**Bài 4: Supervised Learning: Classification phần 1 - Lập trình AI bằng Python**

Trong bài này, chúng ta sẽ tập trung vào việc học có giám sát **classifcation**.

Kỹ thuật phân loại hoặc mô hình cố gắng thu được một số kết luận từ các giá trị quan sát được. Trong bài toán phân loại, chúng tôi có đầu ra được phân loại như “Đen” hoặc “trắng” hoặc “Dạy” và “Không dạy”. Trong khi xây dựng mô hình phân loại, chúng ta cần có tập dữ liệu huấn luyện chứa các điểm dữ liệu và các nhãn tương ứng. Ví dụ, nếu chúng ta muốn kiểm tra xem hình ảnh có phải của một chiếc ô tô hay không. Để kiểm tra điều này, chúng tôi sẽ xây dựng một tập dữ liệu đào tạo có hai lớp liên quan đến “ô tô” và “không có ô tô”. Sau đó, chúng ta cần đào tạo mô hình bằng cách sử dụng các mẫu đào tạo. Các mô hình phân loại chủ yếu được sử dụng trong nhận dạng khuôn mặt, nhận dạng thư rác, v.v.

**1. Các bước xây dựng Classifier trong Python :**

Để xây dựng bộ phân loại bằng Python, chúng ta sẽ sử dụng Python 3 và Scikit-learning là một công cụ dành cho học máy. Làm theo các bước sau để xây dựng bộ phân loại bằng Python -

**Bước 1** : Import Scikit-learn :

Đây sẽ là bước đầu tiên để xây dựng một bộ phân loại trong Python. Trong bước này, tai sẽ cài đặt một gói Python có tên là Scikit-learning, là một trong những mô-đun học máy tốt nhất trong Python.Cú pháp như sau:

**Import** Sklearn

**Bước 2**: Import Scikit-learn dataset:

Chúng ta có thể bắt đầu làm việc với tập dữ liệu cho mô hình học máy của mình. Ở đây, chúng tôi sẽ sử dụng Cơ sở dữ liệu chẩn đoán ung thư vú của Wisconsin. Bộ dữ liệu bao gồm nhiều thông tin khác nhau về các khối u ung thư vú, cũng như các nhãn phân loại ác tính hoặc lành tính. Tập dữ liệu có 569 trường hợp hoặc dữ liệu về 569 khối u và bao gồm thông tin về 30 thuộc tính hoặc đặc điểm, chẳng hạn như bán kính của khối u, kết cấu, độ mịn và diện tích. Với sự trợ giúp của lệnh sau, chúng tôi có thể nhập bộ dữ liệu về bệnh ung thư vú của Scikit-learning -

from sklearn.datasets **import** load\_breast\_cancer

Sau đó bắt đầu load dữ liệu :

data = load\_breast\_cancer()

Một vài tham số quan trọng :

* Classification label names(target\_names)
* The actual labels(target)
* The attribute/feature names(feature\_names)
* The attribute (data)

label\_names = data['target\_names']

labels = data['target']

feature\_names = data['feature\_names']

features = data['data']

Ta hãy xem qua dữ liệu :

print(label\_names)

Lệnh trên sẽ in ra tên lớp tương ứng là ác tính và lành tính. Nó được hiển thị như đầu ra bên dưới:

['malignant' 'benign']

Bây giờ, lệnh dưới đây sẽ cho thấy rằng chúng được ánh xạ tới các giá trị nhị phân 0 và 1. Ở đây 0 đại diện cho ung thư ác tính và 1 đại diện cho ung thư lành tính. Bạn sẽ nhận được kết quả sau:

print(labels[0])

0

Hai lệnh dưới đây sẽ tạo ra tên đối tượng và giá trị đối tượng.

print(feature\_names[0])

mean radius

print(features[0])

[ 1.79900000e+01 1.03800000e+01 1.22800000e+02 1.00100000e+03

1.18400000e-01 2.77600000e-01 3.00100000e-01 1.47100000e-01

2.41900000e-01 7.87100000e-02 1.09500000e+00 9.05300000e-01

8.58900000e+00 1.53400000e+02 6.39900000e-03 4.90400000e-02

5.37300000e-02 1.58700000e-02 3.00300000e-02 6.19300000e-03

2.53800000e+01 1.73300000e+01 1.84600000e+02 2.01900000e+03

1.62200000e-01 6.65600000e-01 7.11900000e-01 2.65400000e-01

4.60100000e-01 1.18900000e-01]

Từ kết quả trên, chúng ta có thể thấy rằng trường hợp dữ liệu đầu tiên là một khối u ác tính có bán kính là 1.7990000e + 01.

**Bước 3 :** Tổ chức dữ liệu thành tập hợp

Trong bước này, chúng ta sẽ chia dữ liệu của mình thành hai phần cụ thể là tập huấn luyện và tập kiểm tra. Việc chia nhỏ dữ liệu thành các tập hợp này là rất quan trọng vì chúng ta phải kiểm tra mô hình của mình trên dữ liệu không nhìn thấy. Để chia dữ liệu thành các tập hợp, sklearn có một hàm gọi là hàm train\_test\_split (). Sử dụng các lệnh sau, ta có thể chia dữ liệu trong các tập hợp này:

from sklearn.model\_selection **import** train\_test\_split

Lệnh trên sẽ nhập hàm train\_test\_split từ sklearn và lệnh dưới sẽ chia dữ liệu thành dữ liệu huấn luyện và kiểm tra. Trong ví dụ dưới đây, chúng tôi đang sử dụng 40% dữ liệu để thử nghiệm và dữ liệu còn lại sẽ được sử dụng để đào tạo mô hình.

train, test, train\_labels, test\_labels = train\_test\_split(features,labels,test\_size = 0.40, random\_state = 42)

**Bước 4**: Xây dựng model :

Trong bước này, chúng tôi sẽ xây dựng mô hình của mình. Chúng tôi sẽ sử dụng thuật toán Naïve Bayes để xây dựng mô hình. Các lệnh sau có thể được sử dụng để xây dựng mô hình:

from sklearn.naive\_bayes **import** GaussianNB

Lệnh trên sẽ nhập mô-đun GaussianNB. Bây giờ, lệnh sau sẽ giúp bạn khởi tạo mô hình.

gnb = GaussianNB()

Chúng tôi sẽ đào tạo mô hình bằng cách điều chỉnh nó với dữ liệu bằng cách sử dụng gnb.fit ().

model = gnb.fit(train, train\_labels)

**Bước 5**: Đánh giá mô hình và độ chính xác của mô hình :

Trong bước này, chúng tôi sẽ đánh giá mô hình bằng cách đưa ra các dự đoán trên dữ liệu thử nghiệm của chúng tôi. Sau đó, chúng tôi cũng sẽ tìm ra độ chính xác của nó. Để đưa ra dự đoán, chúng tôi sẽ sử dụng hàm dự đoán (). Lệnh sau sẽ giúp bạn làm điều này:

preds = gnb.predict(test)

print(preds)

[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1

0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0

0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0

0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0

1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0

1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0

1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0

1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 0 1]

Dãy số 0 và 1 ở trên là giá trị dự đoán cho các lớp khối u - ác tính và lành tính.

Bây giờ, bằng cách so sánh hai mảng là test\_labels và preds, chúng ta có thể tìm ra độ chính xác của mô hình của mình. Chúng ta sẽ sử dụng hàm precision\_score () để xác định độ chính xác. Hãy xem xét lệnh sau cho điều này:

from sklearn.metrics **import** accuracy\_score

print(accuracy\_score(test\_labels,preds))

0.951754385965

Kết quả cho thấy trình phân loại NaïveBayes chính xác đến 95,17%.

Bằng cách này, với sự trợ giúp của các bước trên, chúng ta có thể xây dựng bộ phân loại của mình bằng Python.

**2. Xây dựng Classifier bằng python :**

Ở mục này, chúng ta cùng tìm hiểu cách để xây dựng 1 classifier bằng Python :

**Naïve Bayes Classifier :**

Naïve Bayes là một kỹ thuật phân loại được sử dụng để xây dựng bộ phân loại bằng cách sử dụng định lý Bayes. Giả định là các yếu tố dự đoán là độc lập. Nói một cách đơn giản, nó giả định rằng sự hiện diện của một đối tượng cụ thể trong một lớp không liên quan đến sự hiện diện của bất kỳ đối tượng địa lý nào khác. Để xây dựng trình phân loại Naïve Bayes, chúng ta cần sử dụng thư viện python được gọi là scikit learning. Có ba loại mô hình Naïve Bayes có tên Gaussian, Multinomial và Bernoulli trong gói học scikit.

Để xây dựng mô hình phân loại học máy Naïve Bayes, chúng ta cần

**Dataset**

Chúng tôi sẽ sử dụng tập dữ liệu có tên Cơ sở dữ liệu chẩn đoán ung thư vú Wisconsin. Bộ dữ liệu bao gồm nhiều thông tin khác nhau về các khối u ung thư vú, cũng như các nhãn phân loại ác tính hoặc lành tính. Tập dữ liệu có 569 trường hợp hoặc dữ liệu về 569 khối u và bao gồm thông tin về 30 thuộc tính hoặc đặc điểm, chẳng hạn như bán kính của khối u, kết cấu, độ mịn và diện tích. Chúng tôi có thể nhập tập dữ liệu này từ gói sklearn.

**Naïve Bayes Model**

Để xây dựng bộ phân loại Naïve Bayes, chúng ta cần một mô hình Naïve Bayes. Như đã nói trước đó, có ba loại mô hình Naïve Bayes được đặt tên là Gaussian, Multinomial và Bernoulli trong gói học scikit. Ở đây, trong ví dụ sau, chúng ta sẽ sử dụng mô hình Gaussian Naïve Bayes.

Bằng cách sử dụng những điều trên, chúng tôi sẽ xây dựng một mô hình học máy Naïve Bayes để sử dụng thông tin về khối u để dự đoán khối u là ác tính hay lành tính.

Để bắt đầu, chúng ta cần cài đặt mô-đun sklearn. Nó có thể được thực hiện với sự trợ giúp của lệnh sau:

**Import** Sklearn

Ta cần import dataset tên là Breast Cancer Wisconsin Diagnostic Database.

from sklearn.datasets **import** load\_breast\_cancer

Sau đó ta load dữ liệu từ dataset :

data = load\_breast\_cancer()

label\_names = data['target\_names']

labels = data['target']

feature\_names = data['feature\_names']

features = data['data']

Ta thử in giá trị label\_name :

print(label\_names)

Kết quả như sau ;

['malignant' 'benign']

Bây giờ, lệnh đưa ra dưới đây sẽ cho thấy rằng chúng được ánh xạ tới các giá trị nhị phân 0 và 1. Ở đây 0 đại diện cho ung thư ác tính và 1 đại diện cho ung thư lành tính. Nó được hiển thị như đầu ra bên dưới:

print(labels[0])

0

Hai lệnh sau sẽ tạo ra tên đối tượng và giá trị đối tượng.

print(feature\_names[0])

mean radius

print(features[0])

[ 1.79900000e+01 1.03800000e+01 1.22800000e+02 1.00100000e+03

1.18400000e-01 2.77600000e-01 3.00100000e-01 1.47100000e-01

2.41900000e-01 7.87100000e-02 1.09500000e+00 9.05300000e-01

8.58900000e+00 1.53400000e+02 6.39900000e-03 4.90400000e-02

5.37300000e-02 1.58700000e-02 3.00300000e-02 6.19300000e-03

2.53800000e+01 1.73300000e+01 1.84600000e+02 2.01900000e+03

1.62200000e-01 6.65600000e-01 7.11900000e-01 2.65400000e-01

4.60100000e-01 1.18900000e-01]

Từ kết quả trên, chúng ta có thể thấy rằng trường hợp dữ liệu đầu tiên là một khối u ác tính, bán kính chính của nó là 1.7990000e + 01.

Để thử nghiệm mô hình của chúng tôi trên dữ liệu không nhìn thấy, chúng tôi cần chia dữ liệu của mình thành dữ liệu đào tạo và thử nghiệm. Nó có thể được thực hiện với sự trợ giúp của đoạn mã sau:

from sklearn.model\_selection **import** train\_test\_split

Lệnh trên sẽ nhập hàm train\_test\_split từ sklearn và lệnh dưới sẽ chia dữ liệu thành dữ liệu huấn luyện và kiểm tra. Trong ví dụ dưới đây, chúng tôi đang sử dụng 40% dữ liệu để thử nghiệm và dữ liệu nhắc nhở sẽ được sử dụng để đào tạo mô hình.

train, test, train\_labels, test\_labels =

train\_test\_split(features,labels,test\_size = 0.40, random\_state = 42)

Chúng ta cùng xây dựng mô hình sau :

from sklearn.naive\_bayes **import** GaussianNB

Lệnh trên sẽ nhập mô-đun GaussianNB. Bây giờ, với lệnh dưới đây, chúng ta cần khởi tạo mô hình.

gnb = GaussianNB()

Chúng tôi sẽ đào tạo mô hình bằng cách điều chỉnh nó với dữ liệu bằng cách sử dụng gnb.fit ().

model = gnb.fit(train, train\_labels)

Bây giờ, đánh giá mô hình bằng cách đưa ra dự đoán trên dữ liệu thử nghiệm và nó có thể được thực hiện như sau:

preds = gnb.predict(test)

print(preds)

[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1

0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0

0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0

0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0

1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0

1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0

1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0

1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 0 1]

Dãy số 0 và 1 ở trên là các giá trị dự đoán cho các lớp khối u, tức là ác tính và lành tính.

Bây giờ, bằng cách so sánh hai mảng là test\_labels và preds, chúng ta có thể tìm ra độ chính xác của mô hình của mình. Chúng ta sẽ sử dụng hàm precision\_score () để xác định độ chính xác.

from sklearn.metrics **import** accuracy\_score

print(accuracy\_score(test\_labels,preds))

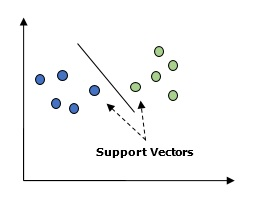
0.951754385965

Kết quả cho thấy trình phân loại NaïveBayes chính xác đến 95,17%.

Đó là trình phân loại học máy dựa trên mô hình Naïve Bayse Gaussian.

**3. Support Vector Machines (SVM) :**

Về cơ bản, Support vector machine (SVM) là một thuật toán học máy được giám sát có thể được sử dụng cho cả hồi quy và phân loại. Khái niệm chính của SVM là vẽ mỗi mục dữ liệu dưới dạng một điểm trong không gian n chiều với giá trị của mỗi đối tượng là giá trị của một tọa độ cụ thể. Đây n sẽ là các tính năng chúng tôi sẽ có. Sau đây là một biểu diễn đồ họa đơn giản để hiểu khái niệm về SVM -



Trong sơ đồ trên, chia ra thành 2 vùng dữ liệu. Do đó, trước tiên chúng ta cần vẽ hai biến này trong không gian hai chiều nơi mỗi điểm có hai tọa độ, được gọi là vectơ hỗ trợ. Dòng chia dữ liệu thành hai nhóm được phân loại khác nhau. Dòng này sẽ là bộ phân loại.

Ở đây, chúng ta sẽ xây dựng một bộ phân loại SVM bằng cách sử dụng tập dữ liệu scikit-learning và iris. Thư viện Scikitlearn có mô-đun sklearn.svm và cung cấp sklearn.svm.svc để phân loại. Bộ phân loại SVM để dự đoán loại cây diên vĩ dựa trên 4 đặc điểm được hiển thị bên dưới.

**a. Dataset :**

Ta sẽ sử dụng tập dữ liệu iris chứa 3 lớp, mỗi lớp gồm 50 trường hợp, trong đó mỗi lớp đề cập đến một loại cây diên vĩ. Mỗi cá thể có bốn đặc điểm là chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa và chiều rộng cánh hoa. Bộ phân loại SVM để dự đoán loại cây diên vĩ dựa trên 4 đặc điểm được hiển thị bên dưới.

**b. Kernel :**

Nó là một kỹ thuật được sử dụng bởi SVM. Về cơ bản đây là những hàm lấy không gian đầu vào có chiều thấp và biến đổi nó thành không gian có chiều cao hơn. Nó chuyển đổi vấn đề không thể phân tách thành vấn đề có thể phân tách. Hàm nhân có thể là một hàm bất kỳ trong số các hàm tuyến tính, đa thức, rbf và sigmoid. Trong ví dụ này, chúng ta sẽ sử dụng nhân tuyến tính.

**import** pandas as pd

**import** numpy as np

from sklearn **import** svm, datasets

**import** matplotlib.pyplot as plt

Load dữ liệu input :

iris = datasets.load\_iris()

X = iris.data[:, :2]

y = iris.target

Ta sẽ vẽ các ranh giới support vector machine với dữ liệu gốc. Chúng tôi đang tạo một lưới để lập mưu.

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

h = (x\_max / x\_min)/100

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

X\_plot = np.c\_[xx.ravel(), yy.ravel()]

C = 1.0

Tạo SVM classifier :

Svc\_classifier = svm\_classifier.SVC(kernel='linear',

C=C, decision\_function\_shape = 'ovr').fit(X, y)

Z = svc\_classifier.predict(X\_plot)

Z = Z.reshape(xx.shape)

plt.figure(figsize = (15, 5))

plt.subplot(121)

plt.contourf(xx, yy, Z, cmap = plt.cm.tab10, alpha = 0.3)

plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Set1)

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())

plt.title('SVC with linear kernel')

