CHƯƠNG 12

Máy triển khai Học tập Mô hình

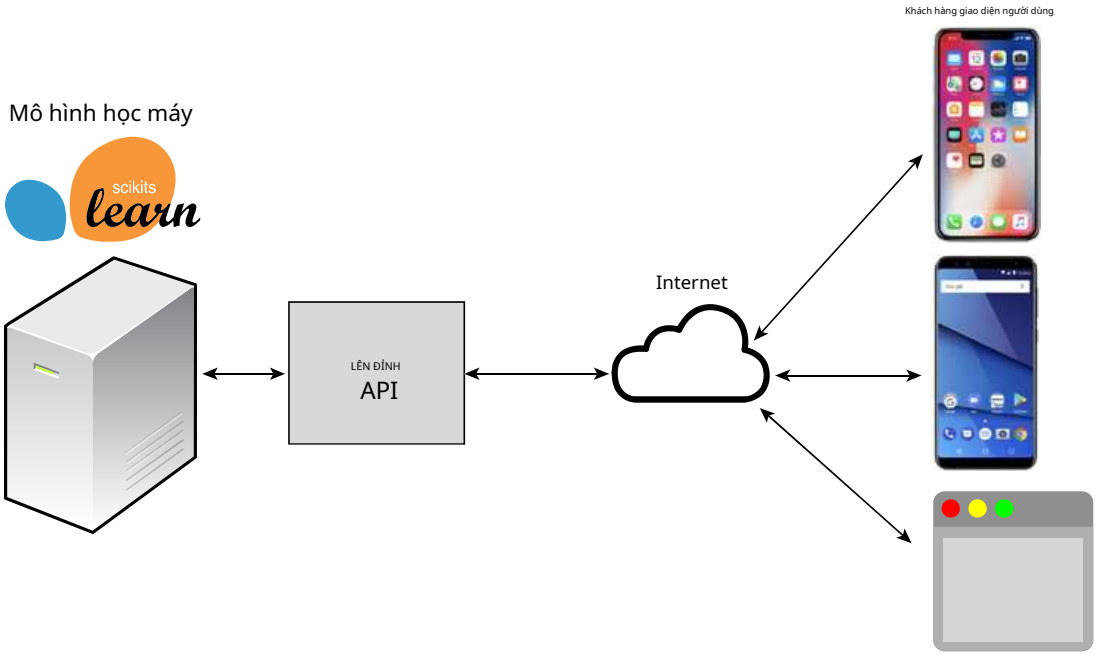
Triển khai ML

Mục tiêu chính của học máy là tạo ra một mô hình mà bạn có thể sử dụng để đưa ra dự đoán. Trong vài chương qua của cuốn sách này, bạn đã biết về các thuật toán khác nhau được sử dụng để xây dựng một mô hình học máy lý tưởng. Vào cuối toàn bộ quy trình, điều bạn thực sự muốn là làm cho người dùng có thể tiếp cận được mô hình của mình để họ có thể sử dụng nó để thực hiện các nhiệm vụ hữu ích, như đưa ra dự đoán (chẳng hạn như giúp bác sĩ chẩn đoán, v.v.).

Một cách tốt để triển khai mô hình học máy của bạn là xây dựngAPI REST (REpresentational State Transfer),để những người khác có thể không quen với cách hoạt động của máy học có thể tiếp cận được mô hình. Sử dụng REST, bạn có thể xây dựng các ứng dụng front-end đa nền tảng (chẳng hạn như iOS, Android, Windows, v.v.) và chuyển dữ liệu đến mô hình để xử lý. Sau đó, kết quả có thể được trả lại cho ứng dụng. Hình 12.1 tóm tắt kiến trúc mà chúng ta sẽ sử dụng để triển khai mô hình học máy của mình.

Trong chương này, chúng ta sẽ đi qua một nghiên cứu điển hình, xây dựng một mô hình học máy và sau đó triển khai nó như một dịch vụ REST. Cuối cùng, chúng tôi sẽ xây dựng một ứng dụng giao diện người dùng giao diện điều khiển bằng Python để cho phép người dùng đưa ra một số dự đoán.

Python® Machine Learning, First Edition. Wei-Meng Lee.



Hình 12.1:Triển khai mô hình học máy của bạn dưới dạng API REST cho phép các ứng dụng front-end sử dụng nó để dự đoán

Nghiên cứu điển hình

Đối với nghiên cứu điển hình này, chúng tôi sẽ giúp dự đoán khả năng một người được chẩn đoán mắc bệnh tiểu đường dựa trên một số phép đo chẩn đoán của người đó.

Tập dữ liệu mà bạn sẽ sử dụng trong chương này là từ cơ sở dữ liệu này: https://www.kaggle.com/uciml/pima-indians-diabetes-database.Bộ dữ liệu này chứa một số yếu tố dự đoán độc lập về y tế và một mục tiêu. Các tính năng của nó bao gồm những điều sau:

- Thời kỳ mang thai:Số lần mang thai
- Glucose:Nồng độ glucose trong huyết tương sau 2 giờ trong xét nghiệm dung nạp glucose qua đường uống
- Huyết áp:Huyết áp tâm trương (mm Hg) Độ dày da:
- Độ dày nếp gấp da cơ tam đầu (mm) Insulin:Insulin
- huyết thanh trong 2 giờ (mu U / ml)
- BMI:Chỉ số khối cơ thể (trọng lượng tính bằng kg / (chiều cao tính
- bằng m)ˆ2) Bệnh tiểu đườngChức năng phả hệ bệnh tiểu đường
- Tuổi tác:Tuổi)
- Kết quả:0 (không tiểu đường) hoặc 1 (tiểu đường)

Bộ dữ liệu có 768 hồ sơ, và tất cả bệnh nhân đều là nữ từ 21 tuổi trở lên và là người gốc Ấn Độ Pima.

Đang tải dữ liệu

Đối với ví dụ này, tập dữ liệu đã được tải xuống cục bộ và được đặt tênbệnh tiểu đường.csv.

Đoạn mã sau tải tập dữ liệu và in ra thông tin về DataFrame bằng cách sử dụngthông tin()hàm số:

```
nhập numpy dưới dạng np
nhập gấu trúc dưới dạng pd

df = pd.read_csv ('etes.csv ') df.info ()
```

Bạn sẽ thấy kết quả sau:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 mục nhập, 0 đến 767 cột Dữ liệu
(tổng cộng 9 cột): Mang thai
                                768 không rỗng   int64
Đường glucoza                   768 không rỗng   int64
Huyết áp                        768 không rỗng   int64
Độ dày da                       768 không rỗng   int64
Insulin                         768 không rỗng   int64
BMI                             768 không rỗng   float64
Bệnh tiểu đườngPedigree Tuổi chức 768 không rỗng   float64
năng                            768 không rỗng   int64
Kết quả                         768 không rỗng   int64
các loại: float64 (2),          int64 (7)
sử dụng bộ nhớ: 54,1 KB
```

Làm sạch dữ liệu

Như với tất cả các tập dữ liệu, công việc đầu tiên là làm sạch dữ liệu để không bị thiếu giá trị của bạn hoặc sai. Hãy Lần đầu tiên kiểm tra các giá trị null trong tập dữ liệu:

```
# - - - kiểm tra giá trị null --- print
("Null")
print ("=====")
print (df.isnull (). sum ())
```

Kết quả là như sau:

```
Nulls
=====
Mang thai                       0
Đường glucoza                   0
```

```
Huyết áp                                0
Độ dày da                               0
Insulin                                 0
BMI                                     0
Bệnh tiểu đườngPedigree Tuổi chức       0
năng                                    0
Kết quả                                 0
dtype: int64
```

Không có rỗng. Tiếp theo, hãy kiểm tra các số 0:

```
# - - - kiểm tra 0s --- print
("0s")
print ("==")
print (df.eq (0) .sum ())
```

Đối với các tính năng như Mang thai và Kết quả, có giá trị 0 là bình thường. Tuy nhiên, đối với các tính năng khác, giá trị 0 cho biết rằng các giá trị không được ghi lại trong tập dữ liệu.

```
0s
==
Mang thai                             111
Đường glucoza                           5
Huyết áp                               35
Độ dày da                             227
Insulin                               374
BMI                                    11
Bệnh tiểu đườngPedigree Tuổi chức       0
năng                                    0
Kết quả                               500
dtype: int64
```

Có nhiều cách để đối phó với trường hợp 0 này cho các tính năng, nhưng để đơn giản hơn, chúng ta hãy thay thế các giá trị 0 bằngNaN:

```
df [['Glucose', 'Áp lực máu', 'Độ dày da',
      'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']] df [['Glucose',          = \
      'BloodPressure', 'SkinThickness',
            'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]. Thay thế (0, np.NaN)
```

Khi các giá trị NaN đã thay thế các số 0 trong DataFrame, bây giờ bạn có thể thay thế chúng bằng giá trị trung bình của mỗi cột như sau:

```
df.fillna (df.mean (), inplace = True)                    # thay thế NaN bằng giá trị trung bình
```

Bây giờ bạn có thể kiểm tra DataFrame để xác minh rằng bây giờ không còn số 0 nào nữa trong DataFrame:

```
print (df.eq (0) .sum ())
```

Bạn sẽ thấy kết quả này:

```
Mang thai                    111
Đường glucoza                  0
Huyết áp                       0
Độ dày da                      0
Insulin                        0
BMI                            0
Bệnh tiểu đườngPedigree Tuổi chức    0
năng                           0
Kết quả                      500
dtype: int64
```

Kiểm tra mối tương quan giữa các tính năng

Bước tiếp theo là kiểm tra xem các tính năng độc lập khác nhau ảnh hưởng như thế nào đến kết quả (bệnh nhân có bị tiểu đường hay không). Để làm điều đó, bạn có thể gọicorr () trên DataFrame:

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

Cáccorr ()hàm tính toán mối tương quan theo cặp của các cột. Ví dụ, kết quả sau đây cho thấy mức đường huyết của một bệnh nhân sau thử nghiệm dung nạp đường uống trong 2 giờ có ít mối quan hệ với số lần mang thai của một bệnh nhân (0,127911), nhưng nó có mối quan hệ đáng kể với kết quả (0,492928):

```
Mang thai          Đường huyết Áp suất Da
Mang thai                1.000000  0,127911  0,208522  0,082989
Đường glucoza            0,127911  1.000000  0,218367  0,192991
Huyết áp                 0,208522  0,218367  1.000000  0,192816
Độ dày da                0,082989  0,192991  0,192816  1.000000
Insulin                  0,056027  0,420157  0,072517  0,158139
BMI                      0,021565  0,230941  0,281268  0,542398
Bệnh tiểu đường
Hàm số                  - 0,033523 0,137060 - 0,002763 0,100966
Tuổi tác                 0,544341  0,266534  0,324595  0,127872
Kết quả                  0,221898  0,492928  0,166074  0,215299

                         Insulin       BMI  Bệnh tiểu đườngPedigreeF Chức năng \
Mang thai               0,056027  0,021565                - 0,033523
Đường glucoza           0,420157  0,230941                  0,137060
Huyết áp                0,072517  0,281268                - 0,002763
Độ dày da               0,158139  0,542398                  0,100966
Insulin                 1.000000  0,166586                  0,098634
BMI                     0,166586  1.000000                  0,153400
Bệnh tiểu đường
Hàm số                  0,098634  0,153400                  1.000000
```

Tuổi tác	0,136734	0,025519	0,033561
Kết quả	0,214411	0,311924	0,173844

	Tuổi tác	Kết quả
Mang thai	0,544341	0,221898
Đường glucoza	0,266534	0,492928
Huyết áp	0,324595	0,166074
Độ dày da	0,127872	0,215299
Insulin	0,136734	0,214411
BMI	0,025519	0,311924
Bệnh tiểu đường		
Hàm số	0,033561	0,173844
Tuổi tác	1.000000	0,238356
Kết quả	0,238356	1.000000

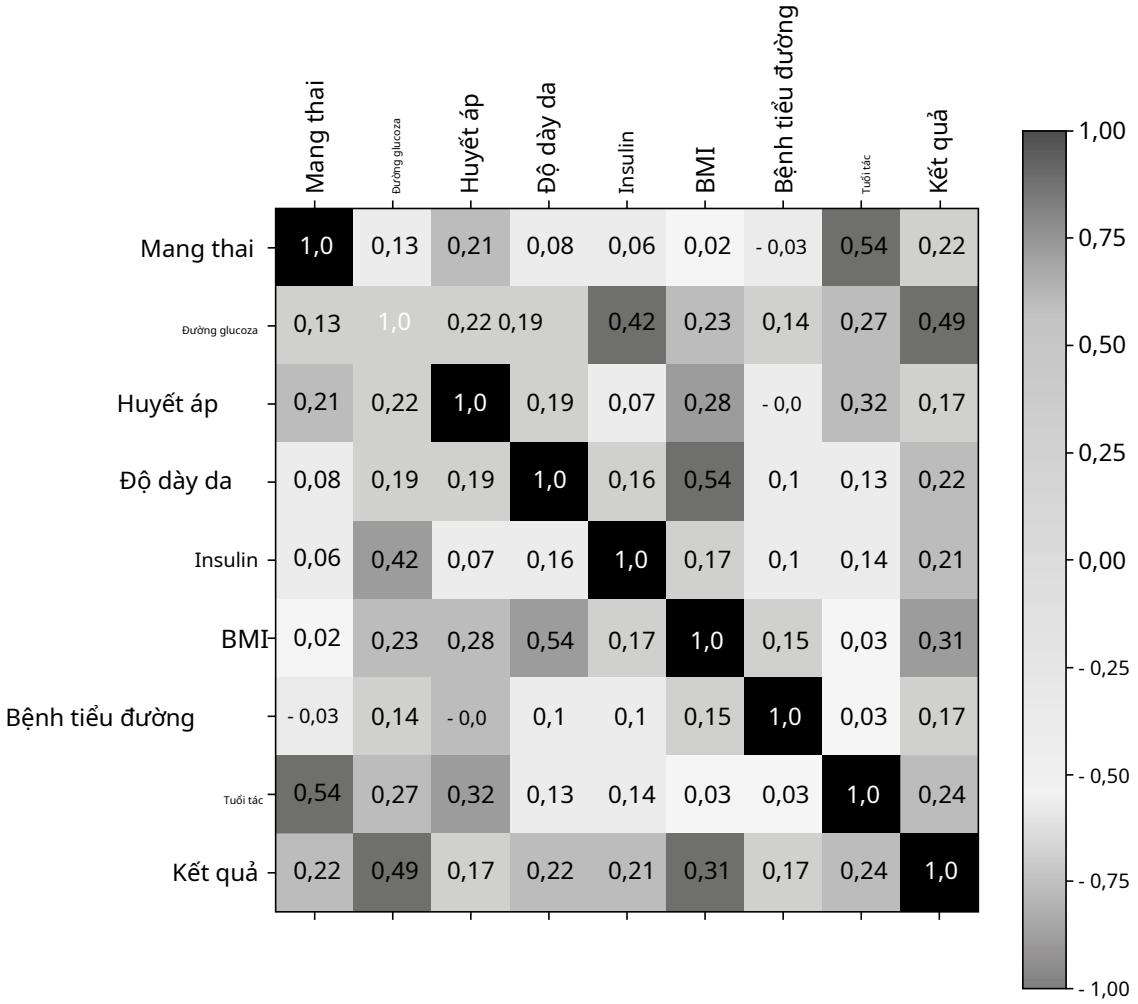
Mục tiêu của chúng tôi ở đây là tìm ra những đặc điểm nào có thể ảnh hưởng đáng kể đến kết quả.

Lập biểu đồ tương quan giữa các tính năng

Thay vì nhìn vào các con số khác nhau đại diện cho các mối tương quan khác nhau giữa các cột, sẽ rất hữu ích nếu bạn có thể hình dung nó một cách trực quan. Đoạn mã sau sử dụngchiếu ()hàm để vẽ các kết quả được trả về bởi corr ()chức năng như một ma trận. Đồng thời, các yếu tố tương quan khác nhau cũng được thể hiện trong ma trận:

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

Fig, ax = plt.subplots (figsize = (10, 10)) cax
            = ax.matshow (corr, cmap = 'coolwarm', vmin = -1, vmax = 1)

fig.colorbar (cax)
tick = np.arange (0, len (df.columns), 1) ax.set_xticks
(tick)

ax.set_xticklabels (df.columns) plt.xticks
(xoay = 90)

ax.set_yticklabels (df.columns)
ax.set_yticks (tích tắc)

# - - - in hệ số tương quan --- cho tôi trong phạm
vi (df.shape [1]):
      cho j trong phạm vi (9):
            text = ax.text (j, i, round (corr.iloc [i] [j], 2),
                                    ha = "center", va = "center", color = "w")
plt.show ()
```

Hình 12.2 cho thấy ma trận. Các hình khối có màu sắc gần với màu đỏ nhất đại diện cho các yếu tố tương quan cao nhất, trong khi các hình khối gần nhất với màu xanh lam đại diện cho các yếu tố tương quan thấp nhất.



Hình 12.2:Ma trận thể hiện các yếu tố tương quan khác nhau

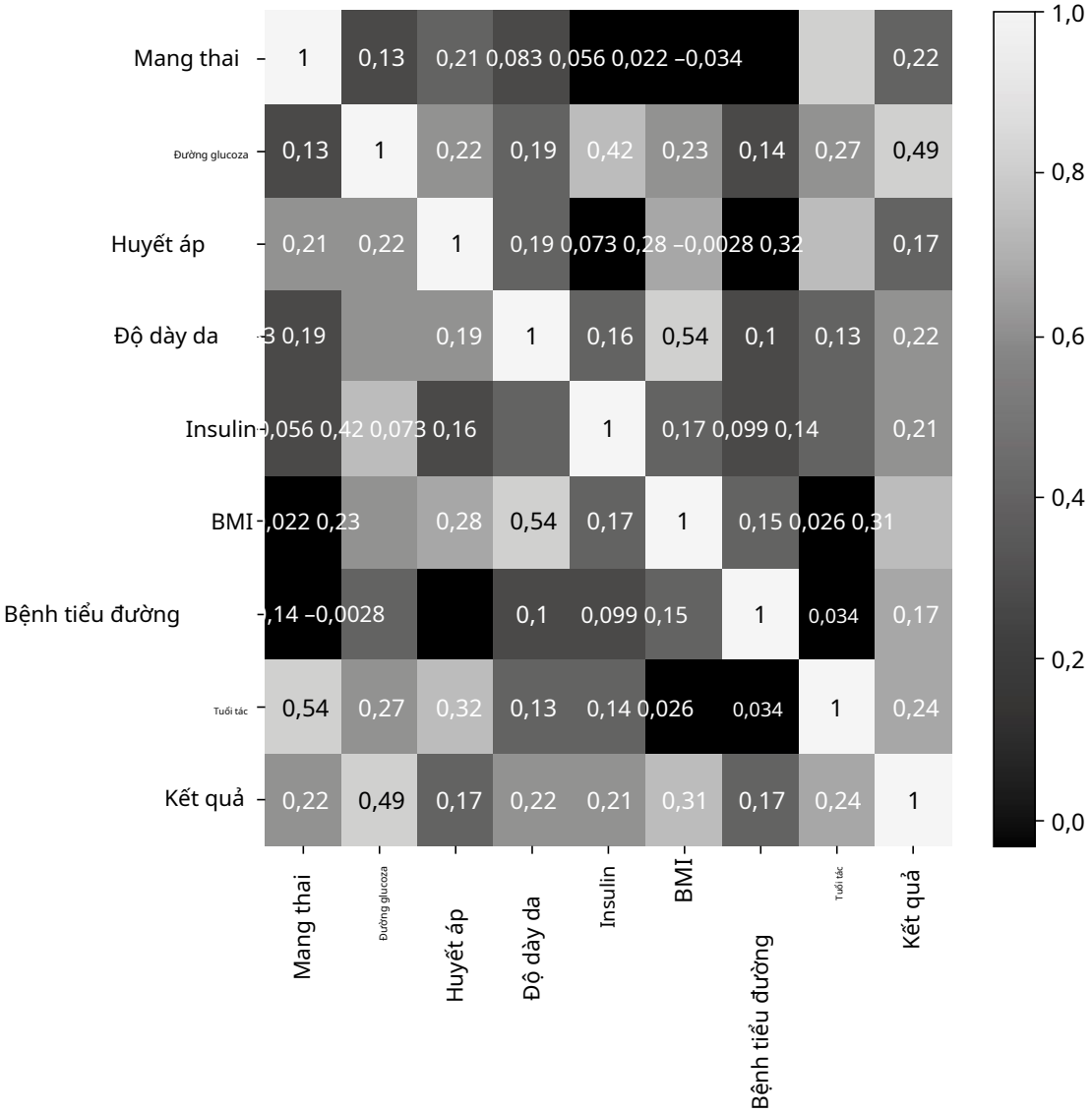
Một cách khác để vẽ ma trận tương quan là sử dụng Seaborn'sbản đồ nhiệt () chức năng như sau:

```
nhập seaborn dưới dạng sns

sns.heatmap (df.corr (), annot = True)

# - - - lấy tham chiếu đến hình hiện tại và đặt kích thước của nó --- fig = plt.gcf ()

fig.set_size_inches (8,8)
```

Hình 12.3 cho thấy bản đồ nhiệt do Seaborn tạo ra.

Bây giờ, hãy in ra bốn tính năng hàng đầu có mối tương quan cao nhất với Kết quả:

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



Hình 12.3:Bản đồ nhiệt do Seaborn sản xuất cho thấy các yếu tố tương quan

```
# - - - in 4 giá trị tương quan hàng đầu --- print (df.corr (). nlargest (4,
'Kết quả'). giá trị [:, 8])
```

Bạn sẽ thấy kết quả sau:

```
Chỉ số (['Kết quả', 'Glucose', 'BMI', 'Tuổi'], dtype = 'object') [1.
                0,49292767 0,31192439 0,23835598]
```

Bây giờ bạn có thể thấy điều đó ngoàiKết quảtính năng, ba tính năng quan trọng nhất là Glucose, BMI,vàTuổi tác.Chúng tôi có thể sử dụng ba tính năng này để đào tạo mô hình của mình.

Đánh giá các thuật toán

Trước khi chúng tôi đào tạo mô hình của mình, luôn tốt để đánh giá một số thuật toán để và cái mang lại hiệu suất tốt nhất. Theo đó, chúng tôi sẽ thử các thuật toán sau:

- Hồi quy logistic
- K-Hàng xóm gần nhất (KNN)
- Hỗ trợ Máy vectơ (SVM) —Linear và RBF Kernels

Hồi quy logistic

Đối với thuật toán đầu tiên, chúng tôi sẽ sử dụng hồi quy logistic. Thay vì chia nhỏ tập dữ liệu thành các tập huấn luyện và thử nghiệm, chúng tôi sẽ sử dụng xác thực chéo 10 lần để thu được điểm trung bình của thuật toán được sử dụng:

```
từ sklearn nhập khẩu tuyến tính
từ sklearn.model_selection nhập cross_val_score

# - - - Tính năng, đặc điểm---
X = df [['Glucose', 'BMI', 'Age']]

# - - - nhãn --- y =
df.iloc [:, 8]

log_regress = linear_model.LogisticRegression () log_regress_score = cross_val_score
(log_regress, X, y, cv = 10, Score = 'precision'). mean ()


print (log_regress_score)
```

Kết quả đào tạo mô hình nên sử dụng trung bình là 0,7617737525632263. Chúng tôi cũng sẽ lưu kết quả này vào một danh sách để chúng tôi có thể sử dụng nó để so sánh với điểm của các thuật toán khác:

```
kết quả = []
result.append (log_regress_score)
```

K-Những người hàng xóm gần nhất

Thuật toán tiếp theo mà chúng ta sẽ sử dụng là K-Nearest Neighbors (KNN). Ngoài việc sử dụng xác thực chéo 10 lần để thu được điểm trung bình của thuật toán, chúng ta cũng cần thử các giá trị khác nhau của k để thu được k tối ưu để chúng ta có được độ chính xác tốt nhất:

```
từ sklearn.neighbors nhập KNeighborsClassifier

# - - - danh sách trống sẽ chứa điểm cv (xác thực chéo) --- cv_scores = []
```

```
# - - - con số của các nếp gấp ---
nếp gấp = 10

# - - - tạo danh sách K lẻ cho KNN ---
ks = danh sách (phạm vi (1, int (len (X) * ((nếp gấp - 1) / nếp gấp)), 2))

# - - - thực hiện xác nhận chéo k-lần --- cho k trong
ks:
    knn = KNeighborsClassifier (n_neighbors = k)
    điểm = cross_val_score (knn, X, y, cv = lần xếp, điểm = 'độ chính xác'). mean () cv_scores.append
    (điểm)

# - - - đạt điểm tối đa --- knn_score =
max (cv_scores)

# - - - tìm k tối ưu cho điểm cao nhất --- tối ưu_k = ks [cv_scores.index
(knn_score)]

print (f "Số láng giềng tối ưu là {tối ưu_k}") print (knn_score)


result.append (knn_score)
```

Bạn sẽ nhận được kết quả sau:

```
Số hàng xóm tối ưu là 19 0,7721462747778537
```

Hỗ trợ Máy Vector

Thuật toán tiếp theo chúng tôi sẽ sử dụng là Máy vectơ hỗ trợ (SVM). Chúng tôi sẽ sử dụng hai loại nhân cho SVM: tuyến tính và RBF. Đoạn mã sau sử dụng nhân tuyến tính:

```
từ sklearn nhập svm

tuyến tính_svm = svm.SVC (kernel = 'tuyến tính') tuyến tính_svm_score =
cross_val_score (tuyến tính_svm, X, y,
                                                cv = 10, Score = 'precision'). mean ()
print (linear_svm_score) result.append
(linear_svm_score)
```

Bạn sẽ nhận được độ chính xác của:

```
0,7656527682843473
```

Đoạn mã tiếp theo sử dụng nhân RBF:

```
rbf = svm.SVC (kernel = 'rbf')
rbf_score = cross_val_score (rbf, X, y, cv = 10, Score = 'precision'). mean () print (rbf_score)


result.append (rbf_score)
```

Bạn sẽ nhận được độ chính xác của:

0,6353725222146275

Chọn thuật toán hoạt động tốt nhất

Bây giờ chúng tôi đã đánh giá bốn thuật toán khác nhau, chúng tôi có thể chọn một thuật toán hoạt động tốt nhất:

```
các thuật toán = ["Hồi quy logistic", "K lân cận gần nhất", "Hạt nhân tuyến tính SVM", "Hạt nhân SVM RBF"]
cv_mean = pd.DataFrame (kết quả, chỉ mục = các thuật toán)
cv_mean.columns = ["Độ chính xác"]
cv_mean.sort_values   (by = "Accuracy", ascending = False)
```

	Sự chính xác
K Những người hàng xóm gần nhất	0,772146
Hạt nhân tuyến tính SVM	0,765653
Hồi quy logistic	0,761774
Nhân SVM RBF	0,635373

Hình 12.4:Xếp hạng hiệu suất của các thuật toán khác nhau

Hình 12.4 cho thấy đầu ra của đoạn mã trước đó.

Đào tạo và lưu mô hình

Vì thuật toán hoạt động tốt nhất cho tập dữ liệu của chúng tôi là KNN với k = 19, bây giờ chúng tôi có thể tiếp tục và đào tạo mô hình của mình bằng cách sử dụng toàn bộ tập dữ liệu:

```
knn = KNeighborsClassifier (n_neighbors = 19) knn.fit (X, y)
```

Sau khi mô hình được đào tạo, bạn cần lưu nó vào đĩa để có thể truy xuất mô hình sau này cho mục đích dự đoán:

```
nhập khẩu dưa chua

# - - - lưu mô hình vào đĩa --- filename
='etes.sav '

# - - - ghi vào tệp bằng chế độ ghi và chế độ nhị phân --- pickle.dump
(knn, open (tên tệp, 'wb'))
```

Mô hình được đào tạo hiện được lưu vào một le có tênbệnh tiểu đường.sav.Hãy tải nó để đảm bảo rằng nó đã được lưu đúng cách:

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

Sau khi mô hình được tải, hãy thực hiện một số dự đoán:

```
Glucose = 65
BMI =    70
Tuổi =   50

dự đoán = loading_model.p Dự đoán ([[Glucose, BMI, Tuổi]]) in (dự đoán)

if (dự đoán [0] == 0):
        print ("Không mắc bệnh tiểu đường")
khác:
        print ("Bệnh tiểu đường")
```

Kết quả đầu ra sẽ in từ “Không mắc bệnh tiểu đường” nếu giá trị trả về của dự đoán là 0; nếu không nó sẽ in từ "Tiểu đường". Bạn sẽ thấy kết quả sau:

```
[0]
Không bị tiểu đường
```

Chúng tôi cũng muốn biết các xác suất của dự đoán và do đó, bạn có được các xác suất và chuyển chúng thành tỷ lệ phần trăm:

```
proba = loading_model.pionary_proba ([[Glucose, BMI, Age]]) print (proba)

print ("Độ tin cậy:" + str (round (np.amax (proba [0]) * 100, 2)) + "%")
```

Bạn sẽ thấy những điều sau:

```
[[0,94736842 0,05263158]] Độ tin
cậy: 94,74%
```

Xác suất được in hiển thị xác suất của kết quả là 0 và xác suất của kết quả là 1. Dự đoán dựa trên dự đoán có xác suất cao nhất và chúng tôi sử dụng xác suất đó và chuyển đổi nó thành tỷ lệ phần trăm kết quả.

Triển khai mô hình

Bây giờ đã đến lúc triển khai mô hình học máy của chúng tôi dưới dạng API REST. Tuy nhiên, trước tiên, bạn cần cài đặtBình giữ nhiệtkhuôn khổ vi mô.

TIỀN BOA Bình giữ nhiệtlà một khuôn khổ vi mô cho Python cho phép bạn xây dựng dựa trên web các ứng dụng. Các khuôn khổ vi mô trong Python có ít hoặc không phụ thuộc vào các thư viện bên ngoài và do đó rất nhẹ. Flask đặc biệt hữu ích để phát triển các API REST. Để biết thêm thông tin về Flask, hãy xem tài liệu của nó tạihttp: // bình. pocoo.org/ docs/1.0/.

Nhập dòng lệnh sau vào Terminal hoặc Command Prompt để cài đặt Flask:

```
$ pip cài đặt bình
```

Sau khi cài đặt Flask, hãy tạo một văn bản có tênREST _ API.py,và nhập đoạn mã sau:

```
nhập khẩu dưa chua
from flask import Flask, request, json, jsonify import numpy as
np

app = Flask (__ name__)

# - - - tên tệp của mô hình đã lưu --- filename
='etes.sav '

# - - - tải mô hình đã lưu ---
Load_model = pickle.load (mở (tên tệp, 'rb'))

@ app.route ('/etes / v1 / dự đoán', phương pháp = ['POST']) def dự đoán
():
    # - - - nhận các tính năng để dự đoán ---
    features = request.json

    # - - - tạo danh sách tính năng để dự đoán --- features_list =
    [features ["Glucose"],
                        tính năng ["BMI"],
                        tính năng ["Tuổi"]]

    # - - - nhận lớp dự đoán ---
    dự đoán = loading_model.p Dự đoán ([features_list])

    # - - - lấy xác suất dự đoán ---
    niềm tin = loading_model.posystem_proba ([features_list])

    # - - - lập công thức phản hồi để trả về ứng dụng khách --- response
    = {}
    đáp ứng ['dự đoán'] = int (dự đoán [0]) phản hồi ['độ tin cậy'] = str (vòng (np.amax (độ tin cậy
    [0]) * 100, 2))

    return jsonify (phản hồi)

nếu __name__ == '__main__':
    app.run (host = '0.0.0.0', port = 5000)
```

Đoạn mã trước hoàn thành những điều sau:

- Tạo một tuyến đường /bệnh tiểu đường / v1 / dự đoánsử dụngtuyến đườngngười trang
- trí. Lộ trình có thể truy cập thông qua động từ POST.

- Để đưa ra dự đoán, người dùng thực hiện cuộc gọi đến tuyến đường này và chuyển các tính năng khác nhau bằng chuỗi JSON.
- Kết quả của dự đoán được trả về dưới dạng chuỗi JSON.

GHI CHÚ Mộtngười trang trítrong Python là một hàm bao bọc và thay thế một hàm khác.

Kiểm tra mô hình

Để kiểm tra REST API, hãy chạy nó trong Terminal bằng cách nhập lệnh sau:

```
$ python REST_API.py
```

Bạn sẽ thấy kết quả sau:

```
* Đang cung cấp ứng dụng Flask "REST_API" (tải chậm)
 * Môi trường: sản xuất
   CẢNH BÁO: Không sử dụng máy chủ phát triển trong môi trường sản xuất. Thay vào đó, hãy sử
   dụng máy chủ WSGI sản xuất.
 * Chế độ gỡ lỗi: tắt
 * Chạy trên http://0.0.0.0:5000/ (Nhấn CTRL + C để thoát)
```

Điều này cho thấy rằng dịch vụ đang hoạt động và đang nghe ở cổng 5000.

Cách dễ nhất để kiểm tra API là sử dụng lệnh cURL (được cài đặt theo mặc định trên macOS) từ cửa sổ Terminal hoặc Command Prompt riêng biệt:

```
$ curl -H "Content-type: application / json" -X ĐĂNG http://
127.0.0.1:5000/diabetes/v1/p Dự đoán
- d '{"BMI": 30, "Tuổi": 29, "Glucose": 100}'
```

Lệnh trước đặt tiêu đề JSON và nó sử dụngBƯU KIỆNđộng từ để kết nối với API REST đang lắng nghe ở cổng 5000. Các tính năng và giá trị của chúng để sử dụng cho dự đoán được gửi dưới dạng chuỗi JSON.

TIỀN BOA Đối với người dùng Windows, lệnh cURL không nhận dạng được các dấu ngoặc kép. Bạn phải sử dụng dấu ngoặc kép và tắt ý nghĩa đặc biệt của dấu ngoặc kép trong chuỗi JSON: "{\"BMI \ ": 30, \" Tuổi \ ": 29, \" Glucose \ ": 100}".

Khi REST API đã nhận được dữ liệu được gửi đến nó, nó sẽ sử dụng nó để thực hiện dự đoán. Bạn sẽ thấy kết quả dự đoán trả về như sau:

```
{"độ tin cậy": "78,95", "dự đoán": 0}
```

Kết quả chỉ ra rằng dựa trên dữ liệu được gửi đến, không có khả năng người đó mắc bệnh tiểu đường (78,95% ý kiến).

Hãy tiếp tục và thử một số giá trị khác, như sau:

```
$ curl -H "Content-type: application / json" -X ĐĂNG http://
127.0.0.1:5000/diabetes/v1/p Dự đoán
- d '{"BMI": 65, "Tuổi": 29, "Glucose": 150}'
```

Lần này, dự đoán cho thấy người đó có khả năng mắc bệnh tiểu đường với 68,42% do:

```
{"độ tin cậy": "68,42", "dự đoán": 1}
```

Tạo ứng dụng khách để sử dụng mô hình

Sau khi API REST được thiết lập và chạy và nó đã được kiểm tra để hoạt động chính xác, bạn có thể xây dựng phía máy khách của mọi thứ. Vì cuốn sách này xoay quanh Python, nên nó đang cố gắng xây dựng ứng dụng khách bằng Python. Rõ ràng, trong cuộc sống thực, rất có thể bạn sẽ xây dựng khách hàng của mình cho các nền tảng iOS, Android, macOS và Windows.

Ứng dụng khách Python của chúng tôi khá đơn giản — xây dựng chuỗi JSON để gửi đến dịch vụ, lấy lại kết quả trong JSON, sau đó truy xuất chi tiết của kết quả:

```
nhập json
yêu cầu nhập khẩu

def dự đoán_đái đường (BMI, Tuổi, Glucose):
    url = 'http://127.0.0.1:5000/diabetes/v1/p Dự đoán' data = {"BMI":
    BMI, "Age": Tuổi, "Glucose": Glucose} data_json = json.dumps (dữ
    liệu)
    headers = {'Content-type': 'application / json'}
    response = request.post (url, data = data_json, headers = headers) kết quả
            = json.loads (response.text) kết quả
    trở về

nếu __name__ == "__main__":
    dự đoán = dự đoán_diabetes (30,40,100) print ("Bệnh tiểu đường" nếu dự đoán
    ["dự đoán"] == 1 khác "Không phải bệnh tiểu đường")

    print ("Độ tin cậy:" + dự đoán ["độ tin cậy"] + "%")
```

Chạy điều này trong Máy tính xách tay Jupyter sẽ thu được kết quả sau:

```
Không bị tiểu đường
Độ tin cậy: 68,42%
```

Hãy lưu đoạn mã trước đó thành le và thêm mã để cho phép người dùng nhập các giá trị khác nhau cho BMI, Tuổi và Glucose. Lưu đoạn mã sau trong một le có tênDự đoán _ Diabetes.py:

```
nhập json
yêu cầu nhập khẩu

def dự đoán_đái đường (BMI, Tuổi, Glucose):
    url = 'http://127.0.0.1:5000/diabetes/v1/p Dự đoán'
```

```
        data = {"BMI": BMI, "Age": Tuổi, "Glucose": Glucose} data_json =
        json.dumps (dữ liệu)
        headers = {'Content-type': 'application / json'}
        response = request.post (url, data = data_json, headers = headers) kết quả
                 = json.loads (response.text) kết quả
        trở về

nếu __name__ == "__main__":
        BMI  =  đầu vào ('BMI?')
        Tuổi tác  =  input ('Tuổi?')
        Glucose =      đầu vào ('Glucose?')
        phỏng đoán       = Pred_diabetes (BMI, Tuổi, Glucose)
        print ("Tiểu đường" nếu dự đoán ["dự đoán"] == 1 khác "Không phải Tiểu đường")

        print ("Độ tin cậy:" + dự đoán ["độ tin cậy"] + "%")
```

Bây giờ bạn có thể chạy ứng dụng trong Terminal:

```
$ python P Dự đoán_Diabetes.py
```

Bây giờ bạn có thể nhập các giá trị:

```
BMI?55
Tuổi tác?29
Đường glucôzơ?120
```

Kết quả bây giờ sẽ được hiển thị:

```
Không bị tiểu đường
Độ tin cậy: 52,63%
```

Bản tóm tắt

Trong chương cuối này, bạn đã biết cách triển khai mô hình học máy của mình bằng cách sử dụng khung vi mô Flask. Bạn cũng đã thấy cách bạn có thể xem mối tương quan giữa các tính năng khác nhau và sau đó chỉ sử dụng các tính năng hữu ích nhất đó để đào tạo mô hình của bạn. Việc đánh giá một số thuật toán học máy và chọn thuật toán hoạt động tốt nhất luôn hữu ích để bạn có thể chọn thuật toán chính xác cho tập dữ liệu cụ thể của mình.

Tôi hy vọng rằng cuốn sách này đã cung cấp cho bạn một cái nhìn tổng quan tốt về học máy và nó đã khởi đầu và truyền cảm hứng cho bạn tiếp tục học. Như tôi đã đề cập, cuốn sách này là một phần giới thiệu nhẹ nhàng về học máy và có một số chi tiết đã được cố ý bỏ qua để giúp bạn dễ dàng theo dõi. Tuy nhiên, nếu bạn đã thử tất cả các bài tập trong mỗi chương, bây giờ bạn sẽ hiểu khá tốt về các nguyên tắc cơ bản của học máy!

Mục lục

Python® Machine Learning, First Edition. Wei-Meng Lee.