29

Data and Data Preprocessing in Scikit-Learn

Scikit-Learn 数据

数据集、缺失值、离群值、特征缩放 ...

三种激情，简单却无比强烈，支配着我的生活：对爱的渴望、对知识的追求以及对人类苦难的无法忍受的怜悯。这些激情，如狂风，将我吹来吹去，任性地，越过痛苦的深海，到了绝望的边缘。

Three passions, simple but overwhelmingly strong, have governed my life: the longing for love, the search for knowledge, and unbearable pity for the suffering of mankind. These passions, like great winds, have blown me hither and thither, in a wayward course, over a deep ocean of anguish, reaching to the very verge of despair.

—— 伯特兰·罗素 (Bertrand Russell) | 英国哲学家、数学家 | 1872 ~ 1970

- `sklearn.covariance.EllipticEnvelope()` 使用基于高斯分布的椭圆包络方法检测异常值
- `sklearn.covariance.mahalanobis()` 计算马哈拉诺比斯距离来检测异常值
- `sklearn.covariance.RobustCovariance()` 使用鲁棒协方差估计进行异常值检测
- `sklearn.datasets.fetch_lfw_people()` 人脸数据集
- `sklearn.datasets.fetch_olivetti_faces()` 奥利维蒂人脸数据集
- `sklearn.datasets.load_boston()` 波士顿房价数据集
- `sklearn.datasets.load_breast_cancer()` 乳腺癌数据集
- `sklearn.datasets.load_diabetes()` 糖尿病数据集
- `sklearn.datasets.load_digits()` 手写数字数据集
- `sklearn.datasets.load_iris()` 鸢尾花数据集
- `sklearn.datasets.load_linnerud()` Linnerud 体能训练数据集
- `sklearn.datasets.load_wine()` 葡萄酒数据集
- `sklearn.datasets.make_blobs()` 生成聚类数据集
- `sklearn.datasets.make_circles()` 生成圆环形状数据集
- `sklearn.datasets.make_classification()` 生成合成的分类数据集
- `sklearn.datasets.make_moons()` 生成月牙形状数据集
- `sklearn.datasets.make_regression()` 生成合成的回归数据集
- `sklearn.ensemble.IsolationForest()` 使用隔离森林方法检测异常值
- `sklearn.impute.IterativeImputer()` 使用多个回归模型来估计缺失值
- `sklearn.impute.KNNImputer()` 使用最近邻样本的值来进行插补
- `sklearn.impute.SimpleImputer()` 提供了一些基本的插补策略来处理缺失值
- `sklearn.neighbors.LocalOutlierFactor()` 使用局部离群因子方法检测异常值
- `sklearn.preprocessing.MaxAbsScaler()` 通过除以每个特征的“最大绝对值”完成特征缩放
- `sklearn.preprocessing.MinMaxScaler()` 通过除以每个特征的“最大值减最小值”完成特征缩放
- `sklearn.preprocessing.PowerTransformer()` 对特征应用幂变换来使数据更加服从高斯分布
- `sklearn.preprocessing.QuantileTransformer()` 将特征转换为均匀分布
- `sklearn.preprocessing.RobustScaler()` 通过减去中位数并除以 IQR 来对特征进行缩放
- `sklearn.preprocessing.StandardScaler()` 标准化特征缩放
- `sklearn.svm.OneClassSVM()` 使用支持向量机方法进行单类异常值检测

29.1 Scikit-Learn 中有关数据的工具

除了完成监督学习、无监督学习之外，Scikit-Learn 还提供了丰富的样本数据集、样本数据生成函数和数据处理方法，用于实现机器学习算法的训练、评估和预测。本章主要介绍如下内容。

- 样本数据集。Scikit-Learn 的样本数据集包含在 sklearn.datasets 模块中，比如 sklearn.datasets.load_iris() 可以用来加载鸢尾花数据集。
- 生成样本数据。Scikit-Learn 还提供数据集生成函数，比如 sklearn.datasets.make_blobs()、sklearn.datasets.make_classification()。
- 特征工程。Scikit-Learn 还提供处理缺失值、处理离群值、特征缩放、数据分割等数据特征工程工具。
- 数据分割。将样本数据划分为训练集和测试集。

29.2 样本数据集

表 1 所示为 Scikit-Learn 中常用数据集。

表 1. Scikit-Learn 常用数据集

函数	介绍
sklearn.datasets.load_boston()	波士顿房价数据集，包含 506 个样本，每个样本有 13 个特征，常用于回归任务。
sklearn.datasets.load_iris()	鸢尾花数据集，包含 150 个样本，每个样本有 4 个特征，常用于分类任务。
sklearn.datasets.load_diabetes()	糖尿病数据集，包含 442 个样本，每个样本有 10 个特征，常用于回归任务。
sklearn.datasets.load_digits()	手写数字数据集，包含 1797 个样本，每个样本是一个 8x8 像素的图像，常用于分类任务。
sklearn.datasets.load_linnerud()	Linnerud 体能训练数据集，包含 20 个样本，每个样本有 3 个特征，常用于多重输出回归任务。
sklearn.datasets.load_wine()	葡萄酒数据集，包含 178 个样本，每个样本有 13 个特征，常用于分类任务。
sklearn.datasets.load_breast_cancer()	乳腺癌数据集，包含 569 个样本，每个样本有 30 个特征，常用于分类任务。
sklearn.datasets.fetch_olivetti_faces()	奥利维蒂人脸数据集，包含 400 张 64x64 像素的人脸图像，常用于人脸识别任务。
sklearn.datasets.fetch_lfw_people()	人脸数据集，包含 13233 张人脸图像，常用于人脸识别和验证任务。

图 1 展示导入 Scikit-Learn 鸢尾花数据所用代码，下面讲解其中关键语句。

a 从 sklearn.datasets 模块导入 load_iris。

b 导入鸢尾花样本数据集对象，将其命名为 iris。

注意，导入数据时，如果采用 X, y = load_iris(as_frame=True, return_X_y=True)，返回的 X 为 Pandas DataFrame，y 为 Pandas Series。请大家自己练习使用这个语句。

c 通过 iris.data 提取鸢尾花数据集的 4 个特征，结果为 NumPy 数组。

d 通过 iris.feature_names 提取鸢尾花 4 个特征名称，结果为 ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']。

e 通过 iris.target 提取鸢尾花数据集的标签，结果也是 NumPy 数组。

f 利用 numpy.unique() 返回独特标签值——0、1、2。

g 利用 iris.target_names 提取鸢尾花问题标签，结果为 ['setosa', 'versicolor', 'virginica']。

h 将鸢尾花前 4 个特征 NumPy 数组创建成 Pandas 数据帧。 i 用 describe() 对数据帧做统计汇总，结果如表 2 所示。

```
a from sklearn.datasets import load_iris
  import numpy as np
  import pandas as pd
  # 导入鸢尾花数据
b iris = load_iris()
  # 鸢尾花数据前4个特征，NumPy数组
c X = iris.data
d print(iris.feature_names)
  # 鸢尾花数据标签：0、1、2
e y = iris.target
f print(np.unique(y))
  # 鸢尾花文字标签
g print(iris.target_names)
  # 创建数据帧
h X_df = pd.DataFrame(X,
                      columns = ['X1','X2','X3','X4'])
i round(X_df.describe(),2)
```

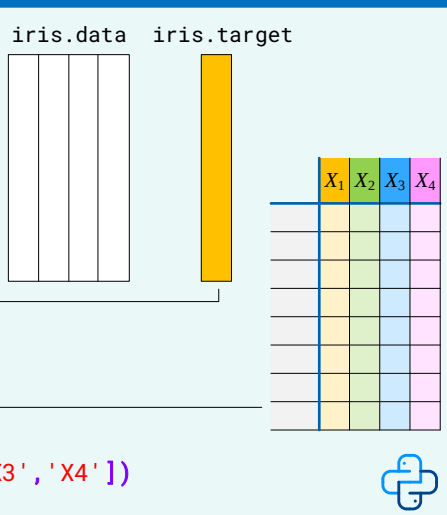

图 1. 导入 Scikit-Learn 中鸢尾花数据

表 2. 鸢尾花数据集的统计总结

	X_1, sepal length (cm)	X_2, sepal width (cm)	X_3, petal length (cm)	X_4, petal width (cm)
count	150	150	150	150
mean	5.84	3.06	3.76	1.20
std	0.83	0.44	1.77	0.76
min	4.30	2.00	1.00	0.10
25%	5.10	2.80	1.60	0.30
50%	5.80	3.00	4.35	1.30
75%	6.40	3.30	5.10	1.80
max	7.90	4.40	6.90	2.50

29.3 生成数据

表 3 总结 Scikit-Learn 中常用来生成样本数据集的函数。图 2 所示为表 3 中一些函数生成的样本数据集。图中颜色代表不同分类标签。图 4 为对应代码，下面介绍其中重要语句。

表 3. Scikit-Learn 中常用来生成样本数据集函数

sklearn.datasets.make_regression()	生成合成的回归数据集，下一章将会用到这个函数。
sklearn.datasets.make_classification()	生成合成的分类数据集，可以指定样本数、特征数、类别数等。
sklearn.datasets.make_blobs()	生成聚类数据集，可以指定样本数、特征数、簇数等
sklearn.datasets.make_moons()	生成月牙形状数据集
sklearn.datasets.make_circles()	生成圆环形状数据集

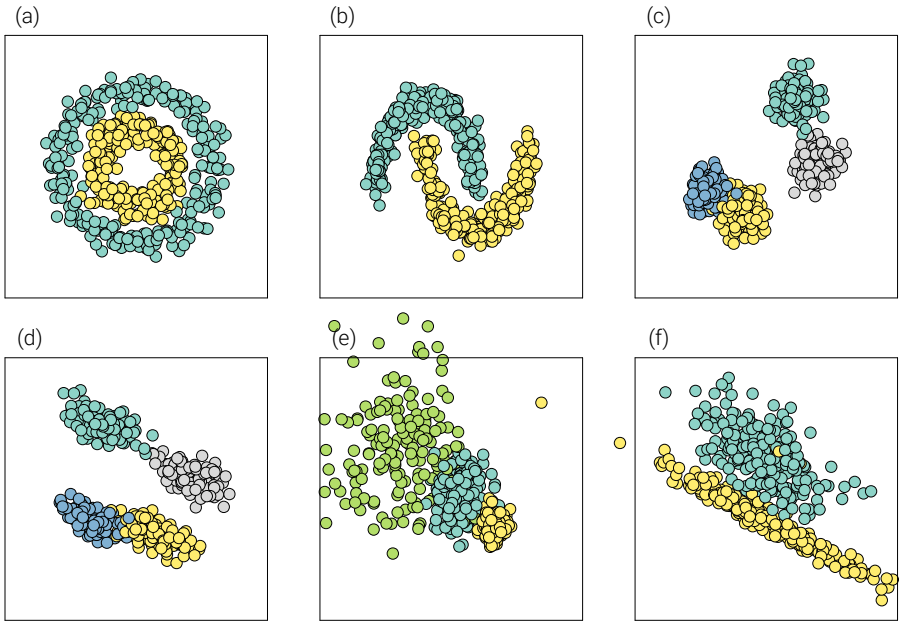

图 2. 生成样本数据集，有标签

a 从 sklearn.preprocessing 模块导入 StandardScaler()。StandardScaler()是 scikit-learn 中的一个预处理类，用于在机器学习流程中对数据进行标准化处理。标准化是数据预处理的一种常见方式，目的是将数据的特征值缩放成均值为 0，标准差为 1 的分布，即计算 Z 分数，以消除不同特征之间的尺度差异。本章后文将介绍更多预处理方法。

b 中 sklearn.datasets.make_circles() 生成环形数据集的函数，结果如图 2 (a) 所示。数据点位于两个同心圆上，可以用于测试机器学习算法。参数 n_samples 设定数据点数量，默认为 100。参数 noise 为添加到数据中的高斯噪声的标准差。参数 factor 为内外圆之间的比例因子。factor 取值在 0 到 1 之间，0.0 表示两个圆重叠，1.0 表示完全分离的两个圆。

c 中 sklearn.datasets.make_moons() 用于生成月牙形状的数据集，结果如图 2 (b) 所示。这个函数可以用于测试在非线性数据上表现良好的算法。参数 n_samples 指定生成的数据点数量。参数 noise 指定添加到数据中的高斯噪声的标准差。

d 中 sklearn.datasets.make_blobs() 生成一个由多个高斯分布组成的数据集，结果如图 2 (c) 所示。参数 n_samples 为生成的样本数。参数 n_features 为每个样本的特征数。参数 centers 是要生成的数据的质心数量，或高斯分布质心的具体位置。参数 cluster_std 为每个聚类的标准差，用于控制每个聚类中数据点的分布紧密程度。

e 对 sklearn.datasets.make_blobs() 生成的数据集进行几何变换 (缩放 + 旋转)，结果如图 2 (d) 所示。大家要是想知道具体的几何变换，需要采用特征值分解。

f 在利用 sklearn.datasets.make_blobs() 时，每个高斯分布指定不同的标准差，结果如图 2 (e) 所示。

g 中 sklearn.datasets.make_classification() 生成一个虚拟的分类数据集，可以用于测试和演示分类算法，结果如图 2 (f) 所示。

h 采用 2 行 3 列子图布局可视化上述样本数据集。i 利用前文导入的 StandardScaler() 对 $\boldsymbol{X}$ 标准化。标准化是特征缩放的一种。在机器学习中，特征缩放是一个重要的预处理步骤，其目的是为了在不同特征之间建立更好的平衡，以便模型能够更好地进行学习和预测。表 4 总结 Scikit-Learn 中常用特征缩放函数。

表 4. Scikit-Learn 中常用特征缩放的函数

函数	介绍
sklearn.preprocessing.MaxAbsScaler()	通过除以每个特征的"最大绝对值"来将特征缩放到 [-1, 1] 的范围内，保留了特征的正负关系，助于防止异常值对数据缩放的影响。
sklearn.preprocessing.MinMaxScaler()	通过除以每个特征的"最大值减最小值"将特征缩放到指定范围之内，默认范围为 (0, 1)。它可以保留特征之间的线性关系，适用于受异常值影响较小的数据。
sklearn.preprocessing.Normalizer()	将样本行向量缩放到单位范数 (默认是 L2 范数) 的方法。适用于特征的大小不重要，而只关心方向的情况。
sklearn.preprocessing.PowerTransformer()	对特征应用幂变换来使数据更加服从高斯分布。它支持 Yeo-Johnson 和 Box-Cox 变换，用于处理不符合正态分布的数据。
sklearn.preprocessing.QuantileTransformer()	将特征转换为均匀分布，从而使得变换后的数据服从指定的分位数。可以用来减少离群值的影响，特别是在数据分布不均匀的情况下。
sklearn.preprocessing.RobustScaler()	通过减去中位数并除以 IQR 来对特征进行缩放。本书前文提过，IQR = $Q_3 - Q_1$。这种特征缩放对异常值具有鲁棒性，不会受到异常值的影响。适用于数据包含许多离群值的情况。
sklearn.preprocessing.StandardScaler()	StandardScaler 通过将特征缩放到均值为 0，方差为 1 的标准正态分布来进行标准化。它适用于要求输入数据具有相似的尺度的机器学习算法。

上述函数生成的数据集如果不考虑标签的话，也可以用于测试聚类算法，如图 3 所示。

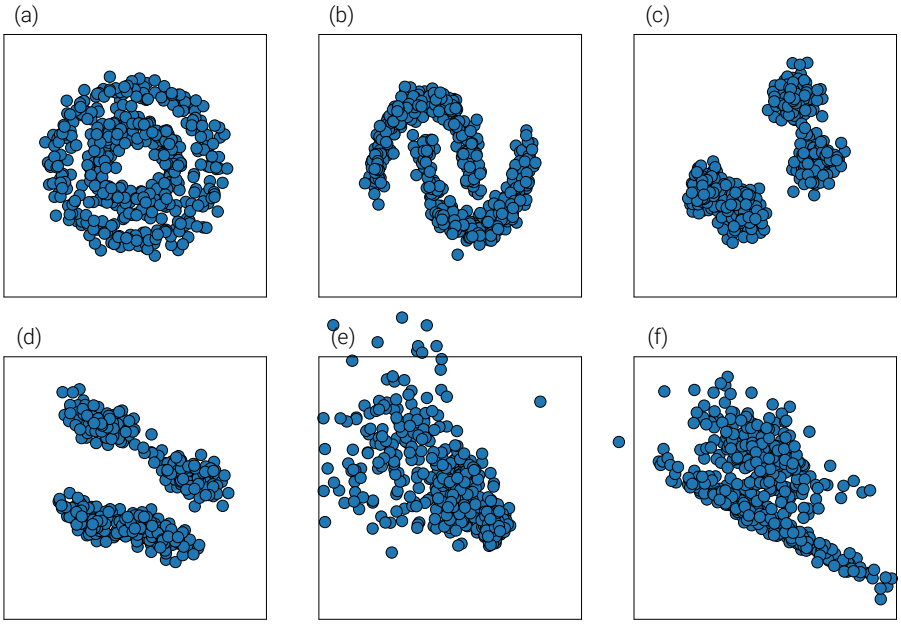

图 3. 生成样本数据集，无标签

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_circles, make_moons
from sklearn.datasets import make_blobs, make_classification

n_samples = 500
# 产生环形数据集
circles = make_circles(n_samples=n_samples,
                       factor=0.5, noise=0.1)
# 产生月牙形状数据集
moons = make_moons(n_samples=n_samples,
                   noise=0.1)
#
blobs = make_blobs(n_samples=n_samples,
                   centers = 4,
                   cluster_std = 1.5)
# 几何变换
transformation = [[0.4, 0.2], [-0.4, 1.2]]
X = np.dot(blobs[0], transformation)
rotated = (X,blobs[1])
# 不同稀疏程度
varied = make_blobs(n_samples=n_samples,
                    cluster_std=[1.0, 2.5, 0.5])
# 用于测试分类算法的样本数据集
classif = make_classification(n_samples=n_samples,
                              n_features=2,
                              n_redundant=0,
                              n_informative=2,
                              n_clusters_per_class=1)

datasets = [circles, moons, blobs, rotated, varied, classif]

# 可视化
fig, axes = plt.subplots(2,3,figsize=(6,4))
axes = axes.flatten()

for dataset_idx, ax_idx in zip(datasets, axes):

    X, y = dataset_idx
    # 标准化
    X = StandardScaler().fit_transform(X)
    ax_idx.scatter(X[:, 0], X[:, 1], s=18,
               c=y, cmap='Set3',
               edgecolors="k")

    ax_idx.set_xlim(-3, 3)
    ax_idx.set_ylim(-3, 3)
    ax_idx.set_xticks(())
    ax_idx.set_yticks(())
    ax_idx.set_aspect('equal', adjustable='box')
```

图 4. 生成样本数据集，代码

29.4 处理缺失值

在数据分析中，缺失值是指数据集中某些观测值或属性值没有被记录或采集到的情况。由于各种原因，数据中缺失值不可避免。缺失值通常被编码为空白，NaN 或其他占位符 (比如−1)。处理缺失值是数据预处理中重要一环。

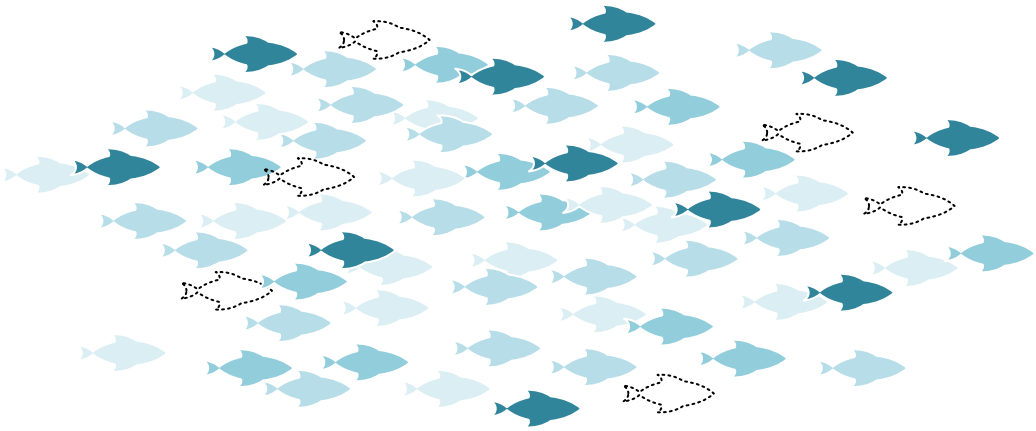

图 5. 缺失值

数据中缺失值产生的原因有很多。比如，在数据采集阶段，设备故障、人为失误、方法局限、拒绝参与调查、信息不完整等等可以造成数据缺失。另外，数据数据存储阶段也可能引入缺失值；比如，数据存储失败、存储器故障等等。

填补缺失值的方法有很多种，包括：

- **删除缺失值**：直接删除缺失值所在的行或列，但这可能会导致数据的丢失和分析结果的偏差。
- **插值法**：通过插值方法填补缺失值，如均值插值、中位数插值、最近邻插值、多项式插值等。
- **模型法**：使用回归、决策树或神经网络等模型预测缺失值，但需要先对数据进行训练和测试，可能会导致模型的过拟合和不准确。
- **多重填补法**：使用多个模型进行填补，可以提高填补缺失值的准确性和可靠性。

本书前文在介绍 Pandas 时，我们了解了一些 Pandas 中处理缺失值的方法。表 5 所示为 Scikit-Learn 中常用处理缺失值方法。需要注意的是，表 5 中方法通常用于数值型数据。

表 5. Scikit-Learn 中常用来处理缺失值的函数

函数	介绍
sklearn.impute.SimpleImputer()	提供了一些基本的插补策略来处理缺失值，例如使用均值、中位数、众数进行插补
sklearn.impute.IterativeImputer()	使用多个回归模型来估计缺失值，每次迭代都更新缺失值的估计
sklearn.impute.KNNImputer()	使用最近邻样本的值来进行插补。它使用欧氏距离或其他指定的距离度量来选择最近邻

下面用图 8 介绍如何使用最邻近插补。

图 6. 鸢尾花数据集中引入缺失值，每条黑带代表缺失值位置

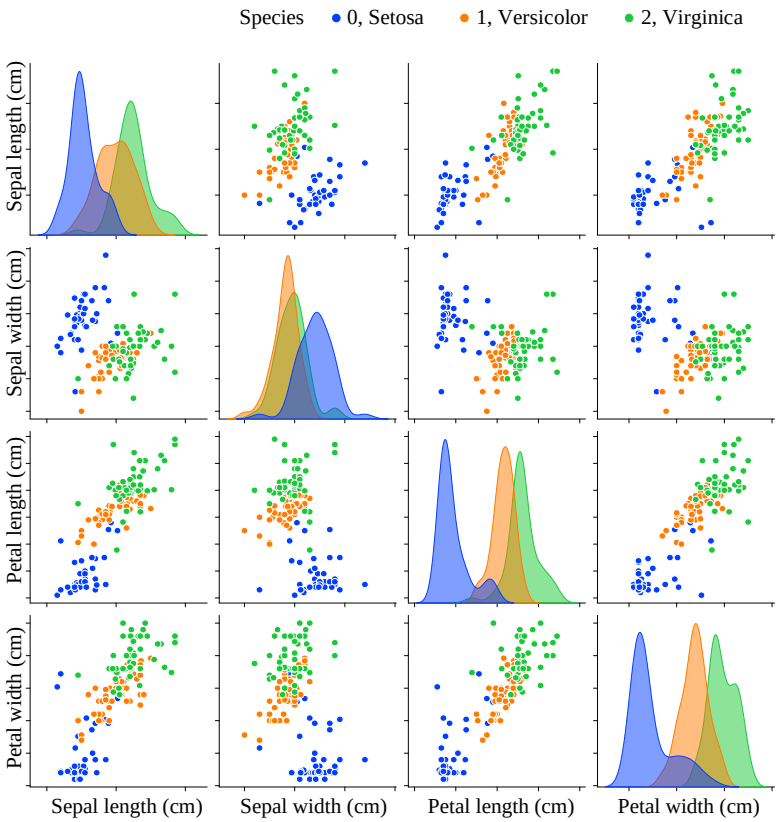

图 7. 鸢尾花数据，最近邻插补

a 从 sklearn.impute 模块导入 KNNImputer() 函数。KNNImputer() 完成 *k* 近邻插补。**k 近邻算法** (*k*-nearest neighbors algorithm, *k*-NN) 是最基本有监督学习方法之一，*k*-NN 中的 *k* 指的是“近邻”的数量。*k*-NN 思路很简单——“近朱者赤，近墨者黑”。本书后文将介绍这种算法。

b 利用 numpy.random.uniform() 产生 [0, 1) 之间连续均匀随机数 NumPy 数组，数组形状和鸢尾花特征数据形状一致。

c 将原先生成的随机数数组 mask 中小于等于 0.4 的元素标记为 True，其余元素标记为 False。这样，mask 数组中的元素将形成一个“面具” (布尔掩码)，用来选择哪些位置将被置为缺失值。

d 将 X_NaN 数组中根据 mask 中对应位置为 True 的元素，设置为缺失值 (NaN)。换句话说，该代码将 X_NaN 数组中部分元素置为缺失值，而其他元素保持不变。

为了准确获取缺失值位置、数量等信息，对于 Pandas 数据帧数据可以采用 isna() 或 notna() 方法。

e 采用 iris_df_NaN.isna()，返回具体位置数据是否为缺失值。数据缺失的话，为 True；否则，为 False。sklearn.impute.MissingIndicator() 也可以用来获取缺失值位置。

f 为采用 seaborn.heatmap() 可视化数据缺失值，图 6 所示热图的每一条黑色条带代表一个缺失值。使用缺失值热图可以粗略观察得到缺失值分布情况。

g 创建了一个 KNNImputer 对象，用于执行 *k* 最近邻插补。参数 n_neighbors 指定了在插补过程中要考虑的最近邻样本的数量。

h 将 KNNImputer 应用于具有缺失值的数据数组 X_NaN。fit_transform() 方法将执行两个步骤：拟合 (fit) 和转换 (transform)。拟合时，KNNImputer 将根据已知数据 (非缺失值) 来训练最近邻模型。转换时，使用训练过的模型，KNNImputer 将执行 k 最近邻插补，将缺失值填充为预测的值。KNNImputer 返回结果被存储在 X_NaN_kNN 中，其中包含了插补后的数据。

h 用 seaborn.pairplot() 绘制成对散点图可视化插补后结果，如图 7 所示。

```
from sklearn.datasets import load_iris
from sklearn.impute import KNNImputer
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
# 导入鸢尾花数据
X, y = load_iris(as_frame=True, return_X_y=True)

# 引入缺失值
X_NaN = X.copy()
mask = np.random.uniform(0,1,size = X_NaN.shape)
mask = (mask <= 0.4)
X_NaN[mask] = np.NaN

iris_df_NaN = X_NaN.copy()
iris_df_NaN['species'] = y

# 可视化缺失值位置
is_NaN = iris_df_NaN.isna()
print(iris_df_NaN.isnull().sum() * 100 / len(iris_df_NaN))

fig, ax = plt.subplots()
ax = sns.heatmap(is_NaN,
                 cmap='gray_r',
                 cbar=False)

# 用kNN插补
knni = KNNImputer(n_neighbors=5)
X_NaN_kNN = knni.fit_transform(X_NaN)

iris_df_kNN = pd.DataFrame(X_NaN_kNN, columns=X_NaN.columns,
                           index=X_NaN.index)
iris_df_kNN['species'] = y

sns.pairplot(iris_df_kNN, hue='species',
             palette = "bright")
```

图 8. 处理缺失值，代码

29.5 处理离群值

离群值 (outlier)，又称逸出值、离群值，是指数据集中与其他数据点有显著差异的数据点，也就是说明显地偏大或偏小。离群值可能是由于异常情况、错误测量、数据录入错误或意外事件等原因而产生。离群值可能会对数据分析和建模造成问题，因为它们可能导致误差或偏差，并降低模型的准确性。因此，数据分析师通常会对数据集中的离群值进行检测和处理。

常见的离群值检测方法包括基于统计学的方法、基于距离的方法、基于密度的方法和基于模型的方法。处理离群值的方法包括删除、替换、调整或利用异常值建立新的模型等。

表 6 所示为 Scikit-Learn 中常用处理离群值函数。

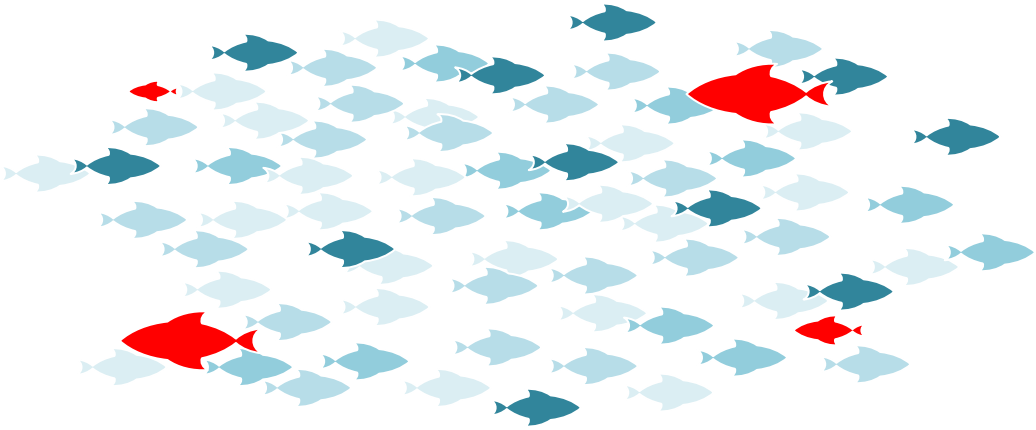

图 9. 离群点

表 6. Scikit-Learn 中常用来处理离群值的函数

函数	介绍
sklearn.ensemble.IsolationForest()	使用隔离森林方法检测异常值
sklearn.svm.OneClassSVM()	使用支持向量机方法进行单类异常值检测
sklearn.covariance.EllipticEnvelope()	使用基于高斯分布的椭圆包络方法检测异常值
sklearn.neighbors.LocalOutlierFactor()	使用局部离群因子方法检测异常值
sklearn.covariance.RobustCovariance()	使用鲁棒协方差估计进行异常值检测
sklearn.covariance.mahalanobis()	计算马哈拉诺比斯距离来检测异常值

图 11 所示代码介绍如何使用 Scikit-Learn 处理离群值。这段代码参考了 Scikit-Learn 官方示例。

a 从 sklearn.svm 模块中导入 OneClassSVM 类，该类实现支持向量机 (Support Vector Machine, SVM) 中的单类异常值检测方法。本书后续将专门介绍支持向量机。

b 从 sklearn.covariance 模块中导入 EllipticEnvelope 类，该类实现基于高斯分布的椭圆包络方法，用于检测异常值。椭圆包络假设正常数据点是从多元高斯分布中产生，然后构建一个椭圆来包围正常数据点，从而将异常数据点识别为离这个椭圆很远的点。

c 从 sklearn.ensemble 导入 IsolationForest 类，该类实现隔离森林 (Isolation Forest) 方法，用于检测异常值。隔离森林利用随机分割数据来构建一棵或多棵树，并通过观察数据点在树中的深度来确定异常值。

d 定义了一个名为 blobs_params 的字典，其中包含了一些参数设置。random_state=0 用于控制随机数生成的种子值。n_samples=n_inliers 控制生成的总样本数。n_features=2 设定每个数据点的特征数量为 2，即两个特征。

e 构造了 4 组数据集。

f 用 EllipticEnvelope() 创建椭圆包络的异常值检测模型。参数 contamination 用于指定异常值的比例。具体来说，它表示数据中异常值的比例。这个参数是一个介于 0 和 0.5 之间的值，通常需要根据具体问题进行调整。参数 random_state 用于控制随机数生成的种子值，以确保每次运行得到相同的结果。

g 使用 OneClassSVM() 创建一个基于支持向量机的异常值检测模型。参数 nu 用于指定异常值的比例，通常在 0 和 1 之间。kernel="rbf" 指定支持向量机所使用的核函数的类型。"rbf" 表示径向基函数 (Radial Basis Function)，也称为高斯核。这个核函数在支持向量机中常用于处理非线性问题。gamma=0.1 是支持向量机模型的核函数参数。较小的 gamma 值会使得支持向量具有更远的影响范围，可能会导致决策边界更平滑；较大的 gamma 值则会使支持向量的影响范围更小，可能会导致决策边界更复杂。

h 使用 IsolationForest() 创建一个基于隔离森林的异常值检测模型。

i 使用 fit() 方法对样本数据进行拟合，然后使用 predict() 方法来预测数据点是否为异常值。

j 用平面等高线可视化异常值检测模型的决策边界。

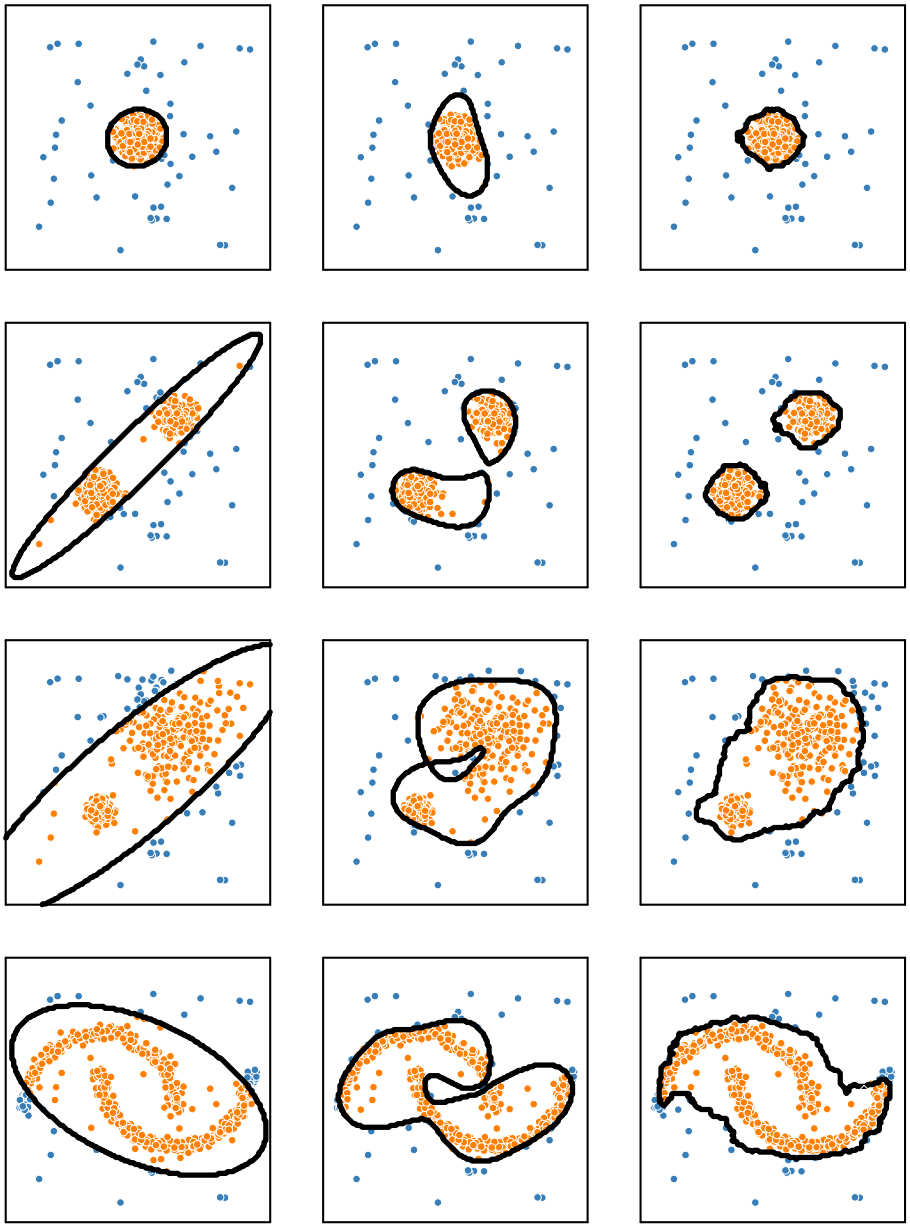

图 10. 用 Scikit-Learn 判断离群点

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import make_blobs, make_moons
from sklearn.svm import OneClassSVM
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest

# 生成数据
n_samples = 500
outliers_fraction = 0.10
n_outliers = int(outliers_fraction * n_samples)
n_inliers  = n_samples - n_outliers
X_outliers = np.random.uniform(low=-6,high=6,
                               size=(n_outliers,2))

np.random.RandomState(0)
blobs_params = dict(random_state=0,
                    n_samples=n_inliers, n_features=2)
datasets = [
    make_blobs(centers=[[0, 0], [0, 0]],
               cluster_std=0.5, **blobs_params)[0],
    make_blobs(centers=[[2, 2], [-2, -2]],
               cluster_std=[0.5, 0.5], **blobs_params)[0],
    make_blobs(centers=[[2, 2], [-2, -2]],
               cluster_std=[1.5, 0.3], **blobs_params)[0],
    4.0 * (make_moons(n_samples=n_samples, noise=0.05,
           random_state=0)[0]- np.array([0.5, 0.25]))]

# 处理离群值
anomaly_algorithms = [
     EllipticEnvelope(contamination=outliers_fraction,
                      random_state=42),
     OneClassSVM(nu=outliers_fraction, kernel="rbf",
                 gamma=0.1),
     IsolationForest(contamination=outliers_fraction,
                     random_state=42)]

# 网格化数据，用来绘制等高线
xx, yy = np.meshgrid(np.linspace(-7, 7, 150),
                     np.linspace(-7, 7, 150))
xy = np.c_[xx.ravel(), yy.ravel()]
colors = np.array(["#377eb8", "#ff7f00"])

# 可视化
fig = plt.figure(figsize=(8,12))
plot_idx = 1
for idx, X in enumerate(datasets):
    X = np.concatenate([X, X_outliers], axis=0)

    for algorithm in anomaly_algorithms:
        algorithm.fit(X)
        y_pred = algorithm.fit(X).predict(X)

        ax = fig.add_subplot(4,3,plot_idx); plot_idx += 1
        Z = algorithm.predict(xy)
        Z = Z.reshape(xx.shape)
        # 绘制边界
        ax.contour(xx, yy, Z, levels=[0],
                   linewidths=2, colors="black")
        # 绘制散点数据集
        ax.scatter(X[:, 0], X[:, 1], s=10,
                   color=colors[(y_pred + 1) // 2])
        ax.set_xlim(-7, 7); ax.set_ylim(-7, 7)
        ax.set_xticks(()); ax.set_yticks(())
```

图 11. 处理离群值，代码

29.6 训练集 vs 测试集

在机器学习中，训练集和测试集是用于训练和评估模型性能的两个关键数据集。Scikit-Learn 库提供了工具和函数来处理和划分这些数据集。

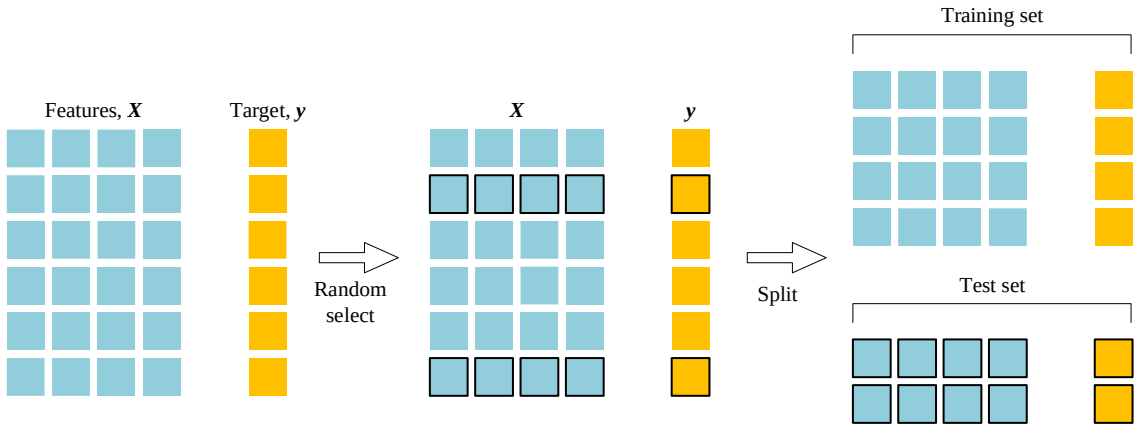

图 12. 拆分数据集为训练集和测试集

训练集 (training set) 是用来训练机器学习模型的数据集。模型在训练集上学习数据的模式、关系和特征，以便能够做出预测 (回归、降维、分类、聚类等等)。训练集通常包含已知的输入特征和对应的目标输出，用于模型进行学习和参数调整。

测试集 (test set) 是用于评估机器学习模型性能的数据集。一旦模型在训练集上进行了学习，它需要在测试集上进行预测，以便判断模型在未见过的数据上的表现如何。测试集应该是与训练集相互独立的样本，以确保对模型的泛化能力进行准确评估。

在划分数据集时，常见的做法是将大部分数据用于训练 (例如 80%)，少部分用于测试 (例如 20%)。通过在测试集上评估模型的性能，可以获得模型在真实环境中的表现，并帮助检测过拟合等问题。图 13 所示为将鸢尾花数据集拆分为训练集和测试集。

图 14 代码完成数据拆分以及可视化，下面介绍其中关键语句。

ⓐ从 sklearn.model_selection 模块导入 train_test_split。train_test_split 将数据集划分为训练集和测试集，以便在机器学习模型的开发和评估过程中使用。train_test_split 函数的作用是将输入的数据集 (通常是特征矩阵和对应的标签向量) 分成两个部分：一个用于训练模型，另一个用于评估模型的性能。这是为了确保模型在未见过的数据上表现良好，以避免过拟合。

ⓑ用 train_test_split 函数将输入的数据集 X 和 y 划分为训练集和测试集，并将划分后的数据分别赋值给了 X_train、X_test、y_train 和 y_test 四个变量。

X 为输入的特征矩阵，包含样本的特征信息。

y 为输入的标签向量，包含与特征对应的目标值。

参数 test_size 为 0.2，表示将数据的 20% 作为测试集，剩余 80% 作为训练集。这个参数决定了训练集和测试集的划分比例。

X_train 为训练集的特征矩阵，包含用于训练机器学习模型的特征数据。

X_test 为测试集的特征矩阵，包含用于评估模型性能的特征数据。

y_train 为训练集的标签向量，包含训练集样本对应的目标值。

y_test 为测试集的标签向量，包含测试集样本对应的目标值。

c 创建一个包含 1 行、2 列的子图布局。gridspec_kw={'width_ratios': [4, 1]} 参数用于控制每个子图的宽度比例，这里设置了第一个子图的宽度为第二个子图的 4 倍。

d 将 np.c_[X,y] 转化成 Pandas DataFrame，以便后续可视化。e 将 np.c_[X_train, y_train] 转化为训练集 Pandas DataFrame。f 将 np.c_[X_test, y_test] 转化为测试集 Pandas DataFrame。

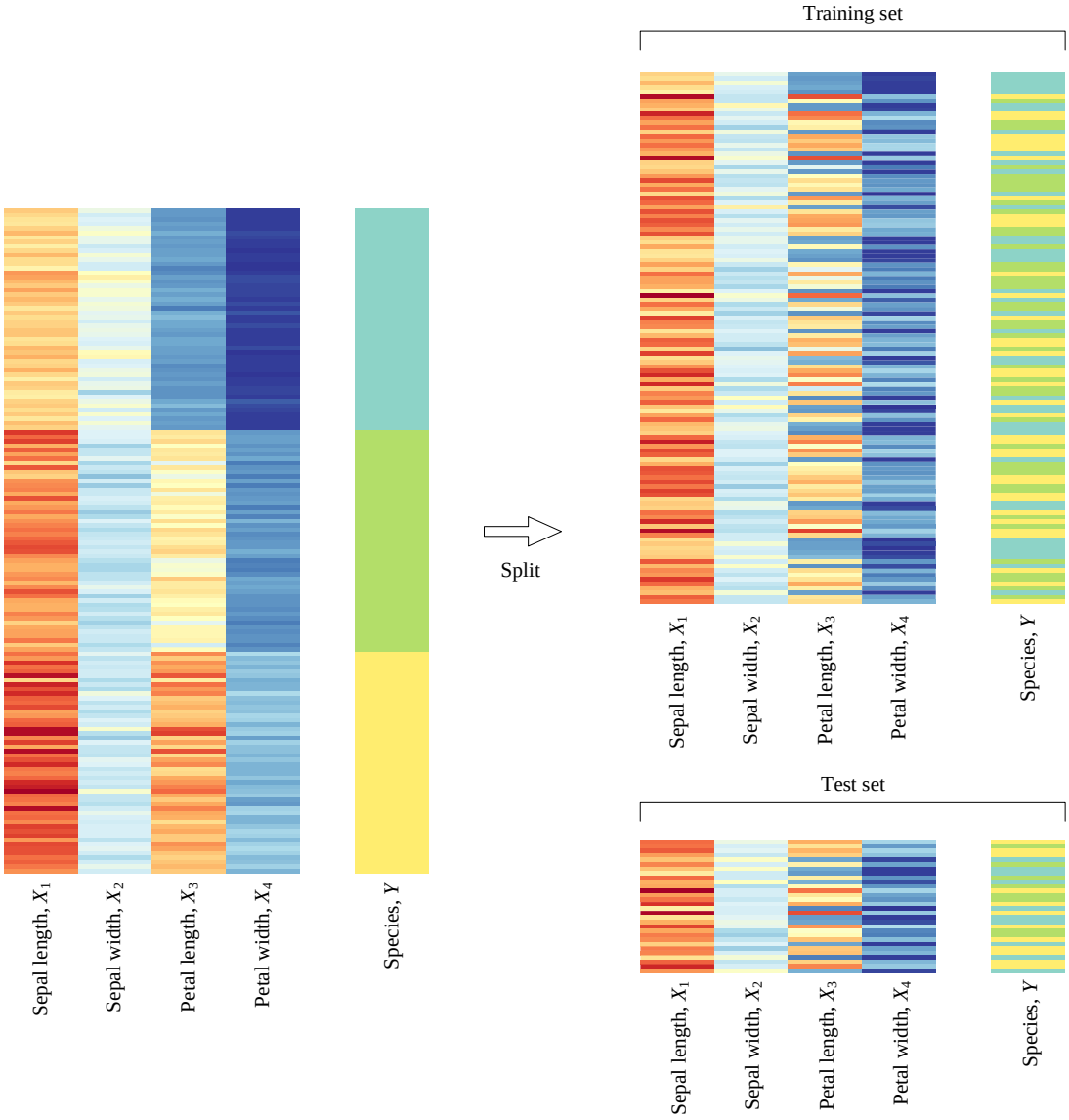

图 13. 拆分鸢尾花数据集为训练集和测试集

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns

# 导入鸢尾花数据
X,y = load_iris(return_X_y=True)
# 拆分鸢尾花数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2)

# 自定义可视化函数
def visualize(df):
    fig, axs = plt.subplots(1, 2,
        gridspec_kw={'width_ratios': [4, 1]})

    sns.heatmap(df.iloc[:,0:-1],
                cmap='RdYlBu_r', yticklabels = False,
                cbar=False, ax = axs[0])

    sns.heatmap(df.iloc[:,[-1]],
                cmap='Set3', yticklabels = False,
                cbar=False, ax = axs[1])

# 转化为Pandas DataFrame
columns = ['Sepal length, X1', 'Sepal width, X2',
           'Petal length, X3', 'Petal width, X4',
           'Species']
df_full = pd.DataFrame(np.c_[X,y],
                       columns = columns)
visualize(df_full)

# 训练集
df_train = pd.DataFrame(np.c_[X_train, y_train],
                        columns = columns)
visualize(df_train)

# 测试集
df_test = pd.DataFrame(np.c_[X_test, y_test],
                       columns = columns)
visualize(df_test)
```

图 14. 拆分鸢尾花数据集为训练集和测试集，代码