1. **Split a set into training and testing sets.**
   1. **Tầm quan trọng của việc phân tách dữ liệu.**

Học máy có giám sát là việc tạo ra các mô hình ánh xạ chính xác các đầu vào đã cho (các biến độc lập hoặc các yếu tố dự đoán ) với các đầu ra nhất định (các biến phụ thuộc hoặc phản hồi ).

Cách đo độ chính xác của mô hình phụ thuộc vào loại vấn đề ta đang cố gắng giải quyết. Trong phân tích hồi quy , thường sử dụng hệ số xác định , sai số trung bình gốc , sai số tuyệt đối trung bình hoặc các đại lượng tương tự. Đối với các bài toán phân loại , thường áp dụng độ chính xác , độ chính xác, độ thu hồi , điểm F1 và các chỉ số liên quan.

Cần đánh giá khách quan để sử dụng đúng các biện pháp này, đánh giá hiệu suất dự đoán của mô hình của bạn và xác nhận mô hình. Điều này có nghĩa là không thể đánh giá hiệu suất dự đoán của một mô hình với cùng dữ liệu đã sử dụng để đào tạo mà cần đánh giá mô hình với dữ liệu mới mà mô hình chưa từng thấy trước đây. Do đó có thể thực hiện điều đó bằng cách chia nhỏ tập dữ liệu của mình trước khi sử dụng.

* 1. **Làm thế nào để phân chia tập dữ liệu.**

Phân tách một tập dữ liệu thành tập huấn luyện và tập thử nghiệm được sử dụng để ước tính hiệu suất của các thuật toán học máy có thể áp dụng cho các Ứng dụng / Thuật toán dựa trên dự đoán. Phương pháp này là một thủ tục nhanh chóng và dễ thực hiện để chúng ta có thể so sánh kết quả mô hình học máy của chính chúng ta với kết quả máy. Theo mặc định, tập thử nghiệm được chia thành 30% dữ liệu huấn luyện và tập thử nghiệm được chia thành 70% dữ liệu thực tế.

Chúng ta cần chia tập dữ liệu thành các tập huấn luyện và thử nghiệm để đánh giá mô hình học máy của chúng tôi hoạt động tốt như thế nào.

Tập huấn luyện được sử dụng để xây dựng mô hình.

Tập thử nghiệm chỉ được sử dụng cho các dự đoán.

* 1. **Khi nào sử dụng phân tách dữ liệu.**

Khi kích thước tập dữ liệu đủ lớn. Ngoài kích thước tập dữ liệu, một lý do khác để sử dụng quy trình đánh giá phân tách thử nghiệm đào tạo là hiệu quả tính toán.

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. Lúc này ta cần phân tách dữ liệu

Các mẫu từ tập dữ liệu đào tạo ban đầu được chia thành hai tập con bằng cách sử dụng lựa chọn ngẫu nhiên. Điều này nhằm đảm bảo rằng tập dữ liệu huấn luyện và kiểm tra là đại diện cho tập dữ liệu gốc.

* 1. **Cấu hình phân tách.**

Thông thường sẽ được biểu thị dưới dạng tỷ lệ phần trăm từ 0 đến 1 cho tập dữ liệu về huấn luyện hoặc thử nghiệm. Ví dụ: tập huấn luyện có kích thước 0,67 (67 phần trăm) có nghĩa là phần trăm còn lại 0,33 (33 phần trăm) được gán cho tập thử nghiệm.

Không có tỷ lệ phần trăm phân chia tối ưu.

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: 80%-20%, 67%-33%, 50%-50%.

* 1. **Sử dụng thư viện Sklearn để phân tách tập dữ liệu.**

Cài đặt skearn bằng pip install:

**python -m pip install -U "scikit-learn==0.23.1"**

Nhập khẩu hàm train\_test\_split() từ thư viện sklearn. Phương thức train\_test\_split() được sử dụng để chia dữ liệu của chúng ta thành các tập huấn luyện và thử nghiệm:

**import numpy as np**

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

Cú pháp:

**sklearn.model\_selection.train\_test\_split(\*arrays, \*\*options) -> list**

**arrays** là chuỗi danh sách , mảng NumPy , DataFrames pandas hoặc các đối tượng giống mảng tương tự chứa dữ liệu bạn muốn tách.

**Options** là các đối số từ khóa tùy chọn mà bạn có thể sử dụng để có được hành vi mong muốn:

**train\_size** là số xác định kích thước của tập huấn luyện. Nếu bạn cung cấp một float, thì nó phải nằm giữa 0.0và 1.0 và sẽ xác định phần chia sẻ của tập dữ liệu được sử dụng để thử nghiệm. Nếu bạn cung cấp một int, thì nó sẽ đại diện cho tổng số mẫu đào tạo. Giá trị mặc định là None.

**test\_size** là số xác định kích thước của tập kiểm tra. Nó rất giống với train\_size. Bạn nên cung cấp một trong hai train\_size hoặc test\_size. Nếu cả hai đều không được đưa ra, thì phần mặc định của tập dữ liệu sẽ được sử dụng để thử nghiệm là 25 %.

**random\_state** là đối tượng điều khiển sự ngẫu nhiên trong quá trình tách. Nó có thể là một int hoặc một trường hợp của RandomState. Giá trị mặc định là None.

**Shuffle** là đối tượng Boolean ( mặc định True) xác định xem có xáo trộn tập dữ liệu trước khi áp dụng phân tách hay không.

**Stratify** là một đối tượng giống mảng, nếu không None, sẽ xác định cách sử dụng phân chia phân tầng .

* 1. **Ví dụ.**

Code:

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.iloc[:, :-1] # all rows, all the features and no labels

y = df.iloc[:, -1] # all rows, label only

# using the 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,

                                train\_size=0.8,

                                  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())

Ta chia train, test = train\_test\_split(dataset,...)

Thành X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, ...)

Đầu vào (X) và đầu ra (y)

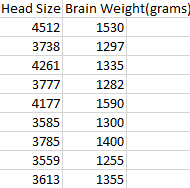
x\_train: Phần huấn luyện của dãy đầu tiên (x)

x\_test: Phần kiểm tra của dãy số đầu tiên (x)

y\_train: Phần huấn luyện của dãy thứ hai (y)

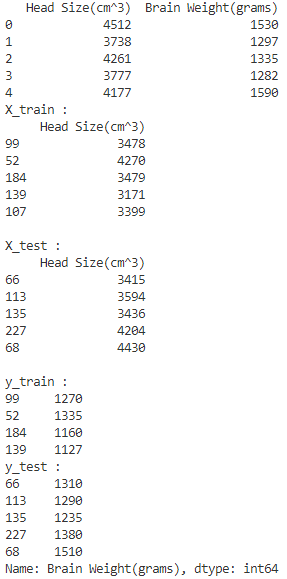
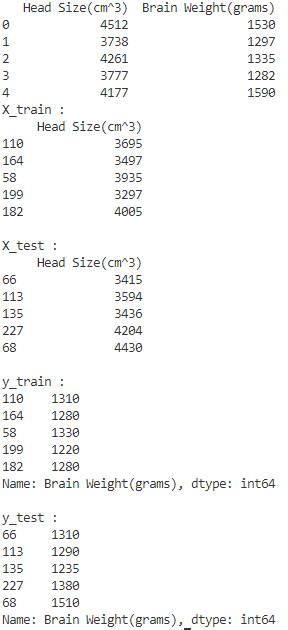
y\_test: Phần kiểm tra của dãy thứ hai (y)

File csv có dạng như sau:

****

Trong ví dụ trên ta sử dụng test\_size = 0.25 và đã xáo trộn dữ liệu đầu vào. Tương tự ta cũng có thể sử dụng train\_size = 0.8 thay vì sử dụng test\_size.

Đầu ra cho bên trái cho test\_size = 0.25 và đầu ra bên phải cho train\_size = 0.8:

Ví dụ 2:

Code:

import numpy as np

import pandas as pd

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

df = pd.read\_csv('./Real-estate.csv')

# get the location

X = df.iloc[:, :-1] # all rows, all the features and no labels

y = df.iloc[:, -1] # all rows, label only

# split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# print out the training and testing sets

print('X\_train : ')

print(X\_train.head())

print('')

print('X\_test : ')

print(X\_test.head())

print('')

print('y\_train : ')

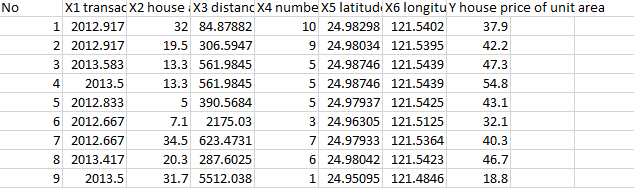
print(y\_train.head())

print('')

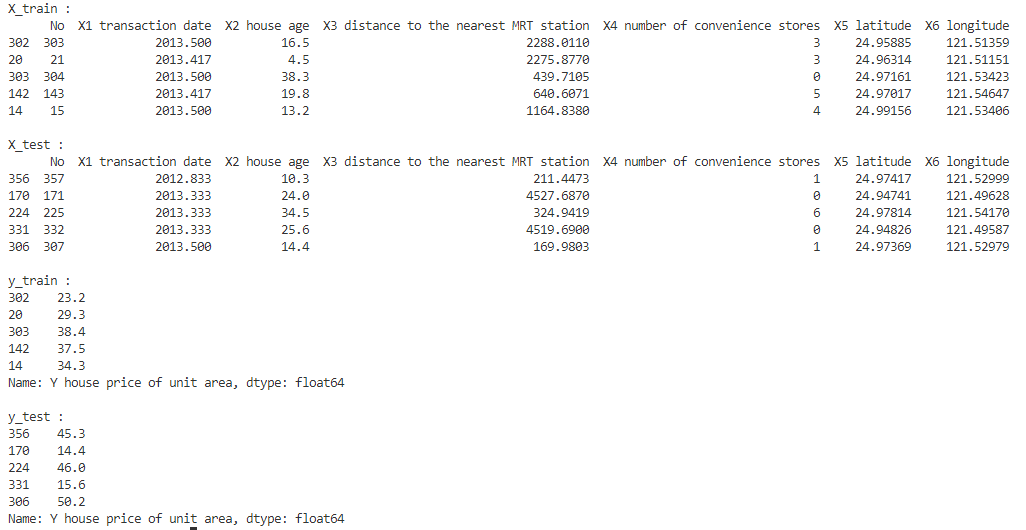
print('y\_test : ')

print(y\_test.head())

File csv:



Kết quả với test\_size = 0.2:



Ví dụ 3:Phân tầng dữ liệu

Code:

# 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=[0.5], 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=0.30, random\_state=1, stratify=y)

print(Counter(y\_train))

print(Counter(y\_test))

Kết quả:



Ví dụ 4: Phân tách để đánh giá mô hình học máy

Code:

# summarize the sonar dataset

# train-test split evaluation random forest on the sonar dataset

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

# xác định và phù hợp với mô hình trên tập dữ liệu đào tạo

model = RandomForestClassifier(random\_state=1)

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

# Sau đó, sử dụng mô hình phù hợp để đưa ra dự đoán và đánh giá các dự đoán bằng cách sử dụng số liệu hiệu suất độ chính xác của phân loại.

# make predictions

yhat = model.predict(X\_test)

# evaluate predictions

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

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

Kết quả:



Ví dụ 5: Train-Test Split for Regression tính toán sai số tuyệt đối trung bình

Code:

# train-test split evaluation random forest on the housing dataset

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)

Kết quả:



1. **Evaluate ML models [1.3]:**
   1. **The train/test/validation split.**

Không đào tạo mô hình trên toàn bộ tập dữ liệu vì nó rất quan trọng. ta phân chia đào tạo / kiểm tra điển hình sẽ là sử dụng 70% dữ liệu để đào tạo và 30% dữ liệu để kiểm tra.

Khi đánh giá mô hình của chúng ta để ngăn khả năng trang bị quá mức cho tập huấn luyện. Ta phân tách đào tạo / kiểm tra / xác thực điển hình sẽ là sử dụng 60% dữ liệu để đào tạo, 20% dữ liệu để xác nhận và 20% dữ liệu để kiểm tra.

Lưu ý rằng điều rất quan trọng là phải xáo trộn dữ liệu trước khi thực hiện các phần tách này để mỗi phần tách có một đại diện chính xác của tập dữ liệu.

* 1. **Evaluating Classification Models.**

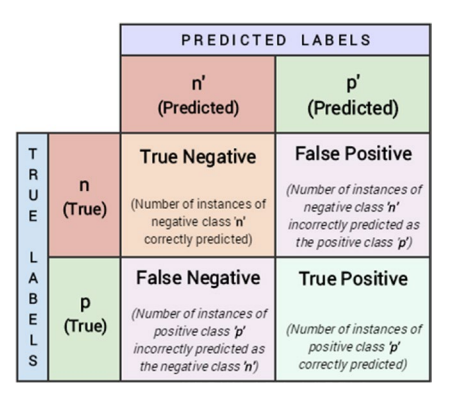
**True positives:** là khi bạn dự đoán một quan sát thuộc về một lớp và nó thực sự thuộc về lớp đó.

**True negatives:** là khi bạn dự đoán một quan sát không thuộc một lớp và nó thực sự không thuộc lớp đó.

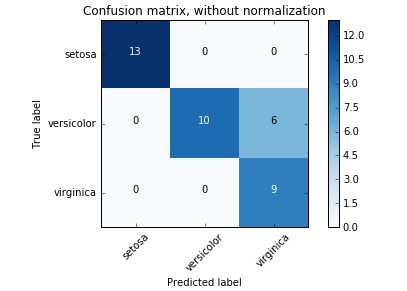
**False positives:** xảy ra khi bạn dự đoán một quan sát thuộc về một lớp trong khi thực tế thì không.

**False negatives:** xảy ra khi bạn dự đoán một quan sát không thuộc về một lớp trong khi thực tế thì nó lại như vậy.

Bốn kết quả này thường được vẽ trên một ma trận nhầm lẫn:



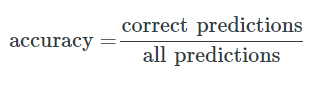
Có thể mở rộng ma trận nhầm lẫn này để vẽ các dự đoán phân loại nhiều lớp. Sau đây là một ví dụ về ma trận nhầm lẫn để phân loại các quan sát từ tập dữ liệu hoa Iris.



**Chỉ số Hiệu suất**

Mô tả cách các số liệu chính có thể được tính toán từ ma trận nhầm lẫn, tính toán chúng theo cách thủ công bằng các công thức cần thiết, sau đó so sánh kết quả với các hàm do scikit-learning cung cấp trên kết quả dự đoán của chúng tôi và đưa ra trực quan về các tình huống mà mỗi số liệu đó có thể được dùng.

**Accuracy**: Được định nghĩa là phần trăm dự đoán đúng cho dữ liệu thử nghiệm. Nó có thể được tính toán dễ dàng bằng cách chia số lần dự đoán đúng cho tổng số lần dự đoán:



Hay:



Code:

fw\_acc = round(meu.metrics.accuracy\_score(y\_true=y\_test, y\_pred=y\_pred), 5)

mc\_acc = round((TP + TN) / (TP + TN + FP + FN), 5)

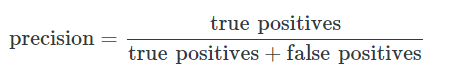
print('Framework Accuracy:', fw\_acc)

print('Manually Computed Accuracy:', mc\_acc)

# Framework Accuracy: 0.96491

# Manually Computed Accuracy: 0.96491

**Precision:** được định nghĩa là phần nhỏ của các ví dụ có liên quan (dương tính thực sự) trong số tất cả các ví dụ được dự đoán là thuộc một lớp nhất định.



Hay:



Code:

fw\_prec = round(meu.metrics.precision\_score(y\_true=y\_test, y\_pred=y\_pred), 5)

mc\_prec = round((TP) / (TP + FP), 5)

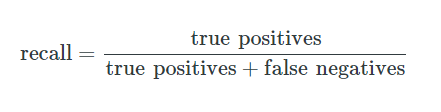
print('Framework Precision:', fw\_prec)

print('Manually Computed Precision:', mc\_prec)

Framework Precision: 0.96364

Manually Computed Precision: 0.96364

**Recall**: được định nghĩa là phần nhỏ của các ví dụ được dự đoán thuộc về một lớp so với tất cả các ví dụ thực sự thuộc về lớp đó.



Hay:



Code:

fw\_rec = round(meu.metrics.recall\_score(y\_true=y\_test, y\_pred=y\_pred), 5)

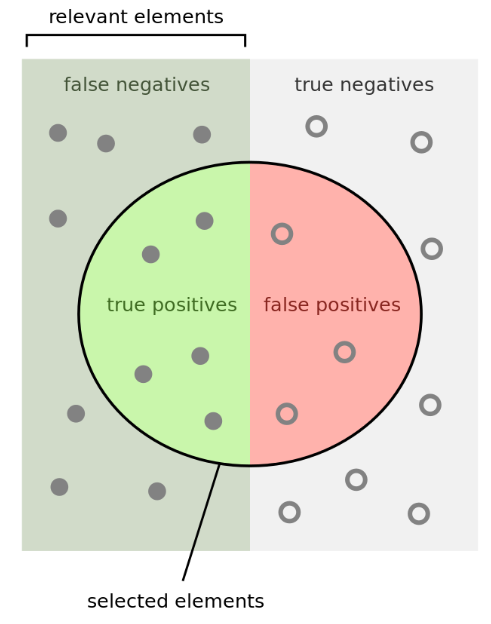
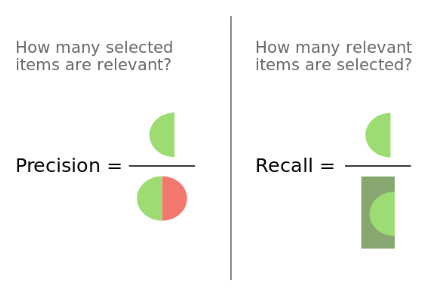
mc\_rec = round((TP) / (TP + FN), 5)

print('Framework Recall:', fw\_rec)

print('Manually Computed Recall:', mc\_rec)

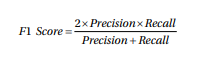
Framework Recall: 0.98148

Manually Computed Recall: 0.98148

**F1 Score**: Có một số trường hợp chúng tôi muốn tối ưu hóa cân bằng cả độ chính xác và khả năng thu hồi.

**F1 Score**: Là một chỉ số là trung bình hài hòa của độ chính xác và thu hồi, đồng thời giúp chúng tôi tối ưu hóa bộ phân loại để có độ chính xác cân bằng và hiệu suất thu hồi.



Hay: F1 = 

Code:

fw\_f1 = round(meu.metrics.f1\_score(y\_true=y\_test, y\_pred=y\_pred), 5)

mc\_f1 = round((2\*mc\_prec\*mc\_rec) / (mc\_prec+mc\_rec), 5)

print('Framework F1-Score:', fw\_f1)

print('Manually Computed F1-Score:', mc\_f1)

Framework F1-Score: 0.97248

Manually Computed F1-Score: 0.97248

* 1. **Evaluating Clustering Models.**
     1. **External Validation**

Có 3 số liệu phổ biến:

**Homogeneity**: Kết quả dự đoán mô hình phân cụm thỏa mãn tính đồng nhất nếu tất cả các cụm của nó chỉ chứa các điểm dữ liệu là thành viên của một lớp (dựa trên nhãn lớp thực sự).

**Completeness**: Kết quả dự đoán mô hình phân cụm đáp ứng tính đầy đủ nếu tất cả các điểm dữ liệu của nhãn lớp chân lý cơ bản cụ thể cũng là các phần tử của cùng một cụm.

**V-measure**: Giá trị trung bình hài hòa của điểm đồng nhất và tính đầy đủ cho chúng ta giá trị đo V.

Code:

km2\_hcv = np.round(metrics.homogeneity\_completeness\_v\_measure(y, km2\_labels), 3)

km5\_hcv = np.round(metrics.homogeneity\_completeness\_v\_measure(y, km5\_labels), 3)

print('Homogeneity, Completeness, V-measure metrics for num clusters=2: ', km2\_hcv)

print('Homogeneity, Completeness, V-measure metrics for num clusters=5: ', km5\_hcv)

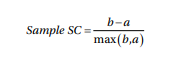
# Homogeneity, Completeness, V-measure metrics for num clusters=2: [ 0.422 0.517 0.465]

# Homogeneity, Completeness, V-measure metrics for num clusters=5: [ 0.602 0.298 0.398]

Chúng ta có thể thấy V-measure tốt hơn 2 mô hình còn lại.

* + 1. **Internal Validation**

**Silhouette Coefficient:**



a = khoảng cách trung bình giữa một mẫu và tất cả các điểm khác trong cùng một lớp

b = khoảng cách trung bình giữa một mẫu và tất cả các điểm khác trong cụm gần nhất tiếp theo

Code:

from sklearn import metrics

km2\_silc = metrics.silhouette\_score(X, km2\_labels, metric='euclidean')

km5\_silc = metrics.silhouette\_score(X, km5\_labels, metric='euclidean')

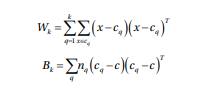
print('Silhouette Coefficient for num clusters=2: ', km2\_silc)

print('Silhouette Coefficient for num clusters=5: ', km5\_silc)

# Silhouette Coefficient for num clusters=2: 0.697264615606

# Silhouette Coefficient for num clusters=5: 0.510229299791

**Calinski-Harabaz Index:**

****

Code:

km2\_chi = metrics.calinski\_harabaz\_score(X, km2\_labels)

km5\_chi = metrics.calinski\_harabaz\_score(X, km5\_labels)

print('Calinski-Harabaz Index for num clusters=2: ', km2\_chi)

print('Calinski-Harabaz Index for num clusters=5: ', km5\_chi)

Calinski-Harabaz Index for num clusters=2: 1300.20822689

Calinski-Harabaz Index for num clusters=5: 1621.01105301

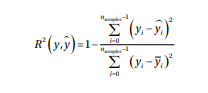
Chúng ta có thể thấy rằng cả hai điểm đều khá cao với kết quả cho năm cụm thậm chí còn cao hơn. Điều này cho thấy rằng chỉ dựa vào số liệu là không đủ và phải thử nhiều phương pháp đánh giá cùng với phản hồi từ các nhà khoa học dữ liệu cũng như các chuyên gia tên miền.

* 1. **Evaluating Regression Models.**

Mô hình hồi quy là một ví dụ về phương pháp học có giám sát và do có sẵn các biện pháp chính xác (các biến phản hồi số có giá trị thực), việc đánh giá chúng tương đối dễ dàng hơn so với các mô hình không có giám sát. Thông thường trong trường hợp các mô hình được giám sát, chúng ta tha hồ lựa chọn các số liệu và quyết định quan trọng là chọn đúng cho trường hợp sử dụng của chúng ta. Các mô hình hồi quy, giống như các mô hình phân loại, có một bộ số liệu khác nhau có thể được sử dụng để đánh giá chúng.

* + 1. **Coefficient of Determination or R2**

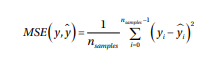
Đại diện cho tỷ lệ phương sai trong kết quả mà mô hình của chúng ta có thể dự đoán dựa trên các tính năng của nó.

****

Trong scikit-learning, sử dụng hàm r2\_score bằng cách cung cấp cho nó các giá trị thực và giá trị dự đoán (của biến đầu ra \ phản hồi).

* + 1. **Mean Squared Error.**

Được định nghĩa đơn giản là giá trị trung bình của sự khác biệt bình phương giữa sản lượng dự đoán và sản lượng thực. Lỗi bình phương thường được sử dụng vì nó không thể xác định được liệu dự đoán quá cao hay quá thấp, nó chỉ báo cáo rằng dự đoán đó không chính xác.



Trong scikit-learning, MSE có thể được tính toán bằng cách gọi hàm mean\_squared\_error từ mô-đun metrics.

1. **Data processing.**

Xử lý dữ liệu xảy ra khi dữ liệu được thu thập và chuyển thành thông tin có thể sử dụng được. Thường được thực hiện bởi một nhà khoa học dữ liệu hoặc nhóm các nhà khoa học dữ liệu, điều quan trọng là việc xử lý dữ liệu phải được thực hiện chính xác để không ảnh hưởng tiêu cực đến sản phẩm cuối cùng hoặc đầu ra dữ liệu.

Quá trình xử lý dữ liệu bắt đầu với dữ liệu ở dạng thô và chuyển đổi nó thành một định dạng dễ đọc hơn (đồ thị, tài liệu, v.v.), tạo cho nó hình thức và ngữ cảnh cần thiết để máy tính diễn giải và nhân viên sử dụng trong toàn tổ chức.

* 1. **Data collection.**

Thu thập dữ liệu là bước đầu tiên trong quá trình xử lý dữ liệu. Dữ liệu được lấy từ các nguồn có sẵn, bao gồm các hồ dữ liệu và kho dữ liệu . Điều quan trọng là các nguồn dữ liệu có sẵn phải đáng tin cậy và được xây dựng tốt để dữ liệu được thu thập (và sau đó được sử dụng làm thông tin) có chất lượng cao nhất có thể.

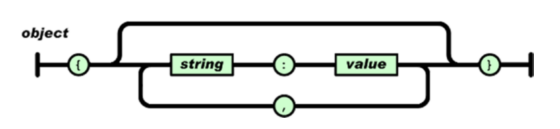
Để hiểu các cơ chế truy xuất dữ liệu khác nhau cho các kiểu dữ liệu khác nhau.

* **CSV:** Tệp giá trị được phân tách bằng dấu phẩy là tệp văn bản được phân tách sử dụng dấu phẩy để phân tách các giá trị. Mỗi dòng của tệp là một bản ghi dữ liệu. Mỗi bản ghi bao gồm một hoặc nhiều trường, được phân tách bằng dấu phẩy. Việc sử dụng dấu phẩy làm dấu phân tách trường là nguồn gốc của tên cho định dạng tệp này.

VD:



* **JSON:** JavaScript Object Notation là một kiểu dữ liệu mở trong JavaScript. Kiểu dữ liệu này bao gồm chủ yếu là text, có thể đọc được theo dạng cặp "thuộc tính - giá trị". Về cấu trúc, nó mô tả một vật thể bằng cách bọc những vật thể con trong vật thể lớn hơn trong dấu ngoặc nhọn.

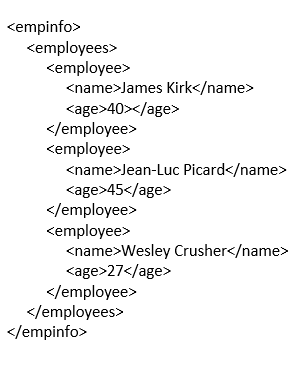
****

VD:



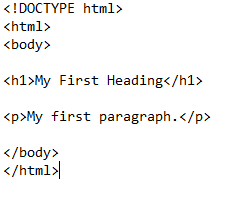
* **XML:** XML là ngôn ngữ đánh dấu với mục đích chung do W3C đề nghị, để tạo ra các ngôn ngữ đánh dấu khác. Đây là một tập con đơn giản của SGML, có khả năng mô tả nhiều loại dữ liệu khác nhau.

VD:



* **HTML:** HTML là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web trên World Wide Web. Nó có thể được trợ giúp bởi các công nghệ như CSS và các ngôn ngữ kịch bản giống như JavaScript.

VD:



* **Web Scraping:** Thu thập dữ liệu web, thu thập dữ liệu web hoặc trích xuất dữ liệu web là việc thu thập dữ liệu được sử dụng để trích xuất dữ liệu từ các trang web. Phần mềm duyệt web có thể truy cập trực tiếp vào World Wide Web bằng Giao thức truyền siêu văn bản hoặc trình duyệt web.
* **SQL:** Dịch là Ngôn ngữ truy vấn mang tính cấu trúc, là một loại ngôn ngữ máy tính phổ biến để tạo, sửa, và lấy dữ liệu từ một hệ quản trị cơ sở dữ liệu quan hệ. Ngôn ngữ này phát triển vượt xa so với mục đích ban đầu là để phục vụ các hệ quản trị cơ sở dữ liệu đối tượng-quan hệ. Nó là một tiêu chuẩn ANSI/ISO.
  1. **Data Description**
* Kiểu số, text và Categorical là kiểu dữ liệu này nằm giữa số và văn bản.
  1. **Data clean.**

Sau khi dữ liệu được thu thập, nó sẽ đi vào giai đoạn chuẩn bị dữ liệu. Chuẩn bị dữ liệu, thường được gọi là “tiền xử lý” là giai đoạn mà dữ liệu thô được làm sạch và sắp xếp cho giai đoạn xử lý dữ liệu sau. Trong quá trình chuẩn bị, dữ liệu thô được kiểm tra cẩn thận để tìm bất kỳ lỗi nào. Mục đích của bước này là loại bỏ dữ liệu xấu (dữ liệu thừa, không đầy đủ hoặc không chính xác) và bắt đầu tạo dữ liệu chất lượng cao để phục vụ cho hoạt động kinh doanh tốt nhất.

Các bước cơ bản data clean:

Bước 1: Xóa lặp lặp giá trị hoặc không liên quan.

Bước 2: Chỉnh sửa cấu trúc lỗi.

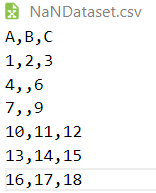
Bước 3: Lọc các ngoại lệ không mong muốn.

Bước 4: Xử lý thiếu dữ liệu.

Bước 5: Xác thực và đảm bảo chất lượng.

VD:

File CSV:



**Cleaning Rows with NaNs:**

Kiểm tra các cột có giá trị NaNs:

Code:

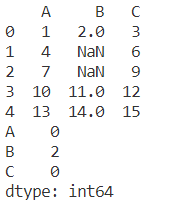
import pandas as pd

df = pd.read\_csv('NaNDataset.csv')

print(df.head())

print(df.isnull().sum())

Kết quả:



**Replacing NaN with the Mean of the Column:**

Thay thế các cột có giá trị NaN bằng trung bình của các cột mà chúng nằm trong đó.

Code:

import pandas as pd

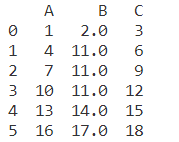
df = pd.read\_csv('NaNDataset.csv')

# replace all the NaNs in column B with the average of column B

df.B = df.B.fillna(df.B.mean())

print(df)

Kết quả:



**Removing Rows:**

Xóa hàng chứa giá trị NaN bằng hàm dropna():

Code:

import pandas as pd

df = pd.read\_csv('NaNDataset.csv')

print('Before droping NaN values')

print(df.head())

# print(df.isnull().sum())

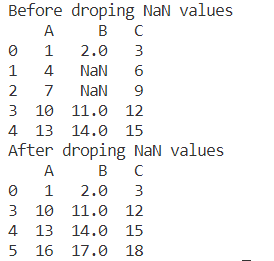
print('After droping NaN values')

df = pd.read\_csv('NaNDataset.csv')

df = df.dropna() # drop all rows with NaN

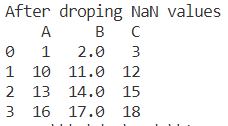
print(df)

Kết quả:



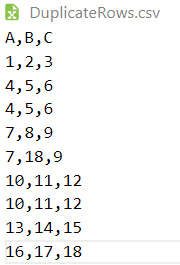
Nếu cần đặt lại chỉ mục có thể sử dụng hàm reset\_index():

df = df.reset\_index(drop=True) # reset the index



**Removing Duplicate Rows:**

File CSV:



Code:

import pandas as pd

df = pd.read\_csv('DuplicateRows.csv')

print(df.duplicated(keep=False))

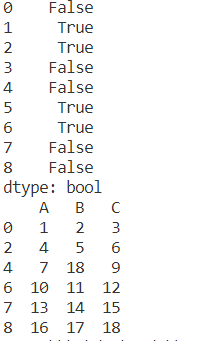
df.drop\_duplicates(subset=['A', 'C'], keep='last',inplace=True) # remove all duplicates in

 # columns A and C and keep

 # the last

print(df)

Kết quả:



**Outliers (Ngoại lai)**

Một ngoại lệ là một quan sát bất thường nằm xa phần lớn dữ liệu. Các yếu tố ngoại lai có thể ảnh hưởng đáng kể đến hiệu suất của mô hình Học máy.

**Loại bỏ ngoại lai**

Để loại bỏ giá trị ngoại lệ, cần xóa chính xác vị trí của dữ liệu ngoại lai

* 1. **Data input.**

Dữ liệu sạch sau đó được nhập vào đích của nó và được dịch sang ngôn ngữ mà nó có thể hiểu được. Đầu vào dữ liệu là giai đoạn đầu tiên trong đó dữ liệu thô bắt đầu ở dạng thông tin có thể sử dụng được.

Sau đó, kiểm tra mối tương quan giữa các tính năng và lập biểu đồ tương quan.

* 1. **Data split.**

Phân chia dữ liệu thành 2 tệp huấn luyện và tệp thử nghiệm theo tỷ lệ dựa trên 100% tổng số, tỷ lệ phân chia phổ biến: 80%-20%, 67%-33%, 50%-50%.

Tập huấn luyện được sử dụng để xây dựng mô hình.

Tập thử nghiệm chỉ được sử dụng cho các dự đoán.

* 1. **Data distribution.**

Phân phối dữ liệu là một hàm xác định tất cả các giá trị có thể có cho một biến và cũng định lượng tần suất tương đối (xác suất tần suất chúng xảy ra). Phân bố được coi là bất kỳ quần thể nào có dữ liệu phân tán.

Ưu điểm cơ bản của phân phối dữ liệu là ước tính xác suất của bất kỳ quan sát cụ thể nào trong không gian mẫu.

Các loại phân phối: Phân phối rời rạc, Phân phối nhị thức, Phân phối Poisson, Phân phối siêu đo, Phân bố hình học, Phân phối liên tục, Phân phối bình thường, Phân phối lognormal, Phân phối F, Phân phối Chi Square, Phân phối hàm mũ, Phân phối T sinh viên, Phân phối Weibull, Phân phối không bình thường, Phân phối kỳ lạ và Bi-modal.