1. Train test split using sklearn in python

-Train\_test\_split() method được sử dụng để split data thành train set và test set.

- First, cần chia data thành features(X) và labels(Y). Khung data được chia thành X\_train,X\_test và Y\_train, Y\_test.

-X\_train và Y\_train sets được sử dụng để training và fitting model.

- X\_test và Y\_test sets được sử dụng để testing model nếu nó predicting đúng ouputs/labels. => có thể kiểm tra chính xác size của train và test sets. Suggested to keep our train sets larger than the test sets.

Train set: this data is seen and learned by the model. (the training dataset).

Test set: this data is a subnet of the training dataset that is utilized to give an accurate evaluation of the final model fit.

Validation set: a validation dataset là 1 mẫu dữ liệu từ training dataset, được sử dụng để ước tính(estimate) hiệu suất của model khi điều chỉnh (tuning) các siêu tham số (hyperparameters) của model.

|  |
| --- |
| # import packages  **import** numpy as np  **import** pandas as pd  **from** sklearn.model\_selection **import** train\_test\_split    # importing data  df **=** pd.read\_csv('headbrain1.csv')    # head of the data  **print**(df.head())    X**=** df['Head Size(cm^3)']  y**=**df['Brain Weight(grams)']    # using the train test split function  X\_train, X\_test,  y\_train, y\_test **=** train\_test\_split(X,y ,                                     random\_state**=**104,                                     test\_size**=**0.25,                                     shuffle**=**True)    # printing out train and test sets    **print**('X\_train : ')  **print**(X\_train.head())  print('')  print('X\_test : ')  print(X\_test.head())  **print**('')  print('y\_train : ')  **print**(y\_train.head())  print('')  print('y\_test : ')  **print**(y\_test.head()) |

# X: chứa các đặc điểm, Y chứa các nhãn

# test\_size=0.25 nghĩa là 25% dữ liệu là test size => 1- test size = train size

# shuffle=True, xáo trộn dữ liệu trước khi tràn

# random\_state hoạt động giống như 1 hạt numpy, nó sử dụng để tái tạo dữ liệu.

# 2. Train-Test Split for Evaluating Machine Learning Algorithms

-train-test split procedure(quy trình, phương thức, thủ tục) được sử dụng để ước tính hiệu suất của thuật toán ML khi chúng được sử dụng để dự đoán trên tập data ko được sử dụng để train model.

- Đây là 1 phương thức nhanh chóng và dễ dàng thực hiện, kết quả của nó cho phép so sánh các hiệu suất của các thuật toán ML cho vấn đề mô hình dự đoán. Nhưng đôi khi ko nên sử dụng nếu tập dữ liệu nhỏ và các tình huống cần dữ liệu bổ sung, classification và tập dữ liệu không cân bằng.

- train-test split procedure thích hợp khi bạn có một tập dữ liệu rất lớn, model tốn kém để train hoặc yêu cầu ước tính tốt về hiệu suất model nhanh chóng.

## Tổng quan về Hướng dẫn

Hướng dẫn này được chia thành ba phần:

1. Đánh giá phân tách train-test
   1. Khi nào sử dụng phân tách train-test
   2. Cách định cấu hình phân tách train-test
2. Quy trình phân chia train-test trong Scikit-Learn
   1. Tách train-test lặp lại
   2. Phân tầng train-test
3. Phân tách train-test để đánh giá mô hình học máy
   1. Tách train-test để phân loại (classification)
   2. Tách Train-Test cho Hồi quy (regression)
4. Đánh giá phân tách Train-test

Phân tách train-test là một kĩ thuật để đánh giá hiệu suất của thuật toán học máy.

Nó có thể được sử dụng trong bài toán phân loại hoặc hồi quy(regression) và có thể sử dụng cho bất kì thuật toán học có giám sát.

Quá trình liên quan đến việc lấy một tập dữ liệu và phân chia nó thành 2 tập con. Tập con đầu tiên được sử dụng để phù hợp với model và được gọi là tập huấn luyện dữ liệu(training dataset). Tập con thứ 2 ko được dùng để huấn luyện model, thay vào đó,phần tử đầu vào của tập dữ liệu được cung cấp cho model, sau đó các dự đoán được thực hiện và so sánh với các giá trị mong đợi. Tập dữ liệu thứ 2 được gọi là tập dữ liệu thử nghiệm(test dataset)

* **Train Dataset**: Used to fit the machine learning model.
* **Test Dataset**: Used to evaluate the fit machine learning model.

Mục tiêu là ước tính hiệu suất của mô hình học máy trên dữ liệu mới: dữ liệu không được sử dụng để huấn luyện mô hình.

* 1. Khi nào sử dụng phân tách train-test
* Dữ liệu đủ để phân tách tập dữ liệu thành 2 tập con và mỗi tập dữ liệu là các dại diện phù hợp của miền vấn đề => tập dữ liệu gốc cũng phải là 1 đại diện phù hợp của miền vấn đề.
* 1 đại diện phù hợp của miền vấn đề có nghĩa là có đủ bản ghi để bao gồm tất cả các trường hợp phổ biến và các trường hợp ko phổ biến nhất trong miền. Sự kết hợp của các biến đầu vào được quan sát trong thực tế. Nó có thể yêu cầu nhiều hoặc rất nhiều ví dụ.
* Quy trình train-test ko thích hợp khi dữ liệu là nhỏ => ko đủ dữ liệu để mô hình học ánh xạ hiệu quả từ đầu vào đến đầu ra=> ko đủ dữ liệu để đánh giá hiệu quả => hiệu suất ước tính có thể ko như mong đợi.
* Ngoài lí do trên còn có một số model rất tốn kém để huấn luyện, và trong trường hợp đó, việc đánh giá lặp đi lặp lại trong các quy trình khác là ko thể thực hiện được.
* Mặt khác, một dự án có thể có một mô hình hiệu quả và một tập dữ liệu rộng lớn, mặc dù có thể yêu cầu ước tính hiệu suất mô hình một cách nhanh chóng. Một lần nữa, thủ tục tách train-test được tiếp cận trong tình huống này.
  1. Cách định cấu hình phân tách train-test

Quy trình có một tham số cấu hình chính, đó là kích thước của train set và test set. Được biểu thị bằng phần tra từ 0->1

Ko có tỉ lệ phân chia tối ưu

Bạn phải chọn một tỷ lệ phần trăm nhỏ đáp ứng các mục tiêu của dự án của bạn với các cân nhắc bao gồm:

* Chi phí tính toán trong đào tạo mô hình.
* Chi phí tính toán trong đánh giá mô hình.
* Đào tạo thiết lập tính đại diện.
* Kiểm tra tính đại diện.

Tuy nhiên, tỷ lệ phân chia phổ biến bao gồm:

* Đào tạo: 80%, Kiểm tra: 20%
* Đào tạo: 67%, Kiểm tra: 33%
* Đào tạo: 50%, Kiểm tra: 50%

1. Quy trình phân tách train-test trong python sử dụng Scikit-learn

Kích thước của phần tách có thể được chỉ định thông qua đối số “ test\_size ” lấy một số hàng (số nguyên) hoặc một tỷ lệ phần trăm (float) của kích thước tập dữ liệu từ 0 đến 1.

Chạy ví dụ này sẽ chia tập dữ liệu thành các tập huấn luyện và thử nghiệm, sau đó in ra kích thước của tập dữ liệu mới.

Code:

***# split a dataset into train and test sets***

***from sklearn.datasets import make\_blobs***

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

***# create dataset***

***X, y = make\_blobs(n\_samples=1000) # 1000 sample***

***# split into train test sets***

***X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33)***

***print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)***

ouput: (670, 2) (330, 2) (670,) (330,)

Chúng ta có thể thấy rằng 670 ví dụ (67 phần trăm) đã được phân bổ cho tập huấn luyện và 330 ví dụ (33 phần trăm) đã được phân bổ cho tập thử nghiệm

* 1. Tách train-test lặp lại

Một lưu ý quan trọng khác là các hàng được chỉ định cho tập train và tập test một cách ngẫu nhiên.

Điều này được thực hiện để đảm bảo rằng tập dữ liệu là một mẫu đại diện (ví dụ: mẫu ngẫu nhiên) của tập dữ liệu gốc, do đó, phải là một mẫu đại diện cho các quan sát từ miền vấn đề.

Khi so sánh các thuật toán học máy, điều mong muốn (có lẽ là bắt buộc) là chúng phải phù hợp và được đánh giá trên cùng một tập con của tập dữ liệu.

Điều này có thể đạt được bằng cách đặt “ random\_state ” thành một giá trị số nguyên. Bất kỳ giá trị nào sẽ làm; nó không phải là một siêu thông số có thể điều chỉnh được. random\_state là đối tượng điều khiển sử ngẫu nhiên trong quá trình tách

* 1. Phân tầng train-test

Một cân nhắc chỉ dành cho vấn đề phân loại

Một số bài toán phân loại ko có số lượng ví dụ cân bằng cho mỗi nhãn lớp. Do đó, mong muốn chia tập dữ liệu thành các tập huấn luyện và thử nghiệm theo cách duy trì cùng tỉ lệ các ví dụ trong mỗi lớp như được quan sát trong dữ liệu gốc.

Có thể đạt được điều đó bằng cách đặt đối số cho stratify = y

Chúng ta có thể chứng minh điều này bằng một ví dụ về tập dữ liệu phân loại với 94 ví dụ trong một lớp và sáu ví dụ trong lớp thứ hai.

Đầu tiên, chúng ta có thể chia tập dữ liệu thành các tập huấn luyện và thử nghiệm mà không cần đối số " *phân tầng* ". Ví dụ đầy đủ được liệt kê dưới đây.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | # split imbalanced dataset into train and test sets without stratification  from collections import Counter  from sklearn.datasets import make\_classification  from sklearn.model\_selection import train\_test\_split  # create dataset  X, y = make\_classification(n\_samples=100, weights=[0.94], flip\_y=0, random\_state=1)  print(Counter(y))  # split into train test sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50, random\_state=1)  print(Counter(y\_train))  print(Counter(y\_test)) |

Chạy ví dụ trước tiên báo cáo thành phần của tập dữ liệu theo nhãn lớp, hiển thị 94 phần trăm dự kiến ​​so với 6 phần trăm.

Sau đó, tập dữ liệu được tách ra và thành phần của nhóm tàu ​​và thử nghiệm được báo cáo. Ta có thể thấy tập xe lửa có 45/5 ví dụ trong tập kiểm tra có 49/1 ví dụ. Thành phần của đoàn tàu và bộ thử nghiệm khác nhau, và điều này là không mong muốn.

|  |  |  |
| --- | --- | --- |
| 1  2  3 | Counter({0: 94, 1: 6})  Counter({0: 45, 1: 5})  Counter({0: 49, 1: 1}) |  |

Tiếp theo, chúng tôi có thể phân tầng phân chia thử nghiệm tàu ​​và so sánh kết quả.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | # split imbalanced dataset into train and test sets with stratification  from collections import Counter  from sklearn.datasets import make\_classification  from sklearn.model\_selection import train\_test\_split  # create dataset  X, y = make\_classification(n\_samples=100, weights=[0.94], flip\_y=0, random\_state=1)  print(Counter(y))  # split into train test sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50, random\_state=1, stratify=y)  print(Counter(y\_train))  print(Counter(y\_test)) |

Giả sử rằng chúng tôi đã sử dụng mức phân chia 50% cho bộ tàu và bộ thử nghiệm, chúng tôi hy vọng rằng cả đoàn tàu và bộ thử nghiệm sẽ có 47/3 ví dụ tương ứng trong bộ tàu / thử nghiệm.

Chạy ví dụ này, chúng ta có thể thấy rằng trong trường hợp này, phiên bản phân tầng của phân tách thử nghiệm tàu ​​đã tạo ra cả tập dữ liệu tàu và thử nghiệm với 47/3 ví dụ trong tập tàu / thử nghiệm như chúng tôi mong đợi.

|  |  |
| --- | --- |
| 1  2  3 | Counter({0: 94, 1: 6})  Counter({0: 47, 1: 3})  Counter({0: 47, 1: 3}) |

Bây giờ chúng ta đã quen thuộc với hàm *train\_test\_split ()* , hãy xem cách chúng ta có thể sử dụng nó để đánh giá một mô hình học máy.

1. Tách train-test để đánh giá mô hình học máy
2. Phân tách train-test cho phân lớp:

Đánh giá trên tập dữ liệu sonar

from pandas import read\_csv

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# load dataset

url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar.csv'

dataframe = read\_csv(url, *header*=None)

data = dataframe.values

# split into inputs and outputs

X, y = data[:, :-1], data[:, -1]

print(X.shape, y.shape)

# split into train test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.33, *random\_state*=1)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

# fit the model

model = RandomForestClassifier(*random\_state*=1)

model.fit(X\_train, y\_train)

# make predictions

yhat = model.predict(X\_test)

# evaluate predictions

acc = accuracy\_score(y\_test, yhat)

print('Accuracy: %.3f' % acc)

output:

(208, 60) (208,)

(139, 60) (69, 60) (139,) (69,)

Accuracy: 0.783

ví dụ trước tiên sẽ tải tập dữ liệu và xác nhận số hàng trong các phần tử đầu vào và đầu ra.

1. Tách train-test cho regression(hồi quy)

Tập dữ liệu nhà ở là một tập dữ liệu học máy tiêu chuẩn bao gồm 506 hàng dữ liệu với 13 biến đầu vào số và một biến mục tiêu số.

from pandas import read\_csv

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

# load dataset

url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'

dataframe = read\_csv(url, *header*=None)

data = dataframe.values

# split into inputs and outputs

X, y = data[:, :-1], data[:, -1]

print(X.shape, y.shape)

# split into train test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.33, *random\_state*=1)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

# fit the model

model = RandomForestRegressor(*random\_state*=1)

model.fit(X\_train, y\_train)

# make predictions

yhat = model.predict(X\_test)

# evaluate predictions

mae = mean\_absolute\_error(y\_test, yhat)

print('MAE: %.3f' % mae)

ouput:

(506, 13) (506,)

(339, 13) (167, 13) (339,) (167,)

MAE: 2.171

**How to Split a Dataset Into Training and Testing Sets with Python**

3 cách tạo mẫu train-test:

* Sử dụng scikit-learn(aka sklearn)train\_test\_split()
* Sử dụng chức năng numpycủarandn()
* hoặc với pandasphương thức tích hợp được gọi làsample()

**Tại sao chúng ta cần đào tạo và thử nghiệm mẫu**

Một vấn đề rất phổ biến khi đào tạo model là quá khớp (overfitting). Hiện tượng này xảy ra khi một mô hình hoạt động quá tốt trên dữ liệu mà chúng ta đã sử để đào tạo nó nhưng nó không thể tổng quát hóa tốt cho các điểm dữ liệu mới.

Mặt khác, chưa khớp(underfitting) xảy ra khi model có hiệu suất kém ngay cả trên dữ liệu được sử dụng để đào tạo nó. Trong hầu hết các trường hợp underfitting xảy ra khi do model ko phù hợp với vấn đề bạn đang cố gắng giải quyết.

Tạo các mẫu dữ liệu khác nhau để đào tạo và kiểm tra mô hình là cách tiếp cận phổ biến nhất có thể được sử dụng để xác định các loại vấn đề này. Bằng cách này, chúng ta có thể sử dụng tập huấn luyện để huấn luyện mô hình của mình và sau đó coi tập thử nghiệm như một tập hợp các điểm dữ liệu sẽ giúp chúng ta đánh giá xem liệu mô hình có thể tổng quát hóa tốt cho dữ liệu mới, không nhìn thấy được hay không.

Cách đơn giản nhất để chia tập dữ liệu mô hình hóa thành các tập huấn luyện và kiểm tra là gán 2/3 điểm dữ liệu cho điểm trước và 1/3 còn lại cho điểm sau. Do đó, chúng tôi huấn luyện mô hình bằng cách sử dụng tập huấn luyện và sau đó áp dụng mô hình vào tập thử nghiệm.  Bằng cách này, chúng tôi có thể đánh giá hiệu suất của mô hình của chúng tôi. Ví dụ: nếu độ chính xác của quá trình huấn luyện là cực kỳ cao trong khi độ chính xác của thử nghiệm kém thì đây là một chỉ báo tốt cho thấy mô hình có thể đã bị overfitting.

Lưu ý rằng việc chia nhỏ tập dữ liệu thành các tập huấn luyện và thử nghiệm không phải là hành động duy nhất có thể được yêu cầu để tránh các hiện tượng như overfitting. Ví dụ: nếu cả tập huấn luyện và thử nghiệm đều chứa các mẫu không tồn tại trong dữ liệu thế giới thực thì mô hình sẽ vẫn có hiệu suất kém mặc dù chúng tôi không thể quan sát nó từ đánh giá hiệu suất.

Lưu ý thứ hai, bạn nên biết rằng có một số tình huống nhất định bạn nên xem xét việc tạo một tập hợp bổ sung được gọi là tập **hợp xác thực (validation set)** . Bộ xác nhận thường được yêu cầu khi ngoài hiệu suất của mô hình, chúng ta cũng cần chọn trong số nhiều mô hình và đánh giá mô hình nào hoạt động tốt hơn.

**Cách chia tập dữ liệu của chúng tôi thành các tập train-test**

Cách 1: Sử dụng pandas  
Sử dụng phương pháp dataFrames của pandas   
sample():  
 + Trả về một mẫu ngẫu nhiên của các mục từ một trục đối tượng.  
 + Có thể sử dụng random\_state để tái tạo dữ liệu.  
VD:  
 training\_data = df.sample(frac=0.8,random\_state=25) # frac giống như tỉ lệ chia train-test  
 testing\_data = df.drop(training\_data.index)  
 print(training\_data.shape[0]) #120  
 print(testing\_data.shape[0]) #30  
Cách 2: Sử dụng scikit-learn

Tùy chọn thứ hai - và có lẽ được sử dụng phổ biến nhất - là sử dụng sklearnphương thức của được gọi là [train\_test\_split()](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html):

*Chia mảng hoặc ma trận thành tập hợp con thử nghiệm và huấn luyện ngẫu nhiên*

Chúng tôi có thể tạo cả tập huấn luyện và thử nghiệm trong một lớp lót bằng cách chuyển đến train\_test\_split()DataFrame mô hình hóa cùng với phần nhỏ của các ví dụ cần có trong tập thử nghiệm. Như trước đây, chúng tôi cũng đặt một random\_stateđể các kết quả có thể lặp lại, tức là mỗi khi chúng tôi chạy mã, các phiên bản tương tự sẽ được đưa vào các tập huấn luyện và thử nghiệm tương ứng. Phương thức này trả về một bộ giá trị với hai DataFrames chứa các ví dụ đào tạo và thử nghiệm.

Cách 3: Sử dụng numpy

import numpy as np

mask = np.random.rand(len(df)) =< 0.8

training\_data = df[mask]

testing\_data = df[~mask]

print(f"No. of training examples: {training\_data.shape[0]}")

print(f"No. of testing examples: {testing\_data.shape[0]}")

# No. of training examples: 125

# No. of testing examples: 25

**Split Your Dataset With scikit-learn's train\_test\_split()**

* **shuffle**là [đối tượng Boolean](https://realpython.com/python-boolean/) ( Truetheo mặc định) xác định xem có xáo trộn tập dữ liệu trước khi áp dụng phân tách hay không.

II;  
Để đơn giản, chúng ta sẽ chủ yếu thảo luận về vấn đề phân loại nhị phân, nơi giả sử chúng ta sẽ phải tìm xem một hình ảnh là của một con mèo hay một con chó. Hoặc bệnh nhân đang bị ung thư (dương tính) hoặc khỏe mạnh (âm tính). Một số thuật ngữ phổ biến cần làm rõ là:

**True positives (TP)**: Predicted positive and are actually positive.

**False positives (FP)**: Predicted positive and are actually negative.

**True negatives (TN)**: Predicted negative and are actually negative.

**False negatives (FN)**: Predicted negative and are actually positive.

