọng tâm bằng cách lấy giá trị trung bình của tất cả các điểm trong mỗi cụm



Hình 10.8:Đo khoảng cách giữa mỗi centroid; nếu khoảng cách là 0, tâm được tìm thấy

Triển khai K-Means bằng Python

Bây giờ bạn đã có một bức tranh rõ ràng về cách hoạt động của K-Means, việc triển khai điều này bằng Python sẽ rất hữu ích. Đầu tiên bạn sẽ triển khai K-Means bằng Python, sau đó xem cách bạn có thể sử dụng việc triển khai K-Means của Scikit-learning trong phần tiếp theo.

Giả sử bạn có một tệp có tênkmeans.csvvới nội dung sau:

```
x, y
1,1
2,2
```

```
2,3
1,4
3,3
6,7
7,8
6,8
7,6
6,9
2,5
7,8
8,9
6,7
7,8
3,1
8,4
8,6
8,9
```

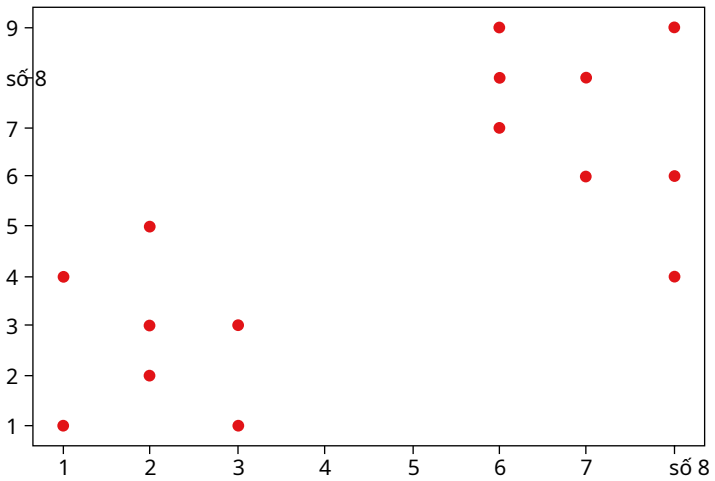
Trước tiên, hãy nhập tất cả các thư viện cần thiết:

```
% matplotlib inline
nhập numpy dưới dạng np
nhập gấu trúc dưới dạng pd
nhập matplotlib.pyplot dưới dạng plt
```

Sau đó, tải tệp CSV vào khung dữ liệu Pandas và vẽ biểu đồ phân tán hiển thị các điểm:

```
df = pd.read_csv ("kmeans.csv") plt.scatter (df ['x'], df
['y'], c = 'r', s = 18)
```

Hình 10.9 cho thấy biểu đồ phân tán với các điểm.

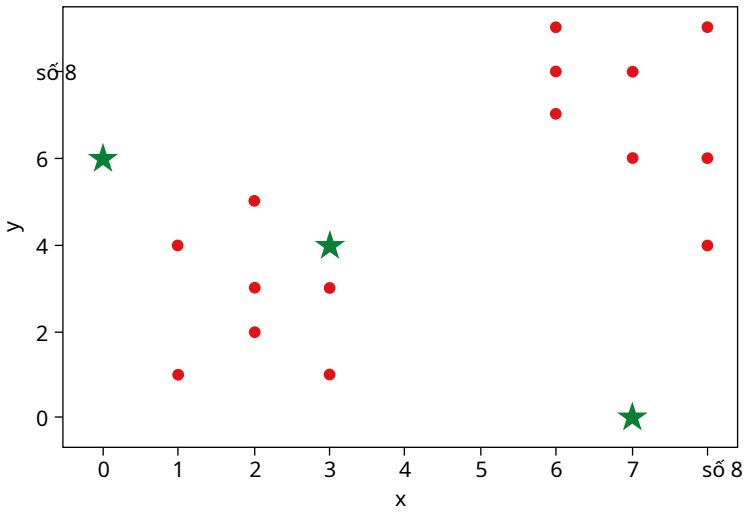


Hình 10.9: Biểu đồ phân tán hiển thị tất cả các điểm

Bây giờ bạn có thể tạo một số centroid ngẫu nhiên. Bạn cũng cần quyết định giá trị của K. Hãy giả sử K là 3 ngay bây giờ. Bạn sẽ học cách xác định K tối ưu ở phần sau của chương này. Đoạn mã sau tạo ra ba trung tâm ngẫu nhiên và đánh dấu chúng trên biểu đồ phân tán:

```
# - - - giả sử k có giá trị --- k = 3


# - - - tạo ma trận chứa tất cả các điểm --- X = np.array (list
(zip (df ['x'], df ['y'])))

# - - - tạo k điểm ngẫu nhiên (centroid) ---
Cx = np.random.randint (np.min (X [:, 0]), np.max (X [:, 0]), size = k) Cy =
np.random.randint (np.min (X [ :, 1]), np.max (X [:, 1]), size = k)

# - - - biểu diễn k centroid dưới dạng ma trận --- C = np.array (list
(zip (Cx, Cy)), dtype = np.float64) print (C)


# - - - vẽ đồ thị các điểm gốc cũng như k trọng tâm --- plt.scatter (df ['x'], df
['y'], c = 'r', s = 8)
plt.scatter (Cx, Cy, marker = '*', c = 'g', s = 160) plt.xlabel ("x")

plt.ylabel ("y")
```

Hình 10.10 cho thấy các điểm, cũng như các trọng tâm trên biểu đồ phân tán.



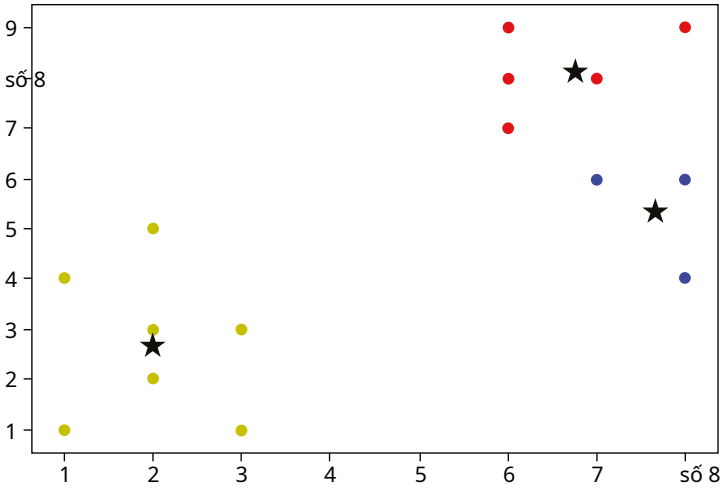
Hình 10.10: Biểu đồ phân tán với các điểm và ba trung tâm ngẫu nhiên

Bây giờ đến phần thịt thực sự của chương trình. Đoạn mã sau đây triển khai thuật toán K-Means mà chúng ta đã thảo luận trước đó trong phần “Cách phân cụm trong K-Means hoạt động”:

```
từ bản sao nhập sâu nội dung

# - - - để tính khoảng cách giữa hai điểm --- def euclidean_distance (a,
b, ax = 1):
    trả về np.linalg.norm (a - b, axis = ax)

# - - - tạo ma trận 0 có cùng thứ nguyên với C (centroid) --- C_prev = np.zeros (C.shape)


# - - - để lưu trữ cụm mà mỗi điểm thuộc về --- cluster = np.zeros
(len (X))

# - - - C là centroid ngẫu nhiên và C_prev đều là 0 ---
# - - - đo khoảng cách giữa centroid và C_prev --- distance_differences =
euclidean_distance (C, C_prev)

# - - - lặp lại miễn là vẫn có sự khác biệt trong
# khoảng cách giữa centroid trước đó và hiện tại --- while
distance_differences.any ()! = 0:
    # - - - gán mỗi giá trị cho cụm gần nhất của nó --- cho tôi trong
    phạm vi (len (X)):
        Khoảng cách = euclidean_distance (X [i], C)

        # - - - trả về các chỉ số của các giá trị nhỏ nhất dọc theo trục --- cluster = np.argmin
        (khoảng cách)
        cluster [i] = cluster

    # - - - lưu trữ các trung tâm thịnh hành ---
    C_prev = deepcopy (C)

    # - - - tìm các centroid mới bằng cách lấy giá trị trung bình --- cho i trong
    khoảng (k): # --- k là số cụm ---
        # - - - lấy tất cả các điểm trong cụm i ---
        points = [X [j] for j in range (len (X)) if cluster [j] == i] if len (points)! = 0:

            C [i] = np.mean (điểm, trục = 0)

    # - - - tìm khoảng cách giữa centroid cũ và centroid mới
trung tâm ---
    distance_differences = euclidean_distance (C, C_prev)

# - - - vẽ biểu đồ phân tán --- color = ['b', 'r', 'y',
'g', 'c', 'm'] cho tôi trong phạm vi (k):

    points = np.array ([X [j] for j in range (len (X)) if cluster [j] == i]) if len (points)> 0:
```

```
        plt.scatter (điểm [:, 0], điểm [:, 1], s = 10, c = màu [i]) khác:

        # điều này có nghĩa là một trong các cụm không có điểm in ("Plesae
        tái tạo lại các trung tâm của bạn.")

    plt.scatter (điểm [:, 0], điểm [:, 1], s = 10, c = color [i]) plt.scatter (C [:, 0], C [:, 1],
    marker = ' * ', s = 100, c =' đen ')
```

Với đoạn mã trước đó, các trọng tâm bây giờ sẽ được tính toán và hiển thị trên biểu đồ phân tán, như thể hiện trong Hình 10.11.



Hình 10.11: Biểu đồ phân tán cho thấy sự phân nhóm của các điểm cũng như các trung tâm mới được tìm thấy

TIỀN BOA **Do vị trí của các điểm, có thể các centroid bạn thu được có thể không giống với vị trí trong Hình 10.11.**

Ngoài ra, có thể có trường hợp sau khi phân cụm, không có điểm nào thuộc về một trung tâm cụ thể. Trong trường hợp này, bạn phải tạo lại centroid và thực hiện phân cụm lại.

Bây giờ bạn cũng có thể in ra các cụm mà mỗi điểm thuộc về:

```
đối với i, cụm trong liệt kê (cụm):
    print ("Điểm" + str (X [i]),
           "Cluster" + str (int (cluster)))
```

Bạn sẽ có thể thấy kết quả sau:

```
Điểm [1 1] Cụm 2 Điểm [2 2]
Cụm 2 Điểm [2 3] Cụm 2
```

```
Điểm [1 4] Cụm 2 Điểm [3 3]
Cụm 2 Điểm [6 7] Cụm 1
Điểm [7 8] Cụm 1 Điểm [6 8]
Cụm 1 Điểm [7 6] Cụm 0
Điểm [6 9] Cụm 1 Điểm [2 5]
Cụm 2 Điểm [7 8] Cụm 1
Điểm [8 9] Cụm 1 Điểm [6 7]
Cụm 1 Điểm [7 8] Cụm 1
Điểm [3 1] Cụm 2 Điểm [8 4]
Cụm 0 Điểm [ 8 6] Cụm 0
điểm [8 9] Cụm 1
```

TIỀN BOA **Các số cụm mà bạn sẽ thấy có thể không giống với các số được hiển thị trong mã trước.**

Quan trọng hơn, bạn muốn biết vị trí của từng centroid. Bạn có thể làm như vậy bằng cách in ra giá trị củaC:

```
in (C)
''
[[7.66666667        5,33333333]
  [6.77777778       8.11111111]
  [2.               2,71428571]]
''
```

Sử dụng K-Means trong Scikit-learning

Thay vì triển khai thuật toán K-Means của riêng bạn, bạn có thể sử dụngKMeans trong Scikit-học cách phân cụm. Sử dụng cùng một tập dữ liệu mà bạn đã sử dụng trong phần trước, đoạn mã sau sẽ tạo một phiên bản củaKMeans lớp có kích thước cụm là 3:

```
# - - - sử dụng sci-kit-learning --- từ
sklearn.cluster import KMeans k = 3

kmeans = KMeans (n_clusters = k)
```

Bây giờ bạn có thể đào tạo mô hình bằng cách sử dụngPhù hợp()hàm số:

```
kmeans = kmeans.fit (X)
```

Để gán nhãn cho tất cả các điểm, hãy sử dụngdự đoán()hàm số:

```
nhãn = kmeans.p Dự đoán (X)
```

Để lấy centroid, hãy sử dụngcluster_centerstài sản:

```
centroids = kmeans.cluster_centers_
```

Hãy in nhãn các cụm và các centroid và xem bạn nhận được gì:

```
in (nhãn)
in (centroid)
```

Bạn sẽ thấy những điều sau:

```
[1 1 1 1 1 0 0 0 2 0 1 0 0 0 0 1 2 2 0] [[6.77777778
                        8.11111111]
  [2.                   2,71428571]
  [7.66666667           5,33333333]]
```

TIỀN BOA **Do vị trí của các điểm, có thể centroid bạn thu được có thể không giống với centroid được hiển thị ở đây trong văn bản.**

Bây giờ chúng ta hãy vẽ các điểm và trọng tâm trên một biểu đồ phân tán:

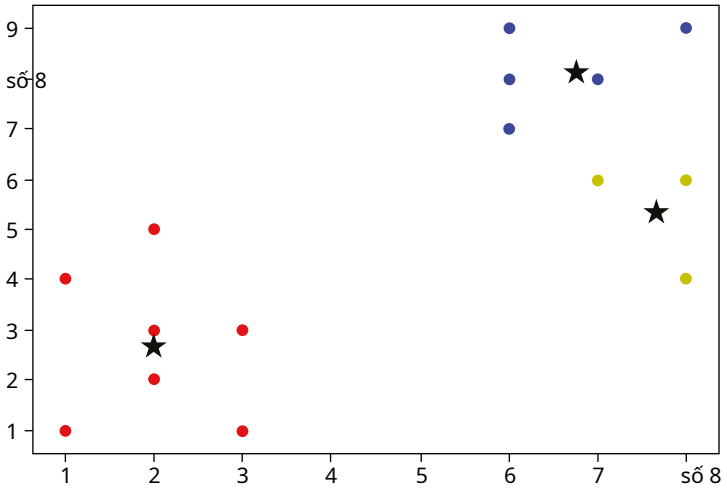
```
# - - - ánh xạ nhãn thành màu --- c = ['b', 'r',
'y', 'g', 'c', 'm'] color = [c [i] for i in label ]


plt.scatter (df ['x'], df ['y'], c = Colors, s = 18)
plt.scatter (centroid [:, 0], centroids [:, 1], marker = '*', s = 100, c = 'black')
```

Hình 10.12 cho thấy kết quả.

Sử dụng mô hình mà bạn vừa được đào tạo, bạn có thể sử dụng nó để dự đoán cụm mà một điểm sẽ thuộc vềdự đoán()hàm số:

```
# - - - đưa ra dự đoán --- cluster =
kmeans.posystem ([[3,4]]) [0] print (c [cluster]) # r


cluster = kmeans.posystem ([[7,5]]) [0] print (c
[cluster]) # y
```

Các câu lệnh trước in cụm trong đó một điểm được đặt bằng cách sử dụng màu của nó:rcho màu đỏ vàycho màu vàng.



Hình 10.12:Sử dụng lớp KMeans trong Scikit-learning để thực hiện phân nhóm

TIỀN BOA **Bạn có thể nhận được các màu khác nhau cho các điểm dự đoán, điều này hoàn toàn ổn.**

Đánh giá kích thước cụm bằng hệ số Silhouette

Cho đến nay, chúng tôi đã đặt K thành một giá trị cố định là 3. Làm thế nào để bạn đảm bảo rằng giá trị K mà bạn đã đặt là số tối ưu cho số cụm? Với một tập dữ liệu nhỏ, có thể dễ dàng suy ra giá trị của K bằng cách kiểm tra trực quan; tuy nhiên, với một tập dữ liệu lớn, nó sẽ là một nhiệm vụ khó khăn hơn. Ngoài ra, bất kể kích thước tập dữ liệu là bao nhiêu, bạn sẽ cần một cách khoa học để chứng minh rằng giá trị K bạn đã chọn là giá trị tối ưu. Để làm điều đó, bạn sẽ sử dụng Hệ số hình bóng.

Các*Hệ số Silhouette*là thước đo chất lượng phân cụm mà bạn đã đạt được. Nó đo lường sự gắn kết cụm, là không gian giữa các cụm. Phạm vi giá trị cho Hệ số Hình bóng nằm trong khoảng –1 đến 1.

Công thức Hệ số Silhouette được đưa ra như sau:

$$- a(/ b)$$

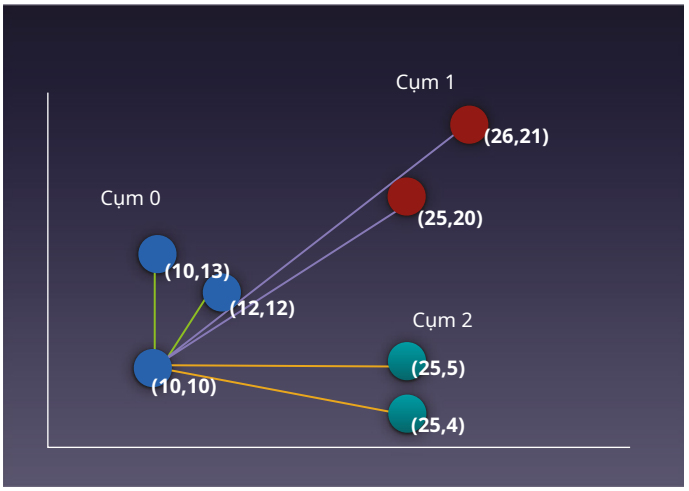
ở đâu:

- ***một***là khoảng cách trung bình của một điểm đến tất cả các điểm khác trong cùng một cụm; nếu***một***nhỏ, sự liên kết cụm tốt, vì tất cả các điểm đều gần nhau
- ***b***là*thấp nhất*khoảng cách trung bình của một điểm đến tất cả các điểm khác trong cụm gần nhất; nếu***b***lớn, tách cụm là tốt, vì cụm gần nhất cách xa nhau

Nếu***một***nhỏ và***b***lớn, Hệ số Silhouette cao. Giá trị của k mang lại Hệ số Hình bóng cao nhất được gọi là*K tối ưu*.

Tính toán hệ số Silhouette

Hãy xem qua một ví dụ về cách tính Hệ số Hình bóng của một điểm. Hãy xem xét bảy điểm và các cụm (K = 3)mà chúng thuộc về, như thể hiện trong Hình 10.13.



Hình 10.13:Tập hợp các điểm và vị trí của chúng

Hãy tính Hệ số Hình bóng của một điểm cụ thể và xem qua phép toán. Hãy xem xét điểm (10,10) trong cụm 0:

- Tính khoảng cách trung bình của nó đến tất cả các điểm khác trong cùng một cụm:
  - (10,10) - (12,12) = √8 = 2,828
  - (10,10) - (10,13) = √9 = 3
  - Trung bình: (2,828 + 3,0) / 2 = 2,914
- Tính khoảng cách trung bình của nó đến tất cả các điểm khác trong cụm 1:
  - (10,10) - (25,20) = √325 = 18,028
  - (10,10) - (26,21) = √377 = 19,416
  - Trung bình: (18.028 + 19.416) / 2 = 18.722
- Tính khoảng cách trung bình của nó đến tất cả các điểm khác trong cụm 2:
  - (10,10) - (25,5) = √250 = 15,811
  - (10,10) - (25,4) = √261 = 16,155
  - Trung bình: (15.811 + 16.156) / 2 = 15.983
- Khoảng cách trung bình tối thiểu từ (10,10) đến tất cả các điểm trong cụm 1 và 2 làtối thiểu (18,722,15,983) =15,983

Do đó, Hệ số Hình bóng của điểm (10,10) là 1 - (a / b) = 1 - (2,914 / 15,983) = 0,817681 — và đây chỉ là một điểm trong tập dữ liệu. Bạn cần tính Hệ số Hình bóng của sáu điểm khác trong tập dữ liệu. May mắn thay, Scikit-learning chứasố liệumô-đun tự động hóa quá trình này.

Sử dụngkmean.csvví dụ mà bạn đã sử dụng trước đó trong chương này, đoạn mã sau sẽ tính Hệ số Hình bóng của tất cả 19 điểm trong tập dữ liệu và in ra giá trị trung bình của Hệ số Hình bóng:

```
từ số liệu nhập khẩu của sklearn

shadow_samples = metrics.silhouette_samples (X, kmeans.labels_) print (shadow_samples)


print ("Trung bình của Hệ số Silhouette cho k =", k) print
("================================== ============ ") print ("
Silhouette có nghĩa là: ", shadow_samples.mean ())
```

Bạn sẽ thấy các kết quả sau:

```
[0,67534567 0,73722797 0,73455072 0,66254937 0,6323039                    0,33332111
  0,63792468 0,58821402 0,29141777 0,59137721 0,50802377 0,63792468 0,52511161
  0,33332111 0,63792468 0,60168807 0,51664787 0,42831295 0,52511161]


Trung bình của Hệ số Silhouette cho k = 3
========================================== ==
Silhouette có nghĩa là: 0,55780519852
```

Trong các câu lệnh trước, bạn đã sử dụngmetrics.silhouette_samples () hàm để lấy một mảng Hệ số Hình bóng cho 19 điểm. Sau đó, bạn đã gọibần tiện()trên mảng để lấy Hệ số Silhouette trung bình. Nếu bạn chỉ quan tâm đến hệ số Hình bóng trung bình chứ không phải Hệ số Hình bóng cho các điểm riêng lẻ, bạn có thể chỉ cần gọicác chỉ số. figure_score ()chức năng, như thế này:

```
print ("Silhouette mean:", metrics.silhouette_score (X, kmeans.labels_))
# Silhouette có nghĩa là: 0.55780519852
```

Tìm K tối ưu

Bây giờ bạn đã thấy cách tính Hệ số Hình bóng trung bình cho một tập dữ liệu có K cụm, điều bạn muốn làm tiếp theo là tìm K tối ưu cho bạn Hệ số Hình bóng trung bình cao nhất. Bạn có thể bắt đầu với một

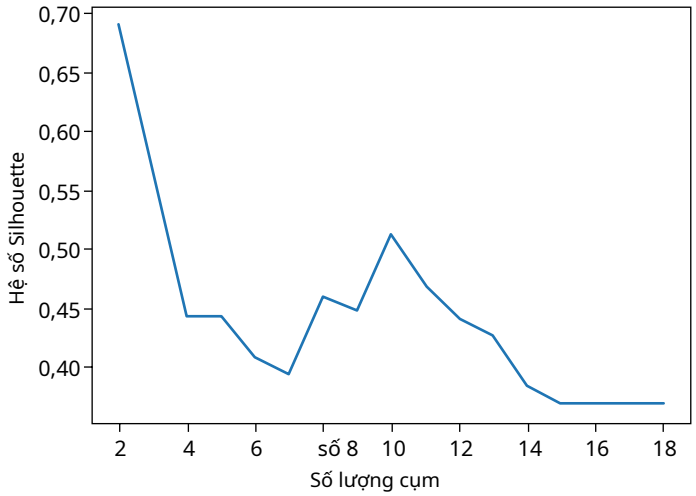
kích thước cụm là 2, tối đa kích thước cụm nhỏ hơn kích thước của tập dữ liệu. Đoạn mã sau thực hiện điều đó:

```
bóng_avgs = []
min_k = 2

# - - - thử k từ 2 đến số nhãn tối đa --- cho k trong phạm vi
(min_k, len (X)):
        kmean = KMeans (n_clusters = k) .fit (X)
        score = metrics.silhouette_score (X, kmean.labels_) print ("Hệ số Silhouette
        cho k =", k, "là", điểm) shadow_avgs.append (điểm)


f, ax = plt.subplots (figsize = (7, 5)) ax.plot (range (min_k, len
(X)), shadow_avgs)

plt.xlabel ("Số lượng cụm") plt.ylabel ("Hệ số hình
bóng")

# - - - k tối ưu là giá trị có hình bóng trung bình cao nhất --- Optimal_K = shadow_avgs.index
(max (bóng_avgs)) + min_k print ("K tối ưu là", Optimal_K)
```

Đoạn mã sẽ in ra một thứ tương tự như sau:

```
Hệ số Silhouette cho k = 2 là 0,689711206994 Hệ số Silhouette cho k = 3
là 0,55780519852 Hệ số Silhouette cho k = 4 là 0,443038181464 Hệ số
Silhouette cho k = 5 là 0,442424857695 Hệ số Silhouette đối với k = 6 là
0,40864180e1739 đối với k = 8 là 0,459039364508 Hệ số bóng cho k = 9
là 0,447750636074 Hệ số bóng cho k = 10 là 0,512411340842 Hệ số
bóng cho k = 11 là 0,469556467119 Hệ số hình bóng cho k = 12 là
0,440983139824 = 14 là 0,383836485201 Hệ số hình bóng cho k = 15 là
0,368421052632 Hệ số hình bóng cho k = 16 là 0,368421052632 Hệ số
hình bóng cho k = 17 là 0.368421052632 Hệ số Silhouette cho k = 18 là
0,368421052632 K tối ưu là 2
```

Như bạn có thể thấy từ đầu ra, K tối ưu là 2. Hình 10.14 cho thấy biểu đồ Hệ số Hình bóng được vẽ dựa trên số lượng cụm (k).



Hình 10.14: Biểu đồ hiển thị các giá trị khác nhau của K và Hệ số Hình bóng tương ứng của chúng

Sử dụng K-Means để giải quyết các vấn đề trong đời thực

Giả sử bạn là một nhà thiết kế quần áo và bạn được giao nhiệm vụ thiết kế một loạt quần short Bermuda mới. Một trong những vấn đề thiết kế là bạn cần phải đưa ra một loạt các kích thước sao cho phù hợp với hầu hết mọi người. Về cơ bản, bạn cần có một loạt quy mô của những người khác nhau:

- Chu vi vòng eo
- Chiều dài chân trên

Vì vậy, làm thế nào để bạn tìm thấy sự kết hợp phù hợp của các kích thước? Đây là lúc thuật toán K-Means trở nên hữu ích. Điều đầu tiên bạn cần làm là nắm được tập dữ liệu chứa các số đo của một nhóm người (ở một độ tuổi nhất định). Sử dụng tập dữ liệu này, bạn có thể áp dụng thuật toán K-Means để nhóm những người này thành các cụm dựa trên số đo cụ thể của các bộ phận cơ thể của họ. Khi các cụm được tìm thấy, bây giờ bạn sẽ có một bức tranh rất rõ ràng về các kích thước mà bạn cần thiết kế.

Đối với tập dữ liệu, bạn có thể sử dụng tập dữ liệu Đo cơ thể từhttps: // data.world/rhoyt/body-measurements.Tập dữ liệu này có 27 cột và 9338 hàng. Trong số 27 cột, hai cột là những gì bạn cần:

BMXWAIST: Chu vi vòng eo (cm)

BMXLEG: Chiều dài chân trên (cm)

Đối với ví dụ này, giả sử rằng tập dữ liệu đã được lưu cục bộ với tên tệp BMX_G.csv.

Nhập dữ liệu

Đầu tiên, nhập dữ liệu vào khung dữ liệu Pandas:

```
% matplotlib inline
nhập numpy dưới dạng np
nhập gấu trúc dưới dạng pd

df = pd.read_csv ("BMX_G.csv")
```

Kiểm tra hình dạng của nó, và bạn sẽ thấy 9338 hàng và 27 cột:

```
print (df.shape)
# (9338, 27)
```

Làm sạch dữ liệu

Tập dữ liệu chứa một số giá trị bị thiếu, vì vậy điều quan trọng là phải làm sạch dữ liệu. Để xem mỗi cột chứa bao nhiêu trường trống, hãy sử dụng câu lệnh sau:

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

Bạn sẽ thấy những điều sau:

```
Không tên: 0      0
seqn              0
bmdstats          0
bmxwt            95
bmiwt          8959
bmxrecum       8259
bmirecum       9307
bmxhead        9102
bmihead        9338
bmxht           723
bmiht          9070
bmxbmi          736
bmdbmic        5983
bmxleg         2383
bmileg         8984
bmxarml         512
bmiarml        8969
bmxarmc         512
bmiarmc        8965
bmxwaist       1134
bmiwaist       8882
bmxsad1        2543
bmxsad2        2543
bmxsad3        8940
```

```
bmxsad4          8940
bmdavsad         2543
bmdsadcm         8853
dtype: int64
```

Quan sát rằng cộtbmxlegcó 2383 giá trị bị thiếu vàbmxwaistcó 1134 giá trị bị thiếu, vì vậy bạn sẽ cần xóa chúng như sau:

```
df = df.dropna (subset = ['bmxleg', 'bmxwaist']) # loại bỏ các hàng bằng NaNs print (df.shape)

# (6899, 27)
```

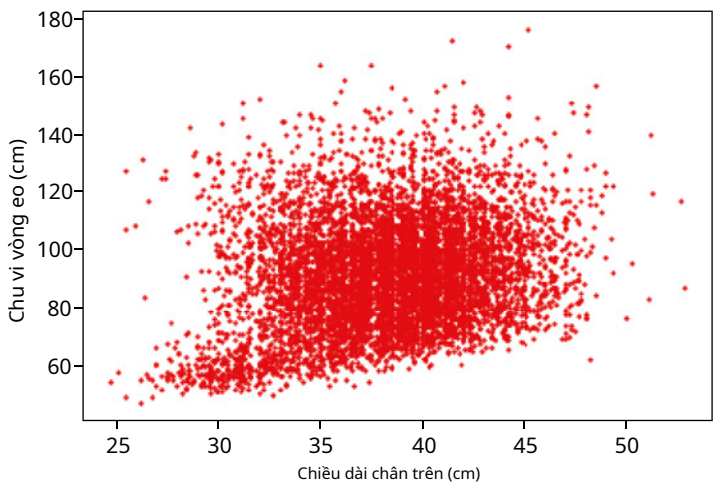
Sau khi loại bỏbmxlegvàbmxwaistcột bị thiếu giá trị, hiện còn lại 6899 hàng.

Vẽ biểu đồ phân tán

Với dữ liệu đã được làm sạch, hãy vẽ một biểu đồ phân tán cho thấy sự phân bố theo chiều dài chân trên và chu vi vòng eo:

```
nhập matplotlib.pyplot dưới dạng plt

plt.scatter (df ['bmxleg'], df ['bmxwaist'], c = 'r', s = 2) plt.xlabel ("Chiều
dài chân trên (cm)")
plt.ylabel ("Chu vi vòng eo (cm)")
```

Hình 10.15 cho thấy biểu đồ phân tán.



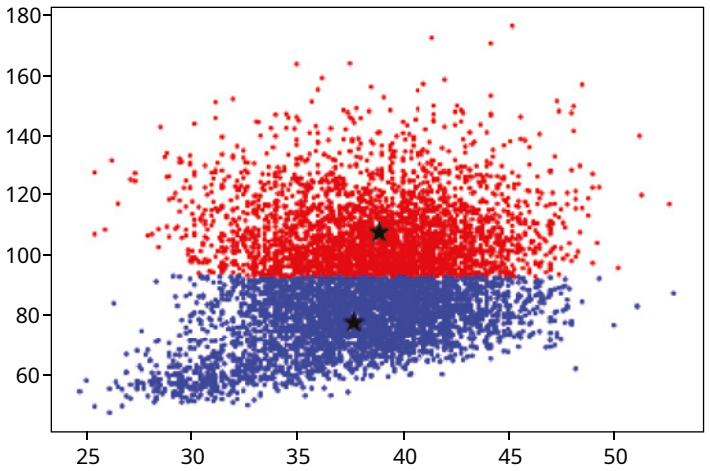
Hình 10.15: Biểu đồ phân tán cho thấy sự phân bố của chu vi vòng eo và chiều dài chân trên

Phân cụm bằng K-Means

Giả sử rằng bạn muốn tạo hai cỡ quần đùi Bermuda. Trong trường hợp này, bạn muốn gom các điểm thành hai cụm; đó là,K = 2.Một lần nữa, chúng ta có thể sử dụng Scikit-learning'sKMeanslớp cho mục đích này:

```
# - - - sử dụng sci-kit-learning --- từ
sklearn.cluster import KMeans

k = 2
X = np.array (list (zip (df ['bmxleg'], df ['bmxwaist'])))

kmeans = KMeans (n_clusters = k)
kmeans = kmeans.fit (X)
nhãn = kmeans.p Dự đoán (X) centroids =
kmeans.cluster_centers_

# - - - ánh xạ nhãn thành màu --- c = ['b', 'r',
'y', 'g', 'c', 'm'] color = [c [i] for i in label ]


plt.scatter (df ['bmxleg'], df ['bmxwaist'], c = Colors, s = 2) plt.scatter (centroids [:, 0], centroids [:, 1],
marker = '*', s = 100, c = 'đen')
```

Hình 10.16 cho thấy các điểm được tách thành hai cụm, màu đỏ và màu xanh, cùng với hai tâm.



Hình 10.16: Nhóm các điểm thành hai cụm

Đối với bạn, thông tin quan trọng nhất là giá trị của hai centroid:

```
in (centroid)
```

Bạn sẽ nhận được những điều sau:

```
[[     37,65663043          77,84326087]
 [     38.81870146        107,9195713]]
```

Điều này có nghĩa là bây giờ bạn có thể thiết kế quần short Bermuda của mình với các kích thước sau:

- Vòng eo 77,8 cm, chiều dài chân trên 37,7 cm
- Vòng eo 107,9 cm, chiều dài chân trên 38,8 cm

Tìm các loại kích thước tối ưu

Trước khi quyết định các kích thước thực tế khác nhau để thực hiện, bạn muốn xem liệu K = 2là giá trị tối ưu, do đó bạn thử các giá trị khác nhau của K từ 2 đến 10 và tìm K tối ưu:

```
từ số liệu nhập khẩu của sklearn

bóng_avgs = []
min_k = 2

# - - - thử k từ 2 đến số nhãn tối đa --- cho k trong phạm vi
(min_k, 10):
      kmean = KMeans (n_clusters = k) .fit (X)
      score = metrics.silhouette_score (X, kmean.labels_) print ("Hệ số Silhouette
      cho k =", k, "là", điểm) shadow_avgs.append (điểm)


# - - - k tối ưu là giá trị có hình bóng trung bình cao nhất --- Optimal_K = shadow_avgs.index
(max (bóng_avgs)) + min_k print ("K tối ưu là", Optimal_K)
```

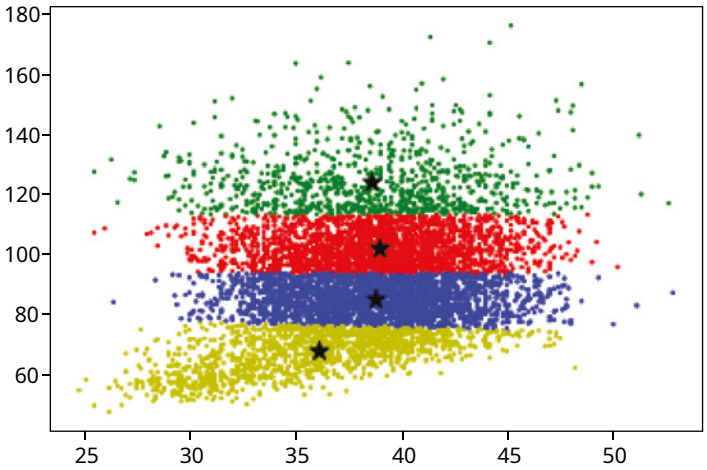
Kết quả được hiển thị ở đây:

```
Hệ số Silhouette cho k = 2 là 0,516551581494 Hệ số Silhouette cho k =
3 là 0,472269050688 Hệ số Silhouette cho k = 4 là 0,436102446644 Hệ
số Silhouette cho k = 5 là 0,418064636123 Hệ số Silhouette cho k = 6
là 0,393783 keffici95 đối với k = 8 là 0,360716292593 Hệ số hình bóng
cho k = 9 là 0,341592231958 K tối ưu là 2
```

Kết quả xác nhận rằng K tối ưu là 2. Nghĩa là, bạn nên có hai kích cỡ khác nhau cho quần short Bermuda mà bạn đang thiết kế.

Tuy nhiên, công ty muốn bạn có nhiều kích cỡ hơn để có thể đáp ứng được nhiều đối tượng khách hàng hơn. Đặc biệt, công ty cảm thấy rằng bốn kích thước sẽ là một quyết định tốt hơn. Để làm như vậy, bạn chỉ cần chạyKMeansđoạn mã mà bạn đã thấy trong phần "Phân cụm bằng cách sử dụng K-Means" và đặtk = 4.

Bây giờ bạn sẽ thấy các cụm như trong Hình 10.17.



Hình 10.17: Nhóm các điểm thành bốn cụm

Các vị trí trung tâm như sau:

```
[[    38.73004292     85.05450644]
 [    38.8849217     102.17011186]
 [    36.04064872     67.30131125]
 [    38.60124294    124.07853107]]
```

Điều này có nghĩa là bây giờ bạn có thể thiết kế quần short Bermuda của mình với các kích thước sau:

- Vòng eo 67,3 cm, chiều dài chân trên 36,0 cm
- Vòng eo 85,1 cm, chiều dài chân trên 38,7 cm
- Vòng eo 102,2 cm, chiều dài chân trên 38,9 cm
- Vòng eo 124,1 cm, chiều dài chân trên 38,6 cm

Bản tóm tắt

Trong chương này, bạn đã học về cách học không giám sát. Học không giám sát là một loại kỹ thuật học máy cho phép bạn tìm các mẫu trong dữ liệu. Trong học tập không giám sát, dữ liệu được sử dụng bởi thuật toán (ví dụ: K-Means, như được thảo luận trong chương này) không được gắn nhãn và vai trò của bạn là khám phá các cấu trúc ẩn của nó và gán nhãn cho chúng.

CHƯƠNG

11

Sử dụng máy Azure Học tập Phòng thu

Microsoft Azure Machine Learning Studio là gì?

Microsoft Azure Machine Learning Studio(do đó được gọi là*MAML*) là một công cụ cộng tác, kéo và thả trực tuyến để xây dựng các mô hình học máy. Thay vì triển khai các thuật toán máy học bằng các ngôn ngữ như Python hoặc R, MAML đóng gói các thuật toán máy học được sử dụng phổ biến nhất dưới dạng mô-đun và nó cho phép bạn xây dựng các mô hình học tập một cách trực quan bằng cách sử dụng tập dữ liệu của mình. Điều này bảo vệ các nhà khoa học dữ liệu ban đầu khỏi các chi tiết của các thuật toán, đồng thời cung cấp khả năng tinh chỉnh các siêu tham số của thuật toán cho người dùng nâng cao. Sau khi mô hình học tập được kiểm tra và đánh giá, bạn có thể xuất bản các mô hình học tập của mình dưới dạng dịch vụ web để các ứng dụng tùy chỉnh hoặc công cụ BI của bạn, chẳng hạn như Excel, có thể sử dụng nó. Hơn nữa, MAML hỗ trợ nhúng các tập lệnh Python hoặc R của bạn vào các mô hình học tập của bạn,

Trong chương này, bạn sẽ tạm dừng tất cả mã hóa mà bạn đã làm trong vài chương trước. Thay vì triển khai học máy bằng Python và Scikit-learning, bạn sẽ xem cách sử dụng MAML để thực hiện học máy một cách trực quan bằng cách sử dụng kéo và thả.