CHƯƠNG 7

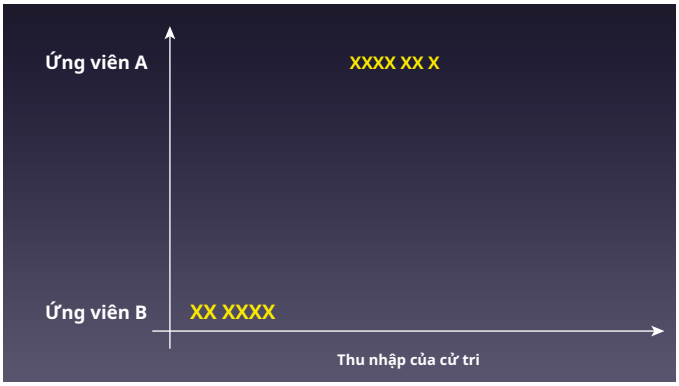
Lê giám sát arning— Phân loại Sử dụng Hậu cần Lại gression

Hồi quy logistic là gì?

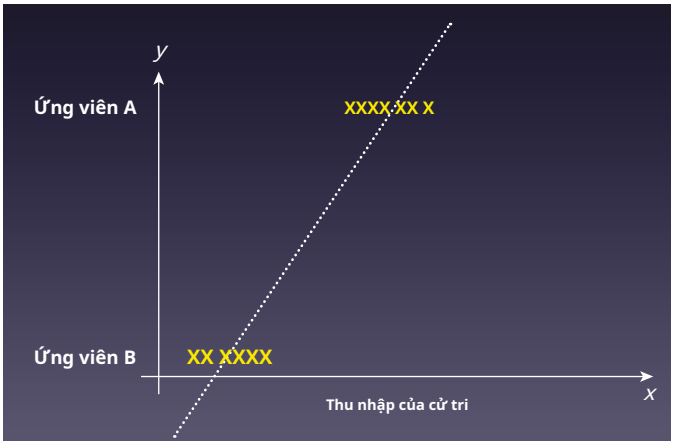
Trong chương trước, bạn đã học về hồi quy tuyến tính và cách bạn có thể sử dụng nó để dự đoán các giá trị trong tương lai. Trong chương này, bạn sẽ tìm hiểu một thuật toán học máy có giám sát khác—*hồi quy logistic*. Không giống như hồi quy tuyến tính, hồi quy logistic không cố gắng dự đoán giá trị của một biến số cho trước một tập hợp các đầu vào. Thay vào đó, đầu ra của hồi quy logistic là xác suất của một điểm đầu vào nhất định thuộc một lớp cụ thể. Đầu ra của hồi quy logistic luôn nằm trong [0,1].

Để hiểu việc sử dụng hồi quy logistic, hãy xem xét ví dụ trong Hình 7.1. Giả sử rằng bạn có một tập dữ liệu chứa thông tin về thu nhập của cử tri và sở thích bỏ phiếu. Đối với tập dữ liệu này, bạn có thể thấy rằng cử tri thu nhập thấp có xu hướng bỏ phiếu cho ứng cử viên B, trong khi cử tri thu nhập cao có xu hướng ủng hộ ứng cử viên A.

Với tập dữ liệu này, bạn sẽ rất quan tâm đến việc cố gắng dự đoán ứng cử viên nào trong tương lai sẽ bỏ phiếu dựa trên mức thu nhập của họ. Thoạt nhìn, bạn có thể bị cám dỗ để áp dụng những gì bạn vừa học được vào vấn đề này; nghĩa là, sử dụng hồi quy tuyến tính. Hình 7.2 cho thấy nó trông như thế nào khi bạn áp dụng hồi quy tuyến tính cho vấn đề này.



Hình 7.1:Một số vấn đề có kết quả nhị phân

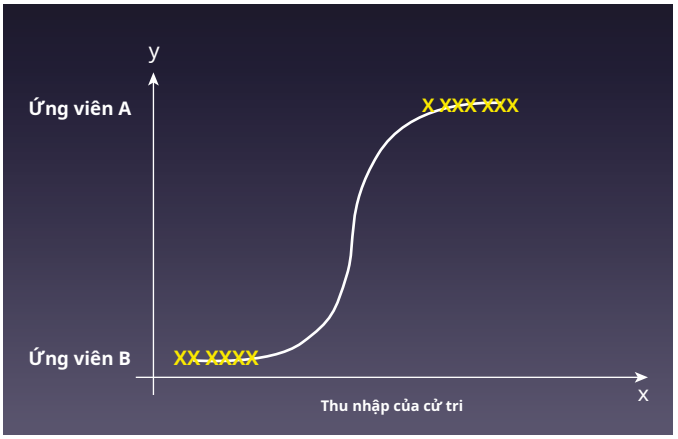


Hình 7.2:Sử dụng hồi quy tuyến tính để giải quyết vấn đề tùy chọn biểu quyết dẫn đến các giá trị lạ

Vấn đề chính của hồi quy tuyến tính là giá trị dự đoán không phải lúc nào cũng nằm trong phạm vi mong đợi. Hãy xem xét trường hợp của một cử tri có thu nhập rất thấp (gần bằng 0), và bạn có thể thấy từ biểu đồ rằng kết quả dự đoán là một giá trị âm. Những gì bạn thực sự muốn là một cách để trả về dự đoán dưới dạng giá trị từ 0 đến 1, trong đó giá trị này đại diện cho xác suất của một sự kiện xảy ra.

Hình 7.3 cho thấy cách hồi quy logistic giải quyết vấn đề này. Thay vì vẽ một đường thẳng cắt qua các điểm, bây giờ bạn sử dụng một đường cong để cố gắng vừa với tất cả các điểm trên biểu đồ.

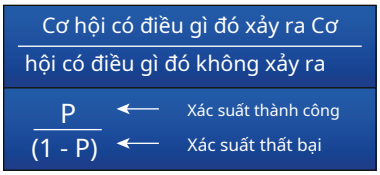
Sử dụng hồi quy logistic, đầu ra sẽ là một giá trị từ 0 đến 1, trong đó bất kỳ giá trị nào nhỏ hơn (hoặc bằng) 0,5 (được gọi là*ngưỡng cửa*) sẽ được coi là bỏ phiếu cho ứng cử viên B và bất kỳ điều gì lớn hơn 0,5 sẽ được coi là bỏ phiếu cho ứng viên A.



Hình 7.3:Hồi quy logistic dự đoán xác suất của một kết quả, thay vì một giá trị cụ thể

Hiểu tỷ lệ cược

Trước khi thảo luận về các chi tiết của thuật toán hồi quy logistic, trước tiên chúng ta cần thảo luận về một thuật ngữ quan trọng—*tỷ lệ cược*. Tỷ lệ cược được định nghĩa là tỷ lệ giữa xác suất thành công và xác suất thất bại (xem Hình 7.4).



Hình 7.4:Làm thế nào để tính toán tỷ lệ cược của một sự kiện xảy ra

Ví dụ, xác suất trúng đầu khi bạn tung đồng xu là 1. Điều này là do bạn có xác suất trúng đầu là 0,5 và xác suất trúng đuôi là 0,5. Khi bạn nói rằng tỷ lệ đánh đầu là 1, điều này có nghĩa là bạn có 50% cơ hội đánh đầu.

Nhưng nếu đồng xu được làm theo cách sao cho xác suất trúng đầu là 0,8 và xác suất trúng đuôi là 0,2, thì xác suất trúng đầu là 0,8 / 0,2 = 4. Tức là bạn gấp 4 lần. có nhiều khả năng tiếp đất bằng đầu hơn là đuôi. Tương tự, tỷ lệ xuất hiện đầu đuôi là 0,2 / 0,8 = 0,25.

Chức năng đăng nhập

Khi bạn áp dụng hàm logarit tự nhiên cho tỷ lệ cược, bạn sẽ nhận được*chức năng logit*. Hàm logit là logarit của tỷ lệ cược (xem Hình 7.5).

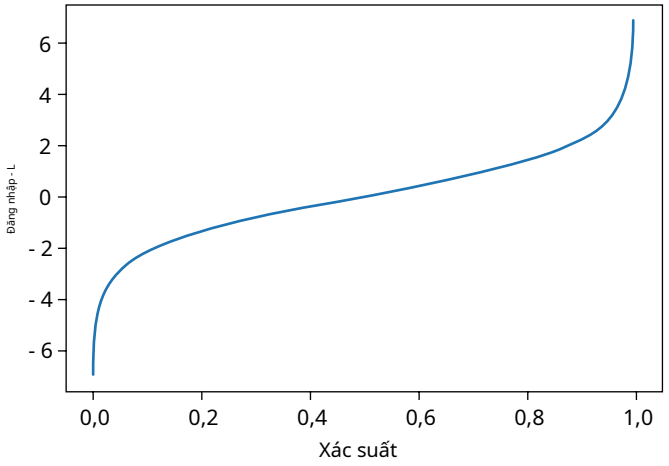
$$L = \text{Trong}\left(\frac{P}{1 - P}\right)$$

Hình 7.5:Công thức cho hàm logit

Hàm logit chuyển một biến trên (0, 1) thành một biến mới trên (–∞, ∞). Để xem mối quan hệ này, bạn có thể sử dụng đoạn mã sau:

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

def logit (x):
    trả về np.log (x / (1 - x))

x = np.arange (0,001,0.999, 0,0001) y = [logit
(n) cho n trong x]
plt.plot (x, y)
plt.xlabel ("Xác suất") plt.ylabel
("Logit - L")
```

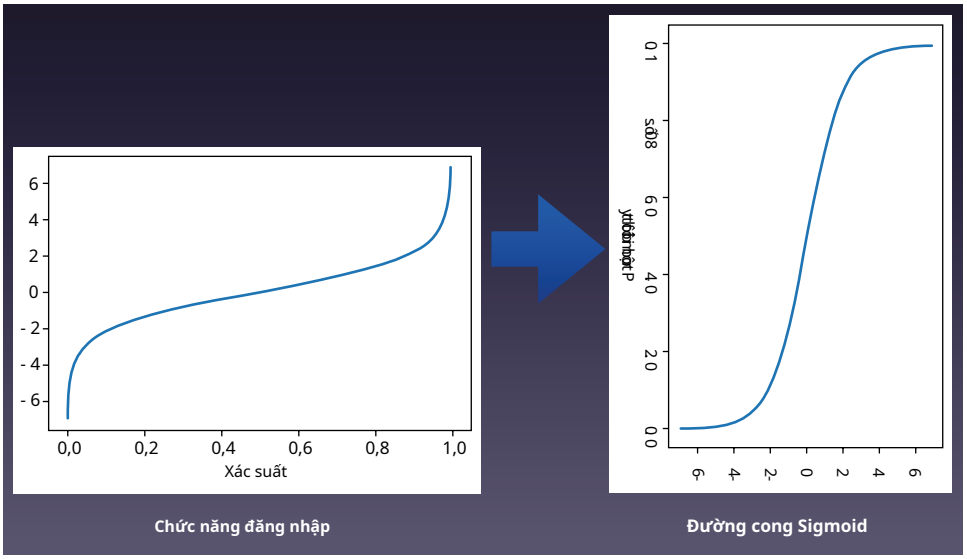
Hình 7.6 cho thấy đường cong logit được vẽ bằng đoạn mã trước đó.



Hình 7.6:Đường cong logit

Đường cong Sigmoid

Đối với đường cong logit, hãy quan sát rằng trục x là xác suất và trục y là phạm vi số thực. Đối với hồi quy logistic, những gì bạn thực sự muốn là một hàm ánh xạ các số trên hệ thống số thực với các xác suất, chính xác là những gì bạn nhận được khi lật các trục của đường cong logit (xem Hình 7.7).



Hình 7.7:Lật đường cong logit thành đường cong Sigmoid

Khi bạn lật các trục, đường cong mà bạn nhận được được gọi là*đường cong sigmoid*. Đường cong sigmoid thu được bằng cách sử dụng*Hàm Sigmoid*, là nghịch đảo của hàm logit. Hàm Sigmoid được sử dụng để biến đổi các giá trị trên (–∞, ∞) thành các số trên (0, 1). Hàm Sigmoid được thể hiện trong Hình 7.8.

$$P = \frac{1}{(1 + e^{-(L)})}$$

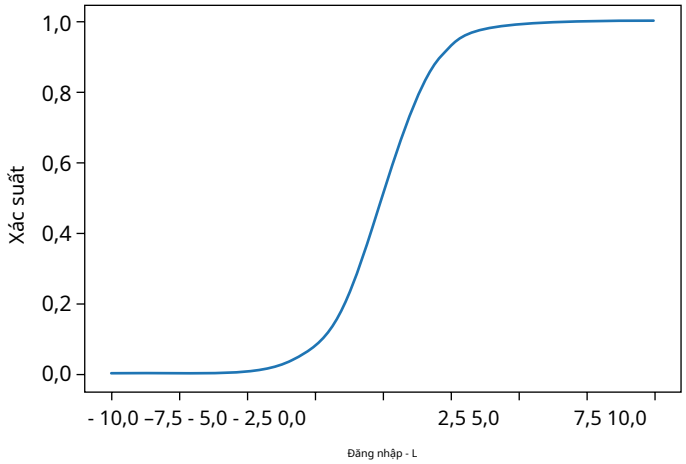
Hình 7.8:Công thức cho hàm Sigmoid

Đoạn mã sau đây cho thấy cách thu được đường cong sigmoid:

```
def sigmoid (x):
        return (1 / (1 + np.exp (-x)))

x = np.arange (-10, 10, 0,0001) y =
[sigmoid (n) cho n trong x] plt.plot (x, y)

plt.xlabel ("Logit - L") plt.ylabel
("Xác suất")
```

Hình 7.9 cho thấy đường cong sigmoid.

Giống như bạn cố gắng vẽ một đường thẳng phù hợp với tất cả các điểm trong hồi quy tuyến tính (như giải thích trong Chương 5), trong hồi quy hậu cần, chúng tôi cũng muốn vẽ một đường cong sigmoid phù hợp với tất cả các điểm. Về mặt toán học, điều này có thể được biểu thị bằng công thức thể hiện trong Hình 7.10.



Hình 7.9:Đường cong sigmoid được vẽ bằng matplotlib

$$P = \frac{1}{(1 + e^{-(\beta 0 + x\beta)})}$$

Hình 7.10: Biểu diễn hàm sigmoid bằng cách sử dụng hệ số chặn và

Lưu ý rằng sự khác biệt chính giữa công thức thể hiện trong Hình 7.8 và 7.10 là bây giờ L đã được thay thế bằng $\beta_0$và xβ. Các hệ số $\beta_0$và β chưa được biết, và chúng phải được ước tính dựa trên dữ liệu đào tạo có sẵn bằng cách sử dụng một kỹ thuật được gọi là*Ước tính khả năng tối đa (MLE)*. Trong hồi quy hậu cần, β_0được gọi là hệ số chặn và xβ được gọi là hệ số.

Sử dụng Tập dữ liệu Wisconsin (Chẩn đoán) Ung thư vú

Scikit-learning đi kèm với Tập dữ liệu Wisconsin (Chẩn đoán) Ung thư vú. Nó là một tập dữ liệu cổ điển thường được sử dụng để minh họa các phân loại nhị phân. Bộ dữ liệu này bao gồm 30 đặc điểm và chúng được tính toán từ hình ảnh số hóa của một kim hút nhỏ (FNA) của một khối vú. Nhãn của tập dữ liệu là phân loại nhị phân — M là ác tính hoặc B là lành tính. Bạn đọc quan tâm có thể kiểm tra
tìm hiểu thêm thông tin tạihttps://archive.ics.uci.edu/ml/datasets/Breast+ Cancer + Wisconsin + (Chẩn đoán).

Kiểm tra mối quan hệ giữa các tính năng

Bạn có thể tải tập dữ liệu về Ung thư vú bằng cách nhậpbộ dữ liệumô-đun từ sklearn.Sau đó, sử dụngtải _ vú _ ung thư ()chức năng như sau:

```
từ sklearn.datasets nhập load_breast_cancer ung thư =
load_breast_cancer ()
```

Hiện tại, tập dữ liệu về Ung thư vú đã được tải, việc kiểm tra mối quan hệ giữa một số đặc điểm của nó là rất hữu ích.

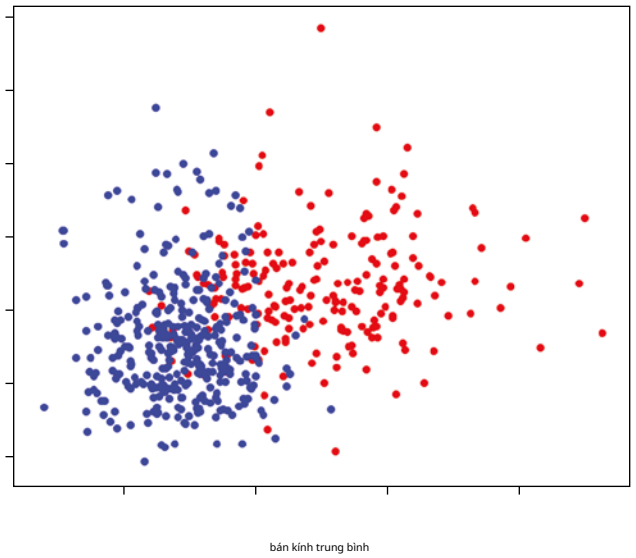
Vẽ các tính năng trong 2D

Để bắt đầu, hãy vẽ biểu đồ hai đặc điểm đầu tiên của tập dữ liệu ở dạng 2D và kiểm tra mối quan hệ của chúng. Đoạn mã sau:

- Tải tập dữ liệu về ung thư vú
- Sao chép hai tính năng đầu tiên của tập dữ liệu vào danh sách hai chiều
- Vẽ một biểu đồ phân tán cho thấy sự phân bố điểm cho hai đối tượng địa lý
- Hiển thị khối u ác tính màu đỏ và khối u lành tính màu xanh lam

```
% matplotlib inline

nhập matplotlib.pyplot dưới dạng plt
từ sklearn.datasets nhập load_breast_cancer

ung thư = load_breast_cancer ()

# - - - sao chép từ tập dữ liệu vào danh sách 2 chiều ---
X = []
cho mục tiêu trong phạm vi (2):
        X.append ([[], []])
        cho tôi trong phạm vi (len (ung thư.data)):                              # mục tiêu là 0 hoặc 1
                if ung thư.target [i] == target:
                        X [target] [0] .append (ung thư.data [i] [0]) # tính năng đầu tiên -
bán kính trung bình
                        X [target] [1] .append (ung thư.data [i] [1]) # tính năng thứ hai -
nghĩa là kết cấu

màu sắc = ("r", "b")                    # r: ác tính, b: lành tính
fig = plt.figure (figsize = (10,8)) ax =
fig.add_subplot (111)
cho mục tiêu trong phạm vi (2):
        ax.scatter (X [target] [0],
                        X [target] [1],
                        c = màu sắc [target])

ax.set_xlabel ("bán kính trung bình")
ax.set_ylabel ("kết cấu trung bình")
plt.show ()
```

Hình 7.11 cho thấy biểu đồ phân tán của các điểm.



Hình 7.11: Biểu đồ phân tán cho thấy mối quan hệ giữa bán kính trung bình và kết cấu trung bình của khối u

Từ biểu đồ phân tán này, bạn có thể thu thập rằng khi khối u phát triển theo bán kính và tăng kết cấu, thì càng có nhiều khả năng được chẩn đoán là ác tính.

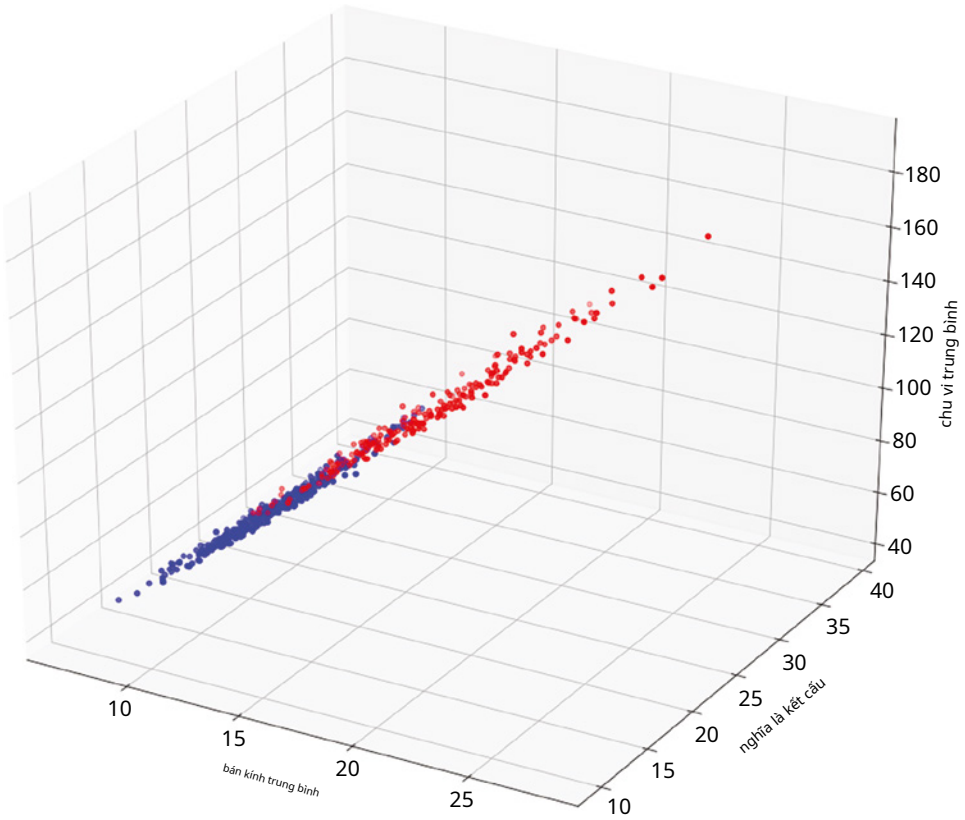
Vẽ đồ thị trong 3D

Trong phần trước, bạn đã vẽ biểu đồ các điểm dựa trên hai đặc điểm bằng cách sử dụng biểu đồ phân tán. Sẽ rất thú vị nếu có thể hình dung nhiều hơn hai tính năng. Trong trường hợp này, chúng ta hãy thử hình dung mối quan hệ giữa ba đối tượng địa lý. Bạn có thể sử dụng matplotlib để vẽ một cốt truyện 3D. Đoạn mã sau đây cho biết cách thực hiện điều này. Nó rất giống với đoạn mã trong phần trước, với các câu lệnh bổ sung được in đậm:

```
% matplotlib inline

nhập matplotlib.pyplot dưới dạng plt từ mpl_toolkits.mplot3d
nhập Axes3D từ sklearn.datasets nhập load_breast_cancer


ung thư = load_breast_cancer ()

# - - - sao chép từ tập dữ liệu vào mảng 2 chiều --- X = []

cho mục tiêu trong phạm vi (2):
        X.append ([[], [], []])
        cho tôi trong phạm vi (len (ung thư.data)):              # mục tiêu là 0,1
                if ung thư.target [i] == target:
```

```
                X [target] [0] .append (ung thư.data [i] [0]) X [target]
                [1] .append (ung thư.data [i] [1]) X [target]
                [2] .append (ung thư.data [i] [2])

màu sắc = ("r", "b")                # r: ác tính, b: lành tính
fig = plt.figure (figsize = (18,15))
ax = fig.add_subplot (111, chiếu = '3d') cho mục tiêu
trong phạm vi (2):
        ax.scatter (X [target] [0],
                    X [target] [1],
                    X [target] [2],
                    c = màu sắc [target])

ax.set_xlabel ("bán kính trung bình")
ax.set_ylabel ("kết cấu trung bình")
ax.set_zlabel ("chu vi trung bình")
plt.show ()
```

Thay vì vẽ biểu đồ bằng hai tính năng, bây giờ bạn có một tính năng thứ ba: chu vi trung bình. Hình 7.12 mô tả sơ đồ 3D.

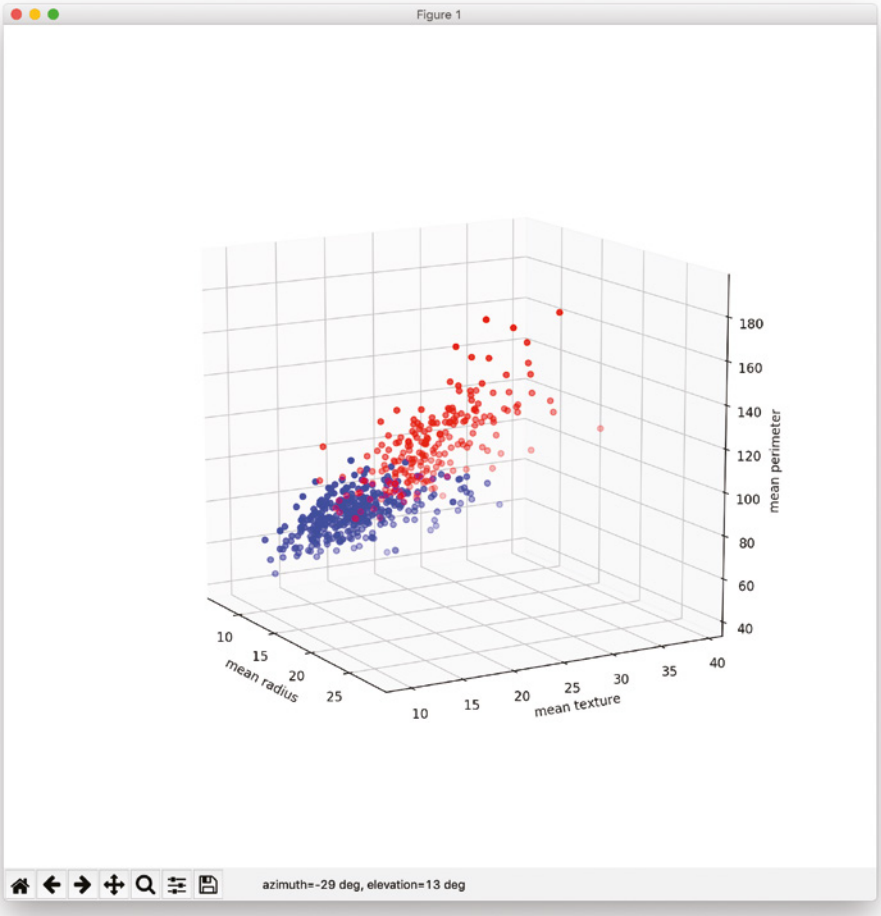


Hình 7.12:Vẽ ba đối tượng địa lý bằng bản đồ 3D

Máy tính xách tay Jupyter hiển thị cốt truyện 3D một cách tĩnh. Như bạn thấy trong Hình 7.12, bạn không thể thực sự có một cái nhìn rõ ràng về mối quan hệ giữa ba đặc điểm. Một cách tốt hơn nhiều để hiển thị cốt truyện 3D là chạy đoạn mã trước đó bên ngoài Jupyter Notebook. Để làm như vậy, hãy lưu đoạn mã (trừ dòng đầu tiên chứa câu lệnh "%matplotlib inline ")vào một tệp có tên, giả sử, 3dplot.py.Sau đó, chạy tệp trong Terminal bằng cách sử dụngcon trănlệnh, như sau:

```
$ python 3dplot.py
```

Khi bạn làm điều đó, matplotlib sẽ mở một cửa sổ riêng để hiển thị cốt truyện 3D. Hơn hết, bạn sẽ có thể tương tác với nó. Sử dụng chuột để kéo cốt truyện và bạn có thể hình dung rõ hơn các mối quan hệ giữa ba đối tượng địa lý. Hình 7.13 cho bạn một góc nhìn rõ hơn: khi chu vi phát triển trung bình của khối u tăng lên, khả năng khối u ác tính cũng tăng lên.



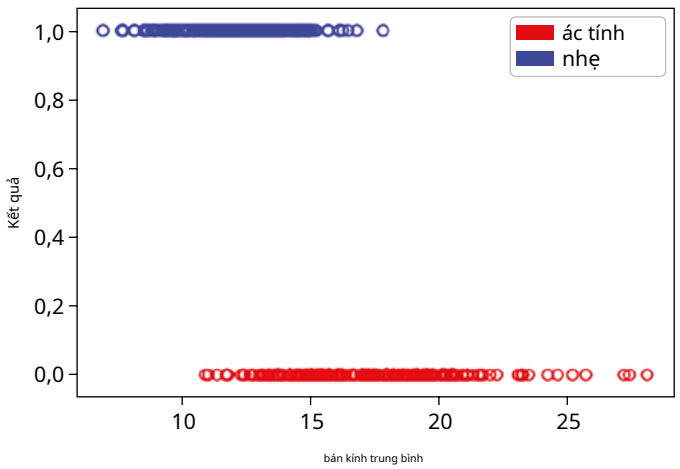
Hình 7.13:Bạn có thể tương tác với cốt truyện 3D khi chạy ứng dụng bên ngoài Jupyter Notebook

Đào tạo sử dụng một tính năng

Bây giờ chúng ta hãy sử dụng hồi quy logistic để cố gắng dự đoán xem một khối u có phải là ung thư hay không. Để bắt đầu, hãy chỉ sử dụng tính năng đầu tiên của tập dữ liệu: bán kính trung bình. Đoạn mã sau vẽ một biểu đồ phân tán cho biết khối u là ác tính hay lành tính dựa trên bán kính trung bình:

```
% matplotlib inline
nhập gấu trúc dưới dạng pd
nhập matplotlib.pyplot dưới dạng plt nhập
matplotlib.patches dưới dạng khớp

từ sklearn.datasets nhập load_breast_cancer

ung thư = load_breast_cancer () x = ung      # Tải tập dữ liệu
thư.data [:, 0]                              # bán kính trung bình
y = ung thư.target                           # 0: ác tính, 1: lành tính
color = {0: 'red', 1: 'blue'}                # 0: ác tính, 1: lành tính

plt.scatter (x, y,
                    facecolors = 'none',
                    edgecolors = pd.DataFrame (ung thư.target) [0] .apply (lambda x:
màu sắc [x]),
                    cmap = màu sắc)

plt.xlabel ("bán kính trung bình")
plt.ylabel ("Kết quả")

màu đỏ      = mpatches.Patch (color = 'red', =          label = 'ác tính')
màu xanh da trời mpatches.Patch (color = 'blue',        nhãn = 'lành tính')

plt.legend (handle = [red, blue], loc = 1)
```

Hình 7.14 cho thấy biểu đồ phân tán.



Hình 7.14:Vẽ một biểu đồ phân tán dựa trên một đặc điểm

Như bạn có thể thấy, đây là một cơ hội tốt để sử dụng hồi quy logistic để dự đoán xem một khối u có phải là ung thư hay không. Bạn có thể cố gắng vẽ một đường cong "s" (mặc dù bị lật theo chiều ngang).

Tìm điểm đánh chặn và hệ số

Scikit-learning đi kèm vớiLogisticRegressionlớp cho phép bạn áp dụng hồi quy logistic để đào tạo một mô hình. Vì vậy, trong ví dụ này, bạn sẽ đào tạo một mô hình bằng cách sử dụng tính năng đầu tiên của tập dữ liệu:

```
from sklearn import linear_model import
numpy as np

log_regress = linear_model.LogisticRegression ()

# - - - đào tạo mô hình ---
log_regress.fit (X = np.array (x) .reshape (len (x), 1),
                 y = y)

# - - - in đánh chặn mô hình được đào tạo --- print
(log_regress.intercept_)                    # [8.19393897]

# - - - in các hệ số mô hình được đào tạo --- print
(log_regress.coef_)                         # [[-0.54291739]]
```

Một khi mô hình được huấn luyện, điều chúng tôi quan tâm nhất lúc này là hệ số đánh chặn và hệ số đánh chặn. Nếu bạn nhớ lại từ công thức trong Hình 7.10, hệ số chặn là $\beta_0$và hệ số là xβ. Biết được hai giá trị này cho phép chúng ta vẽ đường cong sigmoid cố gắng vừa với các điểm trên biểu đồ.

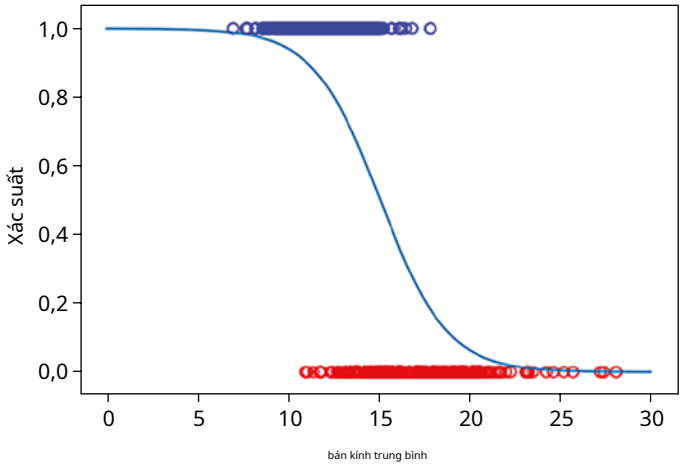
Vẽ đường cong Sigmoid

Với các giá trị của $\beta_0$và thu được xβ, bây giờ bạn có thể vẽ đồ thị đường cong sigmoid bằng cách sử dụng đoạn mã sau:

```
def sigmoid (x):
    trở lại (1 / (1 +
        np.exp (- (log_regress.intercept_ [0] +
        (log_regress.coef_ [0] [0] * x)))))

x1 = np.arange (0, 30, 0,01) y1 =
[sigmoid (n) cho n trong x1]

plt.scatter (x, y,
    facecolors = 'none',
```

```
        edgecolors = pd.DataFrame (cancer.target) [0] .apply (lambda x: Colors [x]),

        cmap = màu sắc)

plt.plot (x1, y1)
plt.xlabel ("bán kính trung bình")
plt.ylabel ("Xác suất")
```

Hình 7.15 cho thấy đường cong sigmoid.



Hình 7.15:Đường cong sigmoid phù hợp với hai tập hợp điểm

Đưa ra dự đoán

Sử dụng mô hình đã đào tạo, chúng ta hãy thử đưa ra một số dự đoán. Hãy thử dự đoán kết quả nếu bán kính trung bình là 20:

```
print (log_regress.pionary_proba (20)) # [[0.93489354 0.06510646]] print
(log_regress.posystem (20) [0]) # 0
```

Như bạn có thể thấy từ đầu ra,dự đoán _ proba ()trong câu lệnh đầu tiên trả về một mảng hai chiều. Kết quả 0.93489354 cho biết xác suất dự đoán là 0 (ác tính) trong khi kết quả 0.06510646 cho biết xác suất dự đoán là 1. Dựa trên mặc định*ngưỡng cửa*là 0,5, dự đoán là khối u ác tính (giá trị 0), vì xác suất dự đoán của nó (0,93489354) là 0 lớn hơn 0,5.

Cácdự đoán()hàm trong câu lệnh thứ hai trả về lớp mà kết quả nằm trong đó (trong trường hợp này có thể là 0 hoặc 1). Kết quả của 0 cho thấy rằng

dự đoán là khối u ác tính. Lần này, hãy thử một ví dụ khác với bán kính trung bình là 8:

```
print (log_regress.posystem_proba (8)) print        # [[0,02082411 0,97917589]]
(log_regress.pionary (8) [0])                       # 1
```

Như bạn có thể thấy từ kết quả, dự đoán là khối u là lành tính.

Đào tạo mô hình sử dụng tất cả các tính năng

Trong phần trước, bạn đã đào tạo cụ thể mô hình bằng một tính năng. Bây giờ chúng ta hãy thử đào tạo mô hình bằng cách sử dụng tất cả các tính năng và sau đó xem nó có thể thực hiện dự đoán chính xác tốt như thế nào.

Đầu tiên, tải tập dữ liệu:

```
từ sklearn.datasets nhập load_breast_cancer ung thư =
load_breast_cancer () # Tải tập dữ liệu
```

Thay vì đào tạo mô hình bằng cách sử dụng tất cả các hàng trong tập dữ liệu, bạn sẽ chia nó thành hai bộ, một để đào tạo và một để thử nghiệm. Để làm như vậy, bạn sử dụng train _ test _ split ()hàm số. Chức năng này cho phép bạn chia dữ liệu của mình thành các tập con thử nghiệm và tàu ngẫu nhiên. Đoạn mã sau chia tập dữ liệu thành tập huấn luyện 75 phần trăm và tập thử nghiệm 25 phần trăm:

```
từ sklearn.model_selection nhập train_test_split train_set, test_set, train_labels, test_labels
= train_test_split (
                                    ung thư.data,                   # Tính năng, đặc điểm
                                    ung thư.target,                 # nhãn
                                    test_size = 0,25,               # tỷ lệ phân chia
                                    random_state = 1,               # đặt ngẫu nhiên
hạt giống
                                    phân tầng = ung thư.target)     # ngẫu nhiên
dựa trên nhãn
```

Hình 7.16 cho thấy cách phân chia tập dữ liệu. Cáctrạng thái _ ngẫu nhiêntham số củatrain _ test _ split ()hàm chỉ định hạt giống được sử dụng bởi trình tạo số ngẫu nhiên. Nếu điều này không được chỉ định, mỗi khi bạn chạy chức năng này, bạn sẽ nhận được một bộ đào tạo và thử nghiệm khác nhau. Cácphân tầngtham số cho phép bạn chỉ định cột (tính năng / nhãn) nào để sử dụng để sự phân chia tương xứng. Ví dụ: nếu cột được chỉ định là một biến phân loại với 80 phần trăm 0s và 20% 1s, thì các tập huấn luyện và kiểm tra mỗi tập sẽ có 80% 0s và 20% 1s.

Hình 7.16:Tách tập dữ liệu thành các tập huấn luyện và thử nghiệm

Sau khi tập dữ liệu được chia nhỏ, bây giờ là lúc huấn luyện mô hình. Đoạn mã sau đào tạo mô hình bằng cách sử dụng hồi quy logistic:

```
từ sklearn import linear_model x = train_set
[:, 0:30]                                   # bán kính trung bình
y = train_labels                            # 0: ác tính, 1: lành tính
log_regress = linear_model.LogisticRegression () log_regress.fit
(X = x,
                        y = y)
```

Trong ví dụ này, chúng tôi đang đào tạo nó với tất cả 30 tính năng trong tập dữ liệu. Khi quá trình huấn luyện được thực hiện xong, hãy in ra các hệ số đánh chặn và mô hình:

```
print (log_regress.intercept_) print            #
(log_regress.coef_)                             #
```

Kết quả đầu ra sau đây cho thấy hệ số và hệ số chặn:

```
[0,34525124]
[[1.80079054e + 00 2.55566824e-01 -3.75898452e-02 -5.88407941e-03
  - 9.57624689e-02 -3.16671611e-01 -5.06608094e-01 -2.53148889e-01
  - 2.26083101e-01 -1.03685977e-02 4.10103139e-03 9.75976632e-01
    2.02769521e-01 -1.22268760e-01 -8.25384020e-03 -1.41322029e-02
  - 5.49980366e-02 -3.32935262e-02 -3.05606774e-02 1.09660157e-04
    1.62895414e + 00 -4.34854352e-01 -1.50305237e-01 -2.32871932e-02
  - 1.94311394e-01 -9.91201314e-01 -1.42852648e + 00 -5.40594994e-01
  - 6.28475690e-01 -9.04653541e-02]]
```

Bởi vì chúng tôi đã đào tạo mô hình bằng cách sử dụng 30 tính năng, có 30 hệ số.

Kiểm tra mô hình

Đã đến lúc đưa ra dự đoán. Đoạn mã sau sử dụng tập hợp thử nghiệm và đưa nó vào mô hình để nhận được các dự đoán:

```
nhập gấu trúc dưới dạng pd

# - - - nhận các xác suất dự đoán và chuyển đổi thành khung dữ liệu --- preds_prob =
pd.DataFrame (log_regress.posystem_proba (X = test_set))

# - - - chỉ định tên cột cho dự đoán --- preds_prob.columns =
["Ác tính", "Lành tính"]

# - - - lấy nhãn lớp được dự đoán --- preds =
log_regress.posystem (X = test_set) preds_class =
pd.DataFrame (preds) preds_class.columns = ["Dự
đoán"]

# - - - chẩn đoán thực tế --- original_result = pd.DataFrame
(test_labels) original_result.columns = ["Kết quả ban đầu"]


# - - - hợp nhất ba khung dữ liệu thành một ---
result = pd.concat ([preds_prob, preds_class, original_result], axis = 1) print (result.head ())
```

Kết quả của các dự đoán sau đó được in ra. Các dự đoán và chẩn đoán ban đầu được hiển thị cạnh nhau để dễ dàng so sánh:

Ác tính		Dự đoán lành tính	Kết quả ban đầu	
0	0,999812	1.883317e-04	0	0
1	0,998356	1.643777e-03	0	0
2	0,057992	9.420079e-01	1	1
3	1.000000	9,695339e-08	0	0
4	0,207227	7.927725e-01	1	0

Bắt ma trận nhầm lẫn

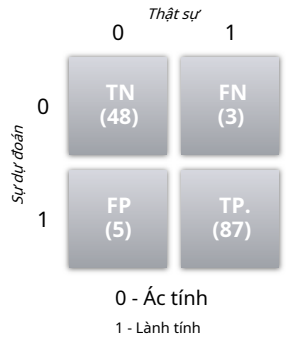
Mặc dù rất hữu ích khi in ra các dự đoán cùng với chẩn đoán ban đầu từ bộ thử nghiệm, nhưng nó không cung cấp cho bạn một bức tranh rõ ràng về mức độ tốt của mô hình trong việc dự đoán khối u có phải là ung thư hay không. Một cách khoa học hơn sẽ là sử dụng*ma trận hỗn loạn*. Ma trận nhầm lẫn hiển thị số lượng nhãn thực tế và dự đoán và bao nhiêu nhãn trong số chúng được phân loại chính xác. Bạn có thể sử dụng Pandas'scrosstab ()hàm để in ra ma trận nhầm lẫn:

```
# - - - tạo bảng dự đoán so với thực --- print ("--- Ma trận nhầm
lẫn ---")
print (pd.crosstab (preds, test_labels))
```

Cáccrosstab ()hàm tính toán một bảng chéo đơn giản của hai yếu tố. Đoạn mã trước in ra như sau:

```
- - - Ma trận nhầm lẫn --- col_0
          0     1
row_0
0        48     3
1         5    87
```

Đầu ra được giải thích như trong Hình 7.17.



Hình 7.17: Ma trận nhầm lẫn cho dự đoán

Các cột đại diện cho chẩn đoán thực tế (0 cho ác tính và 1 cho lành tính). Các hàng đại diện cho dự đoán. Mỗi hộp riêng lẻ đại diện cho một trong những điều sau:

- *Tích cực thực sự (TP)*: Mô hình dự đoán chính xác kết quả là tích cực. Trong ví dụ này, số TP (87) cho biết số dự đoán đúng rằng một khối u là lành tính.
- *Âm tính thực sự (TN)*: Mô hình dự đoán chính xác kết quả là tiêu cực. Trong ví dụ này, các khối u đã được dự đoán chính xác là ác tính.
- *Dương tính giả (FP)*: Mô hình dự đoán không chính xác kết quả là tích cực, nhưng kết quả thực tế là âm. Trong ví dụ này, có nghĩa là khối u thực sự ác tính, nhưng mô hình dự đoán khối u là lành tính.
- *Phủ định Sai (FN)*: Mô hình dự đoán không chính xác kết quả là tiêu cực, nhưng kết quả thực tế là dương. Trong ví dụ này, có nghĩa là khối u thực sự lành tính, nhưng mô hình dự đoán khối u là ác tính.

Bộ số này được gọi là*ma trận hỗn loạn*.

Bên cạnh việc sử dụngcrosstab (), bạn cũng có thể sử dụngma trận hỗn loạn() hàm để in ra ma trận nhầm lẫn:

```
từ số liệu nhập khẩu của sklearn
# - - - xem ma trận nhầm lẫn --- print (metrics.confusion_matrix (y_true
= test_labels,                                                    # Nhãn thật
                                        y_pred = preds))          # Nhãn được dự đoán
```

Lưu ý rằng đầu ra được chuyển đổi cho các hàng và cột.

```
[[48 5]
 [3 87]]
```

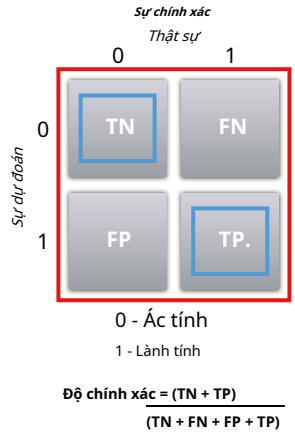
Tính chính xác, thu hồi, chính xác và các chỉ số khác

Dựa trên ma trận nhầm lẫn, bạn có thể tính toán các số liệu sau:

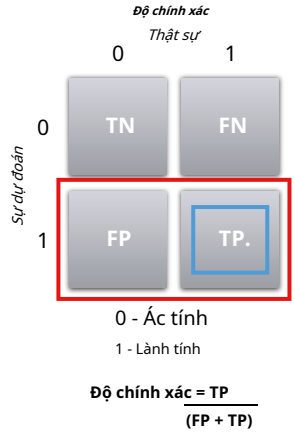
- ***Sự chính xác*:**Đây được định nghĩa là tổng của tất cả các dự đoán đúng chia cho tổng số dự đoán, hoặc theo toán học:

  $$(TP / TN) / TP (TN + FP FN + +)$$

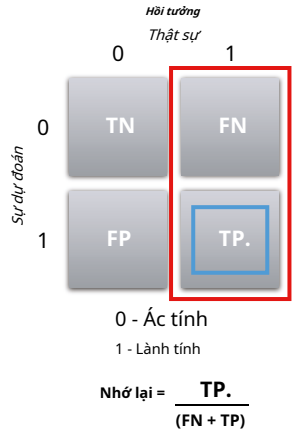
- Số liệu này rất dễ hiểu. Rốt cuộc, nếu mô hình dự đoán chính xác 99 trong số 100 mẫu, thì độ chính xác là 0,99, điều này sẽ rất ấn tượng trong thế giới thực. Nhưng hãy xem xét tình huống sau: Hãy tưởng tượng rằng bạn đang cố gắng dự đoán hư hỏng của thiết bị dựa trên dữ liệu mẫu. Trong số 1.000 mẫu, chỉ có ba mẫu bị lỗi. Nếu bạn sử dụng thuật toán câm luôn trả về số âm (có nghĩa là không có lỗi) cho tất cả các kết quả, thì độ chính xác là 997/1000, là 0,997. Điều này rất ấn tượng, nhưng điều này có nghĩa đó là một thuật toán tốt? Không. Nếu có 500 mục bị lỗi trong tập dữ liệu 1.000 mục, thì chỉ số độ chính xác ngay lập tức chỉ ra lỗi của thuật toán. Nói tóm lại, độ chính xác hoạt động tốt nhất với các điểm dữ liệu được phân bố đồng đều, nhưng nó hoạt động thực sự tồi tệ đối với một tập dữ liệu bị lệch. Hình 7.
- ***Độ chính xác*:**Chỉ số này được xác định là TP / (TP + FP). Số liệu này liên quan đến số lượng các dự đoán dương tính chính xác. Bạn có thể nghĩ về độ chính xác là "trong số những người được dự đoán là tích cực, có bao nhiêu người thực sự được dự đoán đúng?" Hình 7.19 tóm tắt công thức tính độ chính xác.
- ***Hồi tưởng*(còn được biết là *Tỷ lệ tích cực thực (TPR)*):**Chỉ số này được xác định là TP / (TP + FN). Số liệu này liên quan đến số lượng các sự kiện tích cực được dự đoán chính xác. Bạn có thể coi việc nhớ lại là “trong số những sự kiện tích cực đó, có bao nhiêu sự kiện được dự đoán chính xác?” Hình 7.20 tóm tắt công thức thu hồi.



Hình 7.18:Công thức tính độ chính xác



Hình 7.19:Công thức tính toán độ chính xác



Hình 7.20:Công thức tính toán thu hồi

- ***Điểm F1:***Số liệu này được định nghĩa là 2 * (độ chính xác * nhớ lại) / (độ chính xác + nhớ lại). Điều này được gọi là*trung bình hài hòa của độ chính xác và thu hồi*, và đó là một cách tốt để tóm tắt đánh giá của thuật toán trong một số duy nhất.
- ***Tỷ lệ dương tính giả (FPR):***Số liệu này được định nghĩa là FP / (FP + TN). FPR tương ứng với tỷ lệ các điểm dữ liệu tiêu cực bị coi là tích cực, đối với tất cả các điểm dữ liệu âm. Nói cách khác, FPR càng cao, bạn càng phân loại sai các điểm dữ liệu tiêu cực.

Khái niệm về độ chính xác và thu hồi có thể không rõ ràng ngay lập tức, nhưng nếu bạn xem xét tình huống sau đây, nó sẽ rõ ràng hơn nhiều. Hãy xem xét trường hợp chẩn đoán ung thư vú. Nếu một khối u ác tính được biểu thị là âm tính và một khối u lành tính được biểu thị là dương tính, thì:

- Nếu độ chính xác hoặc thu hồi cao, có nghĩa là nhiều bệnh nhân có khối u lành tính được chẩn đoán chính xác, điều này cho thấy thuật toán tốt.
- Nếu độ chính xác thấp, có nghĩa là nhiều bệnh nhân có khối u ác tính được chẩn đoán là lành tính.
- Nếu tỷ lệ thu hồi thấp, có nghĩa là có nhiều bệnh nhân có khối u lành tính được chẩn đoán là ác tính.

Đối với hai điểm cuối cùng, có độ chính xác thấp còn nghiêm trọng hơn tỷ lệ thu hồi thấp (mặc dù chẩn đoán sai là mắc bệnh ung thư vú khi bạn không mắc bệnh có thể sẽ dẫn đến việc điều trị không cần thiết và đau khổ về tinh thần) vì nó khiến bệnh nhân bỏ lỡ việc điều trị và có khả năng gây tử vong. Do đó, đối với các trường hợp như chẩn đoán ung thư vú, điều quan trọng là phải xem xét cả độ chính xác và số liệu thu hồi khi đánh giá hiệu quả của thuật toán ML.

Để có được độ chính xác của mô hình, bạn có thể sử dụngghi bàn()chức năng của mô hình:

```
# - - - lấy độ chính xác của dự đoán --- print ("--- Độ chính
xác ---")
print (log_regress.score (X = test_set,
                                        y = test_labels))
```

Bạn sẽ thấy kết quả sau:

```
- - - Độ chính xác ---
0,9440559440559441
```

Để có được độ chính xác, thu hồi và điểm F1 của mô hình, hãy sử dụngphân loại _ báo cáo ()chức năng củasố liệumô-đun:

```
# Xem tóm tắt các chỉ số phân loại phổ biến in ("--- Metrices ---")
```

```
print (metrics.classification_report (
        y_true = test_labels,
        y_pred = preds))
```

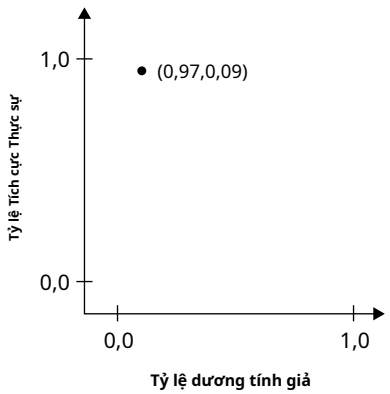
Bạn sẽ thấy kết quả sau:

```
- - - Số liệu ---
                    độ chính xác        nhớ lại điểm số f1        ủng hộ

              0          0,94          0,91          0,92          53
              1          0,95          0,97          0,96          90

trung bình / tổng số     0,94          0,94          0,94         143
```

Đường cong đặc tính hoạt động của máy thu (ROC)

Với rất nhiều số liệu có sẵn, cách dễ dàng để kiểm tra tính hiệu quả của một thuật toán là gì? Một cách sẽ là vẽ một đường cong được gọi là*Đường cong đặc tính hoạt động của máy thu (ROC)*. Đường cong ROC được tạo bằng cách vẽ đồ thị TPR so với FPR ở các cài đặt ngưỡng khác nhau.

Vì vậy, làm thế nào nó hoạt động? Hãy chạy qua một ví dụ đơn giản. Sử dụng dự án hiện có mà bạn đang làm, bạn đã suy ra ma trận nhầm lẫn dựa trên ngưỡng mặc định là 0,5 (có nghĩa là tất cả các xác suất được dự đoán nhỏ hơn hoặc bằng 0,5 thuộc về một lớp, trong khi những xác suất lớn hơn 0,5 thuộc về một lớp khác). Sử dụng ma trận nhầm lẫn này, sau đó bạn tìm thấy sự thu hồi, độ chính xác và sau đó là FPR và TPR. Sau khi tìm thấy FPR và TPR, bạn có thể vẽ điểm trên biểu đồ, như trong Hình 7.21.

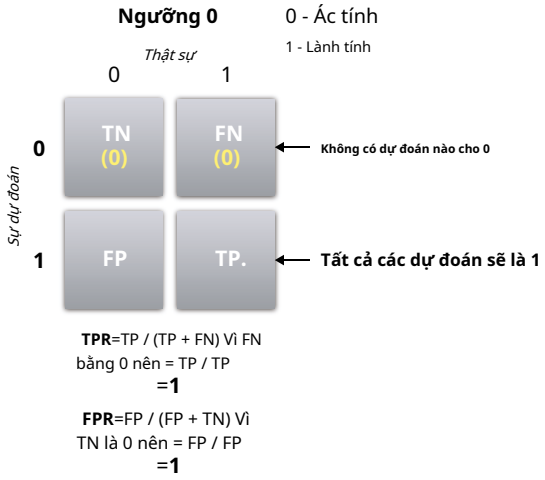


Hình 7.21:Điểm ở ngưỡng 0,5

Sau đó, bạn tạo lại ma trận nhầm lẫn cho ngưỡng 0 và tính toán lại việc thu hồi, độ chính xác, FPR và TPR. Sử dụng FPR và TPR mới, bạn lập kế hoạch

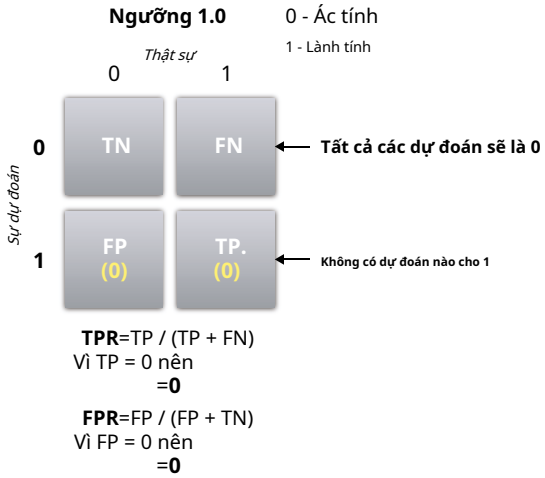
một điểm khác trên biểu đồ. Sau đó, bạn lặp lại quá trình này cho các ngưỡng 0,1, 0,2, 0,3, v.v., cho đến hết mức 1,0.

Ở ngưỡng 0, để một khối u được phân loại là lành tính (1), xác suất dự đoán phải lớn hơn 0. Do đó, tất cả các dự đoán sẽ được phân loại là lành tính (1). Hình 7.22 cho thấy cách tính TPR và FPR. Đối với ngưỡng 0, cả TPR và FPR đều là 1.



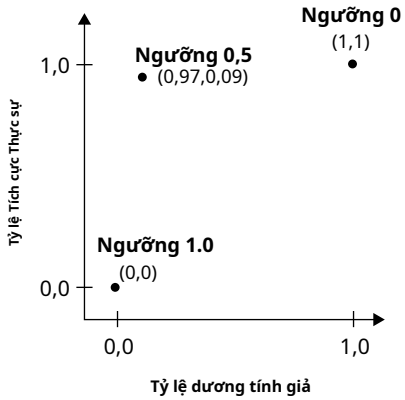
Hình 7.22: Giá trị của TPR và FPR cho ngưỡng 0

Ở ngưỡng 1,0, để một khối u được phân loại là lành tính (1), xác suất dự đoán phải bằng chính xác 1. Do đó, tất cả các dự đoán sẽ được phân loại là ác tính (0). Hình 7.23 cho thấy cách tính TPR và FPR khi ngưỡng là 1,0. Đối với ngưỡng 1,0, cả TPR và FPR đều bằng 0.



Hình 7.23: Giá trị của TPR và FPR cho ngưỡng 1

Bây giờ chúng ta có thể vẽ thêm hai điểm trên biểu đồ của mình (xem Hình 7.24).



Hình 7.24:Vẽ biểu đồ các điểm cho ngưỡng 0, 0,5 và 1,0.

Sau đó, bạn tính toán các chỉ số cho các giá trị ngưỡng khác. Tính toán tất cả các chỉ số dựa trên các giá trị ngưỡng khác nhau là một quá trình rất tẻ nhạt. May mắn thay, Scikit-learning córoc _ curve ()chức năng này sẽ tự động tính toán FPR và TPR cho bạn dựa trên các nhãn thử nghiệm được cung cấp và xác suất dự đoán:

```
from sklearn.metrics import roc_curve, auc

# - - - tìm xác suất được dự đoán bằng cách sử dụng bộ thử nghiệm probs =
log_regress.posystem_proba (test_set)
preds = probs [:, 1]

# - - - tìm FPR, TPR và ngưỡng ---
fpr, tpr, ngưỡng = roc_curve (test_labels, preds)
```

Cácroc _ curve ()hàm trả về một bộ chứa FPR, TPR và ngưỡng. Bạn có thể in chúng ra để xem các giá trị:

```
in (fpr)
in (tpr)
in (ngưỡng)
```

Bạn sẽ thấy những điều sau:

```
[0.                  0.                  0,01886792 0,01886792 0,03773585 0,03773585
   0,09433962 0,09433962 0,11320755 0,11320755 0,18867925 0,18867925 1.
                    ]

[0,01111111 0,88888889 0,88888889 0,91111111 0,91111111 0,94444444
   0,94444444 0,96666667 0,96666667 0,98888889 0,98888889 1. 1.
                    ]

[    9.99991063e-01 9.36998422e-01 9.17998921e-01 9.03158173e-01
     8.58481867e-01 8.48217940e-01 5.43424515e-01 5.26248925e-01
```

```
3.72174142e-01 2.71134211e-01 1.21486104e-01 1.18614069e-01
1.31142589e-21]
```

Như bạn có thể thấy từ đầu ra, ngưỡng bắt đầu từ 0,99999 (9,99e-01) và giảm xuống 1,311e-21.

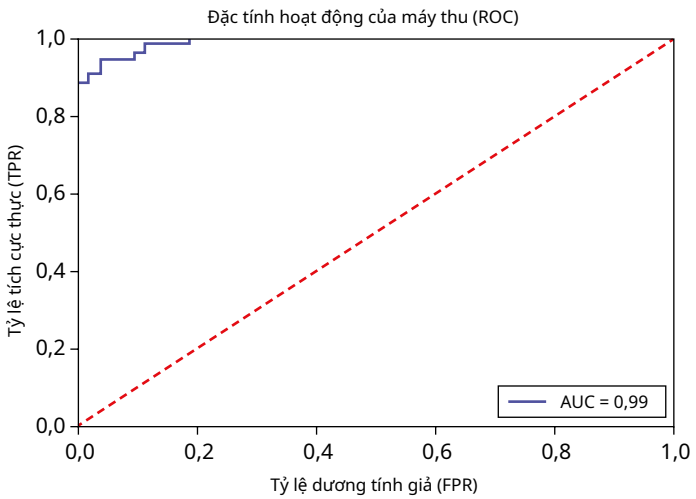
Vẽ ROC và Tìm Khu vực Dưới Đường cong (AUC)

Để vẽ ROC, bạn có thể sử dụng matplotlib để vẽ biểu đồ đường bằng cách sử dụng các giá trị được lưu trữ trong ***fpr*** và ***tpr*** biến. Bạn có thể dùng auc () chức năng để tìm khu vực dưới ROC:

```
# - - - tìm diện tích dưới đường cong --- roc_auc =
auc (fpr, tpr)

nhập matplotlib.pyplot dưới dạng plt
plt.plot (fpr, tpr, 'b', label = 'AUC =% 0.2f'% roc_auc) plt.plot ([0, 1], [0, 1], 'r--')

plt.xlim ([0, 1])
plt.ylim ([0, 1])
plt.ylabel ('Tỷ lệ dương tính thực (TPR)') plt.xlabel ('Tỷ lệ dương tính giả
(FPR)') plt.title ('Đặc tính hoạt động của máy thu (ROC)') plt.legend (loc =
'phía dưới bên phải')

plt.show ()
```

Diện tích dưới đường cong ROC là thước đo mức độ hữu ích của thử nghiệm nói chung, trong đó diện tích lớn hơn có nghĩa là thử nghiệm hữu ích hơn và các diện tích dưới đường cong ROC được sử dụng để so sánh mức độ hữu ích của các thử nghiệm. Nói chung, hãy nhắm đến thuật toán có AUC cao nhất.

Hình 7.25 cho thấy đường cong ROC cũng như AUC.



Hình 7.25: Vẽ đường cong ROC và tính AUC

Bản tóm tắt

Trong chương này, bạn đã tìm hiểu về một thuật toán học máy có giám sát khác — hồi quy hậu cần. Lần đầu tiên bạn đã học về hàm logit và cách chuyển nó thành một hàm sigmoid. Sau đó, bạn áp dụng hồi quy logistic cho tập dữ liệu ung thư vú và sử dụng nó để dự đoán xem khối u là ác tính hay lành tính. Quan trọng hơn, chương này thảo luận về một số số liệu hữu ích trong việc xác định hiệu quả của thuật toán học máy. Ngoài ra, bạn đã học về đường cong ROC là gì, cách vẽ biểu đồ và cách tính diện tích bên dưới đường cong.