第 3 章

常用数据集

在机器学习领域，数据集如同烹饪中的食材一样重要。选择合适的数据集不仅能帮助我们更好地理解算法的工作原理，还能为模型的训练提供坚实的基础。

本章将介绍 3 种类型的数据集：小型数据集、大型数据集和生成数据集，并详细探讨它们的特点及使用方法。

要使用这些数据集，首先需要导入 sklearn 的 datasets 模块。

```
>>> from sklearn import datasets
```

sklearn 提供的数据集通常具有类似字典的结构，主要包含以下键值对。

- 'DESCR'：数据集的描述信息。
- 'data'：二维数组，其中每一行代表一个样本，每一列对应一个特征。
- 'target'：标签数组（对于监督学习任务）。

3.1 小型数据集

常用的小型数据集有 6 个，具体如表 3-1 所示。这些数据集可通过如下命令导入。

```
>>> from sklearn.datasets import load_*
```

其中的“*”需要替换为具体的数据集名称。例如，load_iris 表示导入鸢尾花（Iris）数据集。

表 3-1　sklearn 提供的小型数据集

数据集名称	任务类型	数据规模	导入命令
糖尿病	回归	442×10	load_diabetes
手写数字	分类	1797×64	load_digits
乳腺癌	分类和聚类	569×30	load_breast_cancer
鸢尾花	分类和聚类	150×4	load_iris
葡萄酒	分类	178×13	load_wine
体能训练	回归	20×3	load_linnerud

3.1.1　糖尿病数据集

糖尿病数据集包含 442 名患者的生理数据及其一年后的病情发展情况。每个样本由 10 个属性值(特征值)组成,如表 3-2 所示。

表 3-2　糖尿病数据集的 10 个特征

特　征	说　明	特　征	说　明
age	年龄	s2	低密度脂蛋白
sex	性别	s3	高密度脂蛋白
bmi	体重指数	s4	总胆固醇
bp	血压平均值	s5	血清甘油三酯
s1	总血清胆固醇	s6	血糖水平

示例代码:

```
>>> from sklearn.datasets import load_diabetes
>>> import pandas as pd
>>> diabetes = load_diabetes()
>>> data = diabetes.data
>>> target = diabetes.target                    #目标值
>>> feature_names = diabetes.feature_names #特征名称
>>> feature_names
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
>>> df = pd.DataFrame(data, columns=feature_names)
>>> print(df.head(3))                           #输出前三行①,如图 3-1 所示
```

```
        age       sex       bmi        bp        s1        s2        s3
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356

         s4        s5        s6
0 -0.002592  0.019907 -0.017646
1 -0.039493 -0.068332 -0.092204
2 -0.002592  0.002861 -0.025930
```

图 3-1　糖尿病数据集前三个样本展示

使用 info()方法查看数据集的基本信息,如图 3-2 所示。

```
>>> df.info()
```

该数据集中的每一列都经过标准化处理(包括性别),其标准化公式如下:

$$x' = \frac{x - \mu}{\sigma \sqrt{n}}$$

① 使用命令 pd.set_option('display.max_columns', None)可以取消列方向的省略,显示所有列。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     442 non-null    float64
 1   sex     442 non-null    float64
 2   bmi     442 non-null    float64
 3   bp      442 non-null    float64
 4   s1      442 non-null    float64
 5   s2      442 non-null    float64
 6   s3      442 non-null    float64
 7   s4      442 non-null    float64
 8   s5      442 non-null    float64
 9   s6      442 non-null    float64
dtypes: float64(10)
memory usage: 34.7 KB
```

图 3-2　糖尿病数据集基本信息

其中，x 是某个特征值；μ 是该特征的平均值；σ 是该特征的总体标准差（非样本标准差）；n 是样本数量。因此，该数据集中每一列的平方和都等于 1。

3.1.2　手写数字数据集

手写数字数据集包含 1797 个从 0 到 9 的手写数字图像，每个数字由 8×8[①] 的矩阵表示，矩阵中元素的取值范围为 0～16，代表颜色深度。手写数字 0 对应的 8×8 矩阵如图 3-3 所示。手写数字 0 到 9 的示例图像如图 3-4 所示。该数据集可通过以下命令加载。

```
>>> from sklearn.datasets import load_digits
```

```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0. 13. 15. 10. 15.  5.  0.]
 [ 0.  3. 15.  2.  0. 11.  8.  0.]
 [ 0.  4. 12.  0.  0.  8.  8.  0.]
 [ 0.  5.  8.  0.  0.  9.  8.  0.]
 [ 0.  4. 11.  0.  1. 12.  7.  0.]
 [ 0.  2. 14.  5. 10. 12.  0.  0.]
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```

图 3-3　手写数字 0 对应的 8×8 矩阵

图 3-4　手写数字 0 到 9 的示例图像

load_digits()方法的基本语法如下。

```
load_digits(n_class=10, return_X_y=False)
```

① 也就是说，每个样本有 64 个特征。

- n_class：指定返回样本的类别数量，默认值为 10。若设置为 5，则仅返回类别标签为 0 到 4 的样本。
- return_X_y：控制数据的返回形式。若设为 True，则以元组(data，target)的形式返回特征数据和目标标签；默认值为 False，此时返回一个字典，包含'data'和'target'等键。

示例代码：

```
>>> from sklearn.datasets import load_digits
>>> digits = load_digits()
>>> digits.keys()
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images',
'DESCR'])
#数据集包含 1797 个样本，每个样本有 64 个特征值
>>> digits.data.shape
(1797, 64)
#包含 1797 个样本的目标值，以一维数组的形式表示
>>> digits.target.shape
(1797,)
#该数据集包含 10 个类别，对应标签为 0 到 9
>>> list(digits.target_names)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

3.1.3　乳腺癌数据集

美国威斯康星州乳腺癌数据集共包含 569 个样本，每个样本由 30 个属性值组成。这些特征主要包括肿瘤的各种统计量，如 mean radius、mean texture 等。

示例代码：

```
>>> from sklearn.datasets import load_breast_cancer
>>> cancer = load_breast_cancer()
>>> cancer.data.shape
(569, 30)
>>> cancer.target.shape
(569,)
>>> import pandas as pd
>>> df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
>>> df.head(3)                                    #输出前三行，如图 3-5 所示
>>> print(cancer.feature_names)
["mean radius", "mean texture", …, "worst fractal dimension"]
```

	mean radius	mean texture	...	worst symmetry	worst fractal dimension
0	17.99	10.38	...	0.4601	0.11890
1	20.57	17.77	...	0.2750	0.08902
2	19.69	21.25	...	0.3613	0.08758

图 3-5　乳腺癌数据集前三个样本

目标值包含两个类别："malignant"（恶性，标签为 0）和"benign"（良性，标签为 1）。其中，恶性样本共 212 个，良性样本共 357 个。

```
>>> list(cancer.target_names)
['malignant', 'benign']
```

3.1.4 鸢尾花数据集

鸢尾花数据集由 Fisher 于 1936 年收集整理，包含 3 个品种：山鸢尾（setosa）、变色鸢尾（versicolor）和弗吉尼亚鸢尾（virginica），每个品种各有 50 个样本。

每个样本包含 4 个特征：萼片长度（sepal length）、萼片宽度（sepal width）、花瓣长度（petal length）和花瓣宽度（petal width），如图 3-6 所示。

图 3-6　鸢尾花的 3 个品种

示例代码：

```
>>> from sklearn.datasets import load_iris
>>> import pandas as pd
>>> iris = load_iris()
>>> iris.feature_names                          #特征名称
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
>>> iris.data.shape
(150, 4)
>>> iris.target.shape
(150,)
>>> df = pd.DataFrame(iris.data, columns=iris.feature_names)
>>> df.head(3)                                  #输出前三行，如图 3-7 所示
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2

图 3-7　鸢尾花数据集前三个样本

3 个品种山鸢尾（setosa）、变色鸢尾（versicolor）和弗吉尼亚鸢尾（virginica）的目标值分别对应标签 0、1 和 2。仅使用萼片长度与宽度进行可视化，如图 3-8 所示。

```
>>> dir(iris)                                   #查看数据集属性列表
['DESCR', 'data', 'data_module', 'feature_names', 'filename', 'frame', 'target',
```

```
'target_names']
>>> list(iris.target_names)              #目标值,对应标签 0、1 和 2
['setosa', 'versicolor', 'virginica']
```

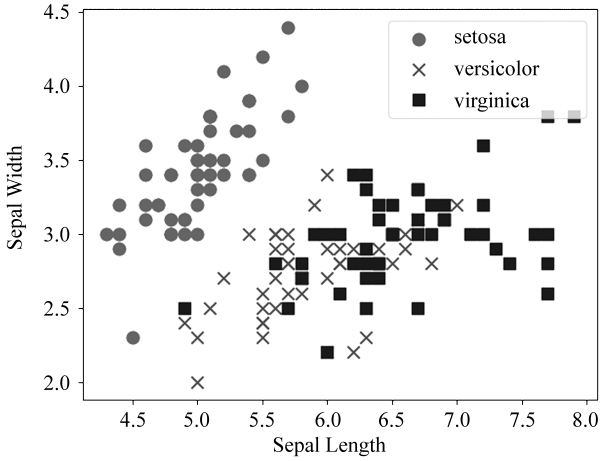

图 3-8　鸢尾花数据集

3.1.5　葡萄酒数据集

葡萄酒识别数据集包含 178 个样本,每个样本由 13 个属性值(特征值)组成,可用于分类任务。可以使用如下命令导入该数据集。

```
>>> from sklearn.datasets import load_wine
```

下面是一段示例代码,展示了如何加载并查看该数据集的基本信息。

```
>>> wine = load_wine()
>>> wine.data.shape                      #178 个样本,每个样本有 13 个特征值
(178, 13)
>>> wine.target.shape                    #目标标签数组
(178,)
>>> print(wine.feature_names)            #葡萄酒数据集的 13 个特征名称
["alcohol", "malic_acid", … , "proline"]
```

将数据转换为 pandas.DataFrame 格式,以便查看前几行数据。

```
>>> df = pd.DataFrame(wine.data, columns=wine.feature_names)
```

设置显示的最大列数为 4,避免因列太多而影响可读性。

```
>>> pd.set_option("display.max_columns", 4)
>>> df.head(3)                           #显示前三行数据,如图 3-9 所示
```

该数据集共有 3 个类别,分别为'class_0'、'class_1'和'class_2',对应的标签依次为 0、1 和 2。可以通过以下代码查看目标类别的名称。

```
>>> list(wine.target_names)
['class_0', 'class_1', 'class_2']
```

```
   alcohol  malic_acid  ...  od280/od315_of_diluted_wines  proline
0    14.23        1.71  ...                          3.92   1065.0
1    13.20        1.78  ...                          3.40   1050.0
2    13.16        2.36  ...                          3.17   1185.0
```

图 3-9　葡萄酒数据集前三个样本

3.1.6　体能训练数据集

体能训练数据集包含 20 个样本，每个样本由 3 个属性值(特征值)组成。目标变量也包含 3 个属性，分别是 Weight(体重)、Waist(腰围)和 Pulse(脉搏)，对应的类别标签为 0、1 和 2。该数据集常用于多输出回归问题或多目标建模任务。可以使用如下命令导入该数据集。

```
>>> from sklearn.datasets import load_linnerud
```

下面是一段示例代码，展示了如何加载并查看该数据集的基本信息。

```
>>> linnerud = load_linnerud()
>>> linnerud.data.shape                    #20 个样本,每个样本有 3 个特征值
(20, 3)
>>> linnerud.target.shape                  #目标数组,共 20 行 3 列
(20, 3)
>>> linnerud.feature_names                 #数据集中 3 个特征的名称
['Chins', 'Situps', 'Jumps']               #引体向上、仰卧起坐、跳跃次数
>>> import pandas as pd
>>> df = pd.DataFrame(linnerud.data, columns=linnerud.feature_names)
>>> df.head(3)                             #显示前三个样本
   Chins  Situps  Jumps
0    5.0   162.0   60.0
1    2.0   110.0   60.0
2   12.0   101.0  101.0
>>> linnerud.target_names                  #目标值的名称: 体重、腰围和脉搏
['Weight', 'Waist', 'Pulse']               #对应的标签分别为 0、1 和 2
```

获取数值属性的统计摘要信息，如图 3-10 所示。

```
>>> df.describe()
```

```
           Chins      Situps      Jumps
count  20.000000   20.000000   20.00000
mean    9.450000  145.550000   70.30000
std     5.286278   62.566575   51.27747
min     1.000000   50.000000   25.00000
25%     4.750000  101.000000   39.50000
50%    11.500000  122.500000   54.00000
75%    13.250000  210.000000   85.25000
max    17.000000  251.000000  250.00000
```

图 3-10　体能训练数据集的统计摘要信息

describe()方法返回的数据包括样本数量、平均值、标准差、最小值、3 个四分位数(25%、50%和 75%)以及最大值。

3.2　大型数据集

在 scikit-learn 中,提供了多个适用于大规模数据分析的大型数据集。其中,常用的大型数据集有 6 个,如表 3-3 所示。这些大型数据集通常不会默认包含在库中,首次使用时会自动从网络下载。导入这些数据集的基本命令如下。

```
>>> from sklearn.datasets import fetch_*
```

其中“*”需要替换为具体的数据集名称。例如,使用 fetch_california_housing 可以导入加州住房价格数据集。下面对这 6 个常用的大型数据集进行简要介绍,并总结于表 3-3 中。

表 3-3　sklearn 提供的六大常用大型数据集

数据集名称	任务类型	数据规模	导入命令
Olivetti 人脸数据集	分类、降维	400×4096	fetch_olivetti_faces
20 新闻组数据集	分类	18846×130107	fetch_20newsgroups
带标签的人脸数据集	分类、降维	13233×5828	fetch_lfw_people
路透社英文新闻文本	分类、降维	804414×47236	fetch_rcv1
加州住房价格数据集	回归	20640×9	fetch_california_housing
MNIST 手写数字数据集	分类	70000×784	fetch_openml(name='mnist_784')

3.2.1　Olivetti 人脸数据集

Olivetti 人脸数据集是由英国剑桥 AT&T 实验室在 1992 年 4 月至 1994 年 4 月期间拍摄的一组人脸图像数据。该数据集共包含 400 张人脸照片,即每人 10 张照片,共 40 人。每个样本由 4096 个特征值组成,表示一幅大小为 64×64 的灰度图像像素值。图 3-11 展示了 Olivetti 人脸数据集的基本特性。

```
=================   ======================
Classes                                  40
Samples total                           400
Dimensionality                         4096
Features            real, between 0 and 1
=================   ======================
```

图 3-11　Olivetti 人脸数据集的基本特性

可以使用以下命令导入该数据集。

```
>>> from sklearn.datasets import fetch_olivetti_faces
```

加载数据集的基本方法如下。

```
>>> faces = fetch_olivetti_faces()
```

fetch_olivetti_faces()方法的参数及说明如下。

```
fetch_olivetti_faces(shuffle=False, download_if_missing=True , return_X_y=
False)
```

- shuffle：是否对样本进行随机打乱，默认值为 False。
- download_if_missing：本地没有数据集时是否尝试联网下载，默认值为 True。
- return_X_y：若为 True，则返回(data，target)形式的元组；否则以字典形式返回所有信息(包括'data'和'target')，默认值为 False。

其他可选参数还包括 data_home 和 random_state[①] 等。

```
>>> faces = fetch_olivetti_faces()
>>> dir(faces)                          #查看数据集属性列表
['DESCR', 'data', 'images', 'target']
>>> faces.data.shape
(400, 4096)
>>> faces.target.shape
(400,)
>>> type(faces.images)
<class 'numpy.ndarray'>
>>> import matplotlib.pyplot as plt
>>> plt.imshow(faces.images[0])
>>> plt.show()                          #如图 3-12 所示
```

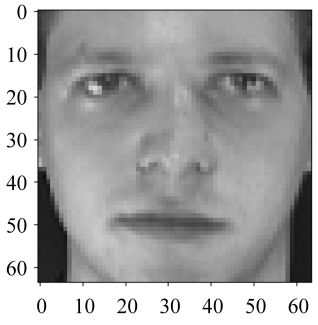

图 3-12　Olivetti 数据集的人脸照片

3.2.2　20 新闻组数据集

20 新闻组数据集(20 Newsgroups Dataset)共包含 18 846 篇新闻文章，涵盖 20 个不同的主题类别，是文本分类、信息检索和自然语言处理领域广泛使用的国际标准数据集之一。其基本结构如图 3-13 所示。

① 设置随机数生成器的种子，默认值为 0。保证程序每一次运行时，对数据集进行分割而得到的训练集和测试集都完全相同。

```
=================   ==========
Classes                     20
Samples total            18846
Dimensionality               1
Features                  text
=================   ==========
```

图 3-13　20 新闻组数据集的基本特性

该数据集提供了以下两个版本。

- 原始文本版本：保留原始文本内容，适用于文本预处理和特征提取。
- 向量化版本：对原始文本进行了向量化处理，便于直接用于机器学习模型训练。

这两个版本分别通过以下命令导入。

```
>>> from sklearn.datasets import fetch_20newsgroups                #原始文本版本
>>> from sklearn.datasets import fetch_20newsgroups_vectorized    #向量化版本
```

fetch_20newsgroups()函数的原型如下。

```
fetch_20newsgroups(subset='train', categories=None, remove=(), return_X_y =
False)
```

- subset：指定要加载的数据子集，默认值为'train'，其他可选值包括'test'和'all'(表示全部数据)。
- categories：指定要加载的新闻类别名称，默认值为 None，即加载所有 20 个类别。
- remove：指定要从文本中移除的内容类型，可能的取值如下。
  - 'headers'：去除邮件头信息。
  - 'footers'：去除邮件底部签名。
  - 'quotes'：去除引用的回复内容。
- return_X_y：若设为 True，则以(data，target)元组形式返回数据；否则以字典形式返回完整数据(包括'data'和'target')，默认值为 False。

其他常用参数还包括 data_home、download_if_missing、shuffle、random_state 等。

示例代码及输出：

```
>>> news20 = fetch_20newsgroups(subset='train')
>>> dir(news20)                    #查看数据集属性列表
['DESCR', 'data', 'filenames', 'target', 'target_names']
```

data 字段是一个字符串列表，每个元素对应一篇新闻文章。

```
>>> type(news20.data)              #data 字段
<class 'list'>
>>> len(news20.data)               #训练集中共有 11314① 篇文章
11314
>>> news20.target.shape            #目标标签数组
```

① 训练集和测试集的样本数分别为 11 314 和 7532，共计 18 846。

```
(11314,)
>>> labels = news20.target_names  #获取 20 个新闻类别的名称
['alt.atheism', 'comp.graphics', …, 'talk.politics.misc', 'talk.religion.
misc']
>>> len(labels)                   #共有 20 个新闻类别
20
>>> print(news20.data[0])         #查看第一篇文章内容,如图 3-14 所示
```

```
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----
```

图 3-14　20 新闻组数据集中第一篇文章的原始文本内容

fetch_20newsgroups_vectorized()函数返回数据集的第二个版本,即已经使用 TF-IDF[①] 进行向量化处理后的特征矩阵。函数原型如下。

```
fetch_20newsgroups_vectorized(subset='train', remove=(), return_X_y=False,
normalize=True)
```

- normalize:是否对每个文档的特征向量进行归一化,默认值为 True。

其他参数与 fetch_20newsgroups()相同,包括 subset、categories、remove、return_X_y、data_home、download_if_missing 等。

示例代码:

```
>>> news20_vec = fetch_20newsgroups_vectorized(subset="train")
```

此时,news20_vec.data 是一个稀疏矩阵,可以直接输入机器学习模型进行训练。

3.2.3　带标签的人脸数据集

带标签的人脸数据集(Labeled Faces in the Wild,LFW)是当前人脸识别领域广泛使用的一个标准数据集。该数据集中的人脸图像均采集自现实生活中非受控场景下的照片,具有多种变化因素,如不同的姿态、光照条件、表情、年龄、遮挡等。因此,即使是同一个人的人脸图像也可能存在较大差异,使得识别任务具有较高难度。

此外,部分图像中包含多张人脸。为简化问题,通常只选取每张图像中心位置的人脸

① 词频-逆文档频率(Term Frequency-Inverse Document Frequency,TF-IDF),参见本书第 4 章。

作为目标对象，其余区域的人脸被视为背景干扰。

整个 LFW 数据集共包含 13233 张人脸图像，每张图像都标注了对应的人物姓名，共计 5749 位不同人物(类别)。其中，绝大多数人物仅有一张图像，这进一步增加了识别和分类的挑战性。每张图像的尺寸均为 62×47 像素，全部为彩色图像。LFW 数据集的基本特性如图 3-15 所示。

Classes	5749
Samples total	13233
Dimensionality	5828
Features	real, between 0 and 255

图 3-15　LFW 数据集的基本特性

在 scikit-learn 中，可以使用如下命令加载 LFW 数据集。

```
>>> from sklearn.datasets import fetch_lfw_people
```

fetch_lfw_people()函数的常用参数包括 return_X_y、min_faces_per_person、slice_、resize、download_if_missing 等。其中，min_faces_per_person 用于筛选至少包含指定数量图像的人物，默认值为 0，表示不进行筛选(保留所有人物，但会导致类别极度不平衡)；slice_对图像进行裁剪操作，可用于提取感兴趣区域；resize 用于调整图像大小的比例，默认值为 0.5，即缩小为原始尺寸的一半。

示例代码：

```
>>> people = fetch_lfw_people()
>>> dir(people)                    #查看数据集属性列表
['DESCR', 'data', 'images', 'target', 'target_names']
>>> people.data.shape
(13233, 2914)
```

3.2.4　路透社英文新闻文本

路透社英文新闻文本(Reuters Corpus Volume 1，RCV1)是由路透社提供的大型英文新闻语料库，广泛应用于文本分类和信息检索领域。该数据集包含 804 414 篇新闻文章，共涉及 103 个主题类别(如图 3-16 所示)，每篇文章可能被标注为多个主题，因此适用于多标签分类任务。

Classes	103
Samples total	804414
Dimensionality	47236
Features	real, between 0 and 1

图 3-16　RCV1 语料库的基本特性

RCV1 数据集是文本挖掘领域的重要基准数据之一，其丰富的内容和多标签特性使其成为研究分类模型性能的理想选择。数据集中的每篇文章都经过人工标注，具有较高的语义质量和可用性。

在 scikit-learn 中，可以通过以下命令导入 RCV1 数据集。

```
>>> from sklearn.datasets import fetch_rcv1
```

fetch_rcv1()函数的常用参数包括 subset、download_if_missing、random_state、shuffle、return_X_y 等。在该数据集中，训练集和测试集的样本数分别为 23149 和 781265。

```
>>> rcv1 = fetch_rcv1()
>>> dir(rcv1)                          #查看数据集属性列表
['DESCR', 'data', 'sample_id', 'target', 'target_names']
#表示共有 804414 篇文章，每篇文章使用 47236 个特征进行表示(通常使用 TF-IDF 向量化
>>> rcv1.data.shape
(804414, 47236)
```

总之，RCV1 数据集因其规模大、类别多、标注准确，已成为评估文本分类算法性能的重要标准数据集之一。

3.2.5 加州住房价格数据集

加州住房价格数据集(California Housing Dataset)是基于 1990 年美国加利福尼亚人口普查数据构建的。该数据集广泛应用于回归任务，主要用于预测某一街区的房价中位数。

如图 3-17 所示，该数据集包含 20640 个样本，每个样本由 9 个属性值(特征值)组成，这些属性描述了不同街区的人口统计和地理信息，如表 3-4 所示。

```
==============   ==============
Samples total            20640
Dimensionality               8
Features                  real
Target           real 0.15 - 5.
==============   ==============
```

图 3-17 加州住房价格数据集的基本特性

表 3-4 加州住房价格数据集中的属性说明

属 性 名	数据类型	说 明
MedInc	浮点数	街区内家庭收入中位数
HouseAge		房屋年龄中位数
AveRooms		每户平均房间数
AveBedrms		每户平均卧室数
Population		街区总人口
AveOccup		每户平均居住人数
Latitude		街区的纬度
Longitude		街区的经度
MedHouseVal		房价中位数(目标变量)

一个“街区”是美国人口普查局发布数据时使用的最小地理单位，通常包含 600～3000 人。通过结合这些人口统计数据，可以对房价进行建模与预测。

导入该数据集的方法如下。

```
>>> from sklearn.datasets import fetch_california_housing
```

加载数据并以 DataFrame 形式查看前几行数据。

```
>>> housing = fetch_california_housing(as_frame=True)
```

默认情况下，可以使用.frame 属性查看整个数据框架(包括特征数据和目标变量)。

```
>>> housing.frame.head()                #输出省略
>>> housing.data.head()                 #单独查看特征数据,输出省略
>>> housing.target.head(3)              #查看目标变量(房价中位数)的前三行
0    4.526
1    3.585
2    3.521
Name: MedHouseVal, dtype: float64
```

使用 info()方法可以查看数据集的结构信息，包括样本总数、各字段名称及其数据类型等。

```
>>> housing.frame.info()                #部分输出结果
Data columns (total 9 columns):
#     Column       Non-Null   Count      Dtype
---   ------      --------   -----      -----
0     MedInc       20640      non-null   float64
1     HouseAge     20640      non-null   float64
2     AveRooms     20640      non-null   float64
3     AveBedrms    20640      non-null   float64
4     Population   20640      non-null   float64
5     AveOccup     20640      non-null   float64
6     Latitude     20640      non-null   float64
7     Longitude    20640      non-null   float64
8     MedHouseVal  20640      non-null   float64
```

使用 describe()方法可以快速获取数值型特征的统计信息，如均值、标准差、极值、四分位数等。

```
#定义需要分析的数值型特征列表
>>> features_of_interest = ["AveRooms", "AveBedrms", "AveOccup", "Population"]
#查看这些特征的统计信息(保留 2 位小数)
>>> housing.frame[features_of_interest].describe().round(2)
```

上述代码输出的结果如表 3-5 所示。

综上所述，该数据集因真实性强、结构清晰、数据适中，已成为回归任务的典型入门数据集，适用于模型训练、评估、特征工程和可视化等多个教学与研究场景。

表 3-5　4 个数值属性的统计摘要信息

统计量	AveRooms	AveBedrms	AveOccup	Population
count	20640.00	20640.00	20640.00	20640.00
mean	5.43	1.10	3.07	1425.48
std	2.47	0.47	10.39	1132.46
min	0.85	0.33	0.69	3.00
25%	4.44	1.01	2.43	787.00
50%	5.23	1.05	2.82	1166.00
75%	6.05	1.10	3.28	1725.00
max	141.91	34.07	1243.33	35682.00

3.2.6　MNIST 手写数字数据集

MNIST(Modified National Institute of Standards and Technology)手写数字数据集由 70 000 张手写数字图像组成,来源于美国高中生和人口普查局工作人员的手写样本,涵盖从 0 到 9 的 10 个数字。每张图像都附有相应的标签,用于标识其代表的数字。

每张图像的尺寸为 28×28 像素,因此每个样本包含 28×28=784 个特征。每个特征值表示对应像素点的灰度强度,取值范围为 0～255,其中 0 表示黑色前景(手写数字),255 表示白色背景。图 3-18 展示了其中 30 张手写数字图像的示例。

在 scikit-learn 中,可以通过以下命令加载该数据集。

```
>>> from sklearn.datasets import fetch_openml
```

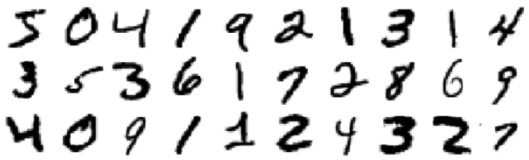

图 3-18　手写数字图像示例

示例代码及输出:

```
>>> mnist = fetch_openml("mnist_784", version=1 , parser='auto')
>>> type(mnist.data)
<class 'pandas.core.frame.DataFrame'>
#该数据集包含 70 000 张图像,每张图像有 784 个像素特征(28×28 像素)
>>> mnist.data.shape                    #数据形状
(70000, 784)
>>> mnist.target.shape                  #标签形状
(70000,)
#通过 dir()函数可以查看数据集对象的所有属性
```

```
>>> dir(mnist)                                    #查看数据集的属性列表
['DESCR', 'categories', 'data', 'details', 'feature_names', 'frame', 'target',
'target_names', 'url']
```

MNIST 数据集常用于手写数字识别任务，是一个典型的多分类问题。为进行模型训练和评估，通常需要将数据集划分为训练集和测试集。使用 train_test_split()函数进行划分的示例如下。

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.
target, test_size=0.2, random_state=42)
```

上述代码将原始数据集按 80％训练集和 20％测试集的比例划分，便于后续建模与评估。

3.3　生成数据集

在 scikit-learn 中，提供了多个用于生成模拟数据的函数。这些函数可以帮助用户快速构造用于模型测试、演示或算法验证的数据集，特别适用于机器学习的学习、调试与性能评估。scikit-learn 提供了 7 个常用的数据集生成函数，如表 3-6 所示。生成数据集的导入命令如下。

```
>>> from sklearn.datasets import make_*
```

其中“*”需要替换为具体的数据集名称，如 make_regression。下面对这 7 个常用数据集进行简要介绍，并汇总于表 3-6 中。

表 3-6　scikit-learn 中常用的模拟数据集

数据集名称	任务类型	数 据 维 度	导 入 命 令
回归模型	回归	n_samples×n_features	make_regression
聚类模型	聚类		make_blobs
分类模型	分类、降维		make_classification
多维正态分布	分类		make_gaussian_quantiles
同心圆数据集	分类、聚类	n_samples×2	make_circles
双半圆数据集	分类、聚类		make_moons
瑞士卷(三维螺旋)	聚类	n_samples× 3	make_swiss_roll

这些数据集支持自定义设置样本数量、特征维度、类别数量等参数，非常适用于机器学习的入门练习和算法性能测试。

3.3.1　make_regression 数据集生成函数

make_regression 是 scikit-learn 中用于生成回归任务模拟数据的常用函数，它可以

帮助开发者快速构建用于测试和验证回归模型的数据集。其基本语法如下。

```
make_regression(n_samples=100, n_features=100, noise=0.0, coef=False)
```

主要参数说明如下。

- n_samples：生成的样本数量，默认值为 100。
- n_features：每个样本的特征数量，默认值为 100。
- noise：添加到目标变量中的高斯噪声标准差，默认值为 0.0(无噪声)。
- coef：是否返回真实的回归系数(权重)，默认值为 False。

此外，该函数还支持一些其他可选参数。

- n_targets=1：目标变量的数量。
- bias=0.0：线性模型的偏置项。
- shuffle=True：是否打乱生成的样本。
- random_state=None：控制随机数生成器的种子值。

下面是一段完整的示例代码，展示了如何使用 make_regression 生成一维回归数据并进行可视化。

```
>>> from sklearn.datasets import make_regression
>>> import matplotlib.pyplot as plt
#生成数据：200 个样本，每个样本一个特征，并返回真实回归系数 coef
>>> X, y①, coef = make_regression(n_samples=200, n_features=1, noise=10, coef
=True)
#绘制散点图
>>> plt.scatter(X, y, color="b", marker="x", s=80, label="Samples")
#绘制真实回归直线
>>> plt.plot(X, X * coef, color="r", linewidth=3, label="True Regression
Line")
>>> plt.title("Generated Regression Dataset")
>>> plt.xlabel("Feature")
>>> plt.ylabel("Target")
>>> plt.legend()
>>> plt.grid(True)
>>> plt.show()
```

上述代码的输出结果如图 3-19 所示。

3.3.2 make_blobs 数据集生成函数

make_blobs 是 scikit-learn 中用于生成聚类任务模拟数据的重要函数。它可以创建具有多个中心(簇)的高斯分布数据集，适用于测试和可视化聚类算法的效果。其基本语法如下。

```
make_blobs(n_samples=100, n_features=2, centers=None, cluster_std=1.0)
```

① 高斯噪声添加到输出值 y。

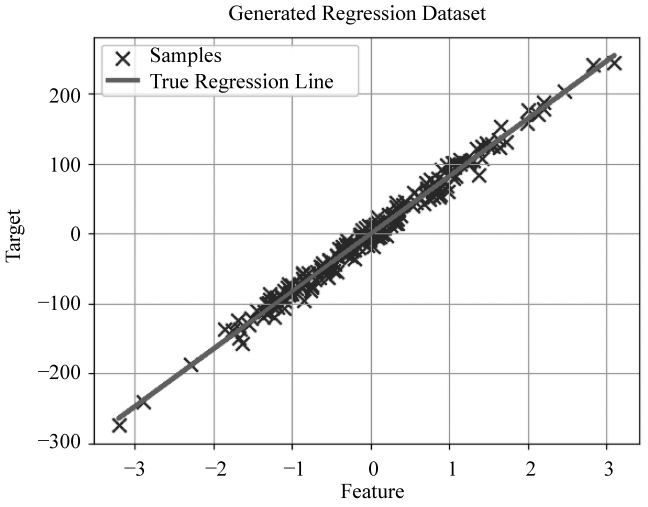

图 3-19　使用 make_regression 生成的一维回归数据

主要参数说明如下。

- centers：指定生成数据的簇中心数量或直接传入自定义的簇中心坐标，默认值为 None。
- cluster_std：各簇的标准差，控制簇内数据点的紧密程度，默认值为 1.0。

此外，该函数还支持其他可选参数。

- center_box：定义簇中心坐标的边界范围。
- return_centers：是否返回生成的簇中心，默认值为 False。

以下是一段完整的示例代码，演示了如何使用 make_blobs 函数生成一个包含 3 个簇的数据集，并通过不同的颜色和形状绘制出来。

```
from sklearn.datasets import make_blobs
plt.xticks(size=14)                          #设置 X 轴刻度标签的字体大小
plt.yticks(size=14)                          #设置 Y 轴刻度标签的字体大小
#生成包含 200 个样本、3 个簇的数据
X, y = make_blobs(n_samples=200, centers=3, random_state=42)
for x, z in zip(X, y):                       #按类别绘制不同颜色和形状的散点
    if z == 0:
        plt.scatter(x[0], x[1], c="r", s=80, marker="o")
    elif z == 1:
        plt.scatter(x[0], x[1], c="g", s=80, marker="x")
    else:
        plt.scatter(x[0], x[1], c="b", s=80, marker="s")
plt.show()
```

上述代码运行后将生成图 3-20 所示的输出结果。

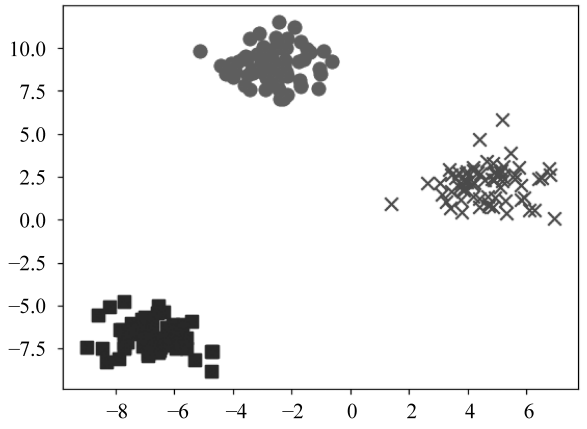

图 3-20　使用 make_blobs 生成一个包含 3 个簇的数据集

3.3.3　make_classification 数据集生成函数

make_ classification 是 scikit-learn 中用于生成分类任务模拟数据的常用函数。它能够创建具有多个类别和可控特征分布的数据集，非常适用于测试和可视化分类模型的效果。其基本语法如下。

```
make_classification(n_samples=100, n_features=20, n_redundant=2, n_classes=2)
```

主要参数说明如下。

- n_redundant：冗余特征的数量，默认值为 2。这些特征是其他特征的线性组合。
- n_classes：目标变量的类别数，默认值为 2（二分类问题），但可以扩展为多分类任务。

此外，该函数还支持多种可选参数。

- n_clusters_per_class：每个类别的簇数量。
- weights：定义各类别的样本比例。

以下是一段完整的示例代码，演示了如何使用 make_classification 函数生成一个包含 3 个类别的二维数据集，并通过不同的颜色和形状进行绘制。

```
from sklearn.datasets import make_classification
plt.xticks(size=12)
plt.yticks(size=12)
X, y = make_classification(n_samples=200, n_features=2, n_redundant=0, \
                    n_clusters_per_class=1, n_classes=3, random_state=0)
for x, z in zip(X, y):          #按类别绘制不同颜色和形状的散点图
    if z == 0:
        plt.scatter(x[0], x[1], c="r", s=80, marker="o")
    elif z == 1:
        plt.scatter(x[0], x[1], c="g", s=80, marker="s")
    else:
        plt.scatter(x[0], x[1], c="b", s=80, marker="x")
```

```
plt.show()
```

上述代码运行后将生成图 3-21 所示的结果。

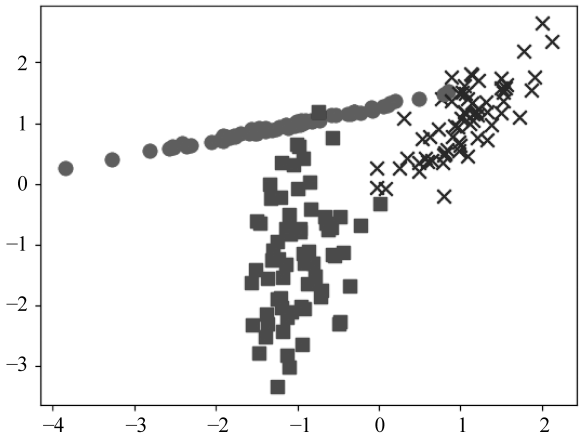

图 3-21　使用 make_classification 生成三分类数据集

3.3.4　make_gaussian_quantiles 数据集生成函数

make_gaussian_quantiles 是 scikit-learn 中用于生成符合多维正态分布的数据集的函数。该数据集特别适用于测试分类算法在不同维度上的表现。其基本语法如下。

```
make_gaussian_quantiles(n_samples=100, n_features=2, mean=None, cov=1.0, n_
classes=3)
```

主要参数说明如下。

- mean：各特征的均值向量。如果未指定(默认值为 None)，则使用零向量。
- cov：样本的协方差矩阵或标量。若为标量，则表示协方差矩阵是单位矩阵乘以此标量。默认值为 1.0。
- n_classes：类别数量，默认值为 3。

以下是一段完整的示例代码，演示了如何使用 make_gaussian_quantiles 函数生成一个包含 3 个类别的二维数据集，并通过不同的颜色和形状进行绘制。

```
from sklearn.datasets import make_gaussian_quantiles
#生成包含 200 个样本、2 个特征、3 个类别的数据集
X, y = make_gaussian_quantiles(n_samples=200, n_features=2, n_classes=3, \
                        mean=[1, 2]①, cov=2, random_state=0)
for x, z in zip(X, y):                          #绘制不同类别的数据点
    if z == 0:
        plt.scatter(x[0], x[1], c="r", s=80, marker="x")
    elif z == 1:
        plt.scatter(x[0], x[1], c="g", s=80, marker="s")
```

① 参数 mean 与 n_features 必须匹配。