Scikit Learn

Xây dựng mô hình đơn giản

HK1, Năm học 2022 - 2023

Các loại bài toán máy học cơ bản

Dựa vào bản chất dữ liệu học và tương tác giữa bộ học và môi trường, thì learning được phân loại

- Học có giám sát (Supervised Learning): bộ học được cung cấp đầu vào và đầu ra mong muốn, mục đích học quy luật ánh xạ từ đầu vào và đầu ra. Quy trình học được tiếp tục đến khi mô hình đạt mức độ chính xác mong muốn. Cặp dữ liệu này còn được gọi là (data, label) hay (data, target) 02 loại chính
    - Phân loại (Classification): có label được chia thành một số hữu hạn nhóm
    - Hồi quy (Regression): label có giá trị thực

Các loại bài toán máy học cơ bản

Dựa vào bản chất dữ liệu học và tương tác giữa bộ học và môi trường, thì learning được phân loại

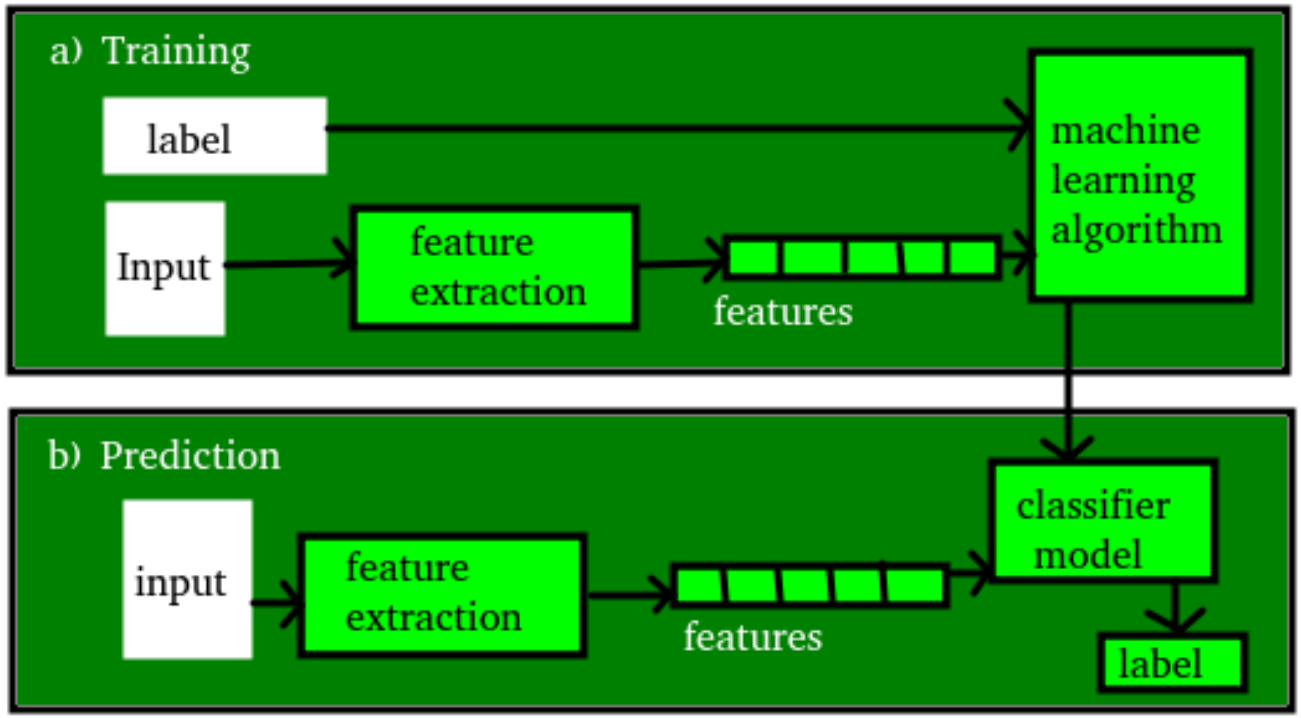
- Học không giám sát (Unsupervised Learning): không biết đầu ra/nhãn mà chỉ có dữ liệu vào. Thuật toán sẽ dựa vào cấu trúc dữ liệu để thực hiện một công việc nào đó như phân cụm, giảm chiều dữ liệu...

- Học bán giám sát (Semi-supervised Learning): một phần dữ liệu được dán nhãn và một phần dữ liệu không có nhãn

Thuật ngữ cơ bản

- Mô hình (model) đại diện cụ thể được học từ dữ liệu bằng cách áp dụng giải thuật học máy và được gọi là giả thuyết (hypothesis)
- Thuộc tính (feature): thuộc tính có thể đo lường được của dữ liệu (data). Tập hợp các features được một tả qua một vector thuộc tính (feature vector). Các feature vector sử dụng như input của model.
- Nhãn (Target/Label): giá trị được dự đoán bởi mô hình
- Huấn luyện (Training): Ý tưởng là đưa một bộ input (features) và đầu ra dự kiến (expected labels). Sau khi huấn luyện, một mô hình sẽ được thiết lập và sử dụng nó nó để dự đoán đầu ra cho dữ liệu mới cùng loại với dữ liệu huấn luyện
- Dự đoán (Prediction): khi mô hình đã sẵn sàng, một tập dữ liệu đầu vào mới sẽ được đưa vào để dự đoán output.

Thuật ngữ cơ bản



Load dữ liệu

Dataset là một tập dữ liệu gồm 2 phần chính

- Features (inputs, attributes): biến dữ liệu, biểu diễn bằng một ma trận (feature matrix). Danh sách tên thuộc tính được gọi là **feature names**
- Response (target, label, output): biến đầu ra phụ thuộc vào biến thuộc tính, thường là một cột response biểu diễn bằng một vector. Các giá trị có thể cho response vector được gọi là **target names**

Tải dữ liệu dựng sẵn trong sklearn

```
# load dataset
from sklearn.datasets import load_wine, load_boston, load_digits
wines = load_wine();
# trich feature matrix (X) va response vector (y)
X = wines.data
y = wines.target

# trich ten thuoc tinh va ten dich
feature_names = wines.feature_names
target_names = wines.target_names
print("Feature name: ", feature_names)
print("Target name: ", target_names)
print("Kich thuoc cua X", X.shape)
print("Kich thuoc cua y", y.shape)
print("In 5 dong dau tien", X[:5,:])

# Khong phai dataset nao cung co feature name hay target name
boston = load_boston()
print(boston.target_names) # se bao loi
digits = load_digits()
print(digits.feature_names) # bao loi
```

Tải dữ liệu từ file csv

- Sử dụng thư viện `pandas`
- 02 kiểu dữ liệu quan trọng
  - Series: mảng dán nhãn một chiều có khả năng chứa bất kỳ loại dữ liệu nào.
  - DataFrame: cấu trúc dữ liệu dánh nhãn 2 chiều với các cột có thể các loại dữ liệu khác nhau.

Tải dữ liệu từ file csv

```
import pandas as pd
#Load tu file csv
data = pd.read_csv('weather.csv')
# In kich thuoc
print("Kich thuoc ", data.shape)
# in ten thuoc tinh
print("Features: ", data.columns)
# luu du lieu vao ma tran feature va response vector
X = data[data.columns[:-1]]
y = data[data.columns[-1]]
# In 05 dong du lieu cua feature matrix
print(X.head())
# In 05 gia tri dau cua response vector
print(y.head())
```

Chia tập dữ liệu

Ở đây chúng ta sẽ bỏ qua bước tiền xử lý dữ liệu và dữ liệu lựa chọn là phù hợp.

Một khía cạnh quan trọng của các mô hình máy là xác định độ chính xác. Để làm được điều này thì mô hình cần được huấn luyện (train) sử dụng dữ liệu và sau đó dự đoán mô hình dự đoan giá trị target với cùng tập dữ liệu. Tuy nhiên phương pháp sẽ có sai sót.

Một cách khắc phục: chia tập dữ liệu thành 02 phần (01 phần dành cho huấn luyện mô hình và phần khác dành cho kiểm tra mô hình)

Chia tập dữ liệu

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target

# package dung chi du lieu
from sklearn.model_selection import train_test_split
# Chia tap du lieu thanh train va test
# voi train chiem 60% du lieu, random_state la seed
    cua bo sinh so random
X_train, X_test, y_train, y_test = train_test_split(X
    , y, test_size=0.4, random_state = 1)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Huấn luyện mô hình

Scikit-learn cung cấp nhiều giải thuật có giao diện nhất quán để học, dự đoán độ chính xác.

Nếu sử dụng giải thuật K láng giềng (KNN - K nearest neighbors) để phân loại

```
# import thu vien
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
# Chuan bi du lieu
# load du lieu vao bien iris
iris = load_iris()
# luu tru feature matrix vao bien X
X = iris.data
# luu tru target vector vao bien y
y = iris.target
# Chia tap du lieu vao tap training va test
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, random_state=123)
```

Huấn luyện mô hình

```
# Thiet lap mo hinh
knn = KNeighborsClassifier(n_neighbors=1)
# Qua trinh hoc
knn.fit(X_train, y_train)
# Qua trinh du doan tren tap test
y_pred = knn.predict(X_test)
# Danh gia
acc = metrics.accuracy_score(y_test, y_pred)
print('kNN model accuracy: ', acc)
# Du doan cho du lieu moi
sample = [[3.2, 5.3, 4.1, 2.8], [2.1,3.2,1.9,0.5]]
preds = knn.predict(sample)
# Chuyen tu gia tri cua target sang ten that su cua
    label
pred_species = [iris.target_names[p] for p in preds]
print('Predictions: ', pred_species)
```

Yêu cầu thực hiện

- Thay đổi cách chia dữ liệu: train - test được chia theo tỷ lệ 7:3 với random_state = 123
- Vẫn sử dụng KNN nhưng xây dựng các mô hình với giá trị neighbors lần lượt 1, 3, 5
- Dựa vào độ chính xác: giá trị neighbors nào thì độ chính xác cao hơn

Thay đổi model. Sử dụng model cây quyết định (Decision Tree) khác:

- Để sử dụng mô hình cần import: `from sklearn.tree import DecisionTreeClassifier`
- Thay mô hình KNN bằng mô hình: `DecisionTreeClassifier(max_depth=5)`
- Thay đổi giá trị `max_depth` thành 1, 3, 7 và xem thử giá trị nào nên lựa chọn