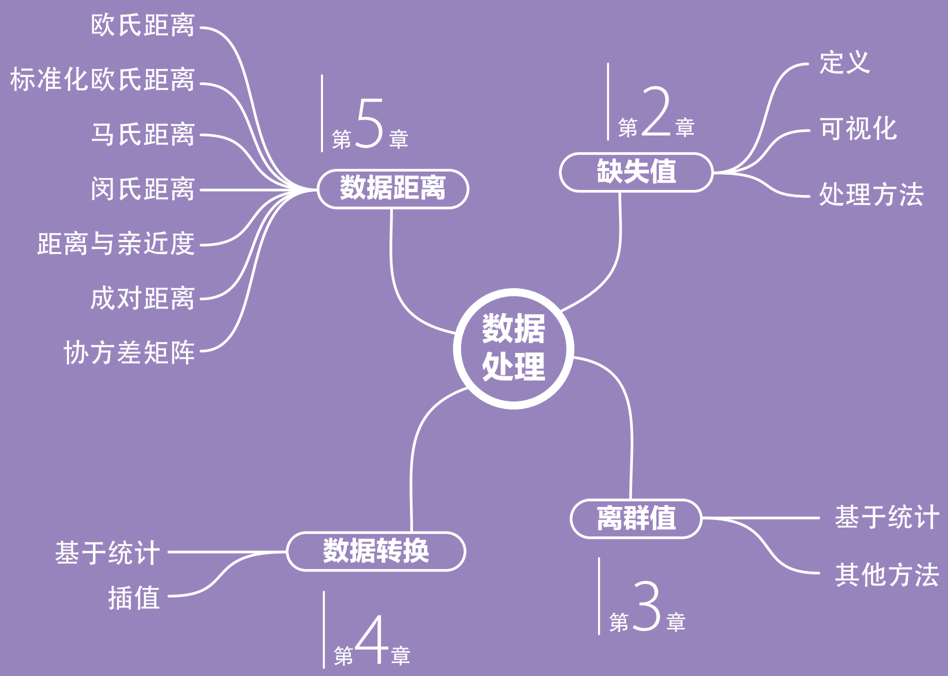

数据处理
第5章
数据距离
欧氏距离
标准化欧氏距离
马氏距离
闵氏距离
距离与亲近度
成对距离
协方差矩阵
第2章
缺失值
定义
可视化
处理方法
第3章
离群值
基于统计
其他方法
第4章
数据转换
基于统计
插值

学习地图 | 第2板块

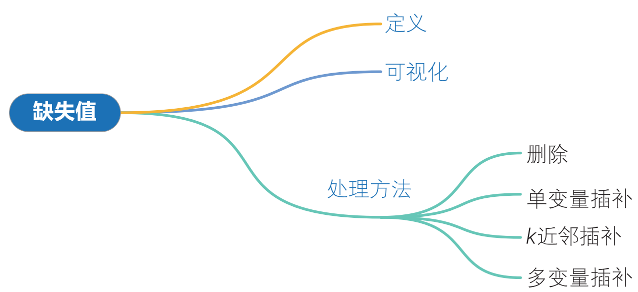
02 Dealing with Missing Data
缺失值
用代数、统计、机器学习算法补齐缺失值

若上天再给一次机会，让我重新开始学业，我定会听从柏拉图，先学数学。

If I were again beginning my studies, I would follow the advice of Plato and start with mathematics.

—— 伽利略·伽利莱 (Galileo Galilei) | 意大利物理学家、数学家及哲学家 | 1564 — 1642年

- `df.dropna(axis = 0, how = 'any')` 中 `axis = 0` 为按行删除，设置 `axis = 1`表示按列删除。`how = 'any'`时，表示某行或列只要有一个缺失值，就删除该行或列；当`how = 'all'`，表示该行或列全部都为缺失值时，才删除该行或列
- `df.isna()` 判断Pandas数据帧是否为缺失值，是使用`True`占位，否使用`False`占位
- `df.notna()` 判断Pandas数据帧是否为非缺失值，是缺失值使用`False`占位，不是缺失值采用`True`占位
- `missingno.matrix()` 绘制缺失值热图
- `numpy.NaN` 产生NaN占位符
- `numpy.random.uniform()` 产生满足连续均匀分布的随机数
- `seaborn.heatmap()` 绘制热图
- `seaborn.pairplot()` 绘制成对特征分析图
- `sklearn.impute.KNNImputer()` 使用k近邻插补
- `sklearn.impute.MissingIndicator()` 将数据转换为相应的二进制矩阵 (`True`和`False`)，以指示数据中缺失值的存在位置
- `sklearn.impute.SimpleImputer()` 使用缺失值所在的行/列中的统计数据平均值 (`'mean'`)、中位数 (`'median'`) 或者众数 (`'most_frequent'`) 来填充，也可以使用指定的常数 `'constant'`



2.1 是不是缺了几个数?

由于各种原因，数据中产生缺失值是不可避免的。缺失值通常被编码为空白、NaN或其他占位符。处理缺失值是数据预处理中重要一环。示意图如图2.1所示。

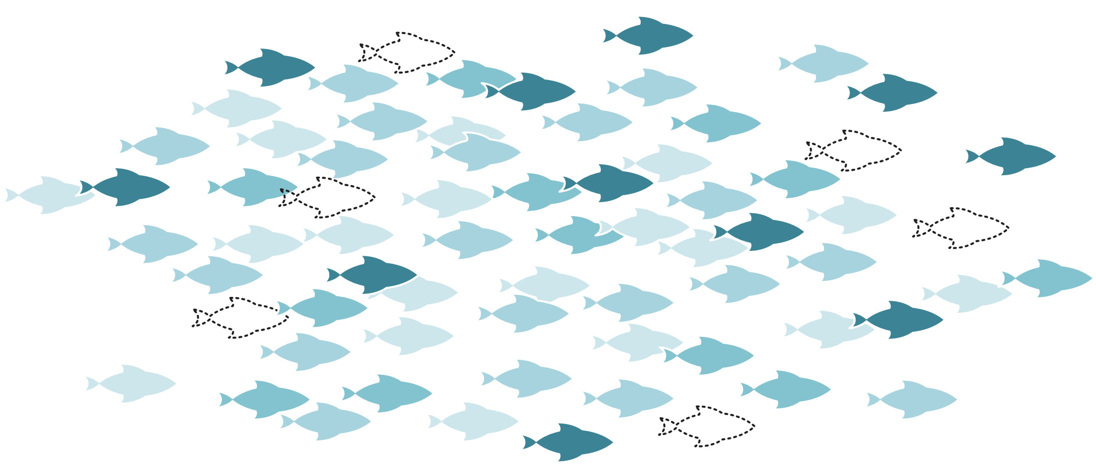


图2.1　缺失值示意图

数据中缺失值产生的原因有很多。比如，在数据采集阶段，人为失误、方法局限等等可以造成数据缺失。另外，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。

三大类

缺失值大致分为三类：

- **完全随机缺失** (Missing Completely at Random，MCAR)：缺失值和自身值无关，且和其他任何变量无关。
- **随机缺失** (Missing at Random，MAR)：其他特征存在数据，但是某个特征缺失值和自身值无关。一个经典例子是，人们是否透露收入可能与性别、教育或职业等因素存在某种联系，而非收入高低。
- **非随机缺失** (Missing Not at Random，MNAR)：数据缺失可能与数据自身值存在一定关系，比如高收入群体不希望透露他们的收入。

NaN

NaN常用于表示缺失值。NaN是not a number的缩写，中文含义是“非数”。numpy.nan可以用来产生NaN。举个例子，如果想要在已知数据帧df中，增加用NaN做占位符一列，就可以用df['holder'] = np.nan，其中“holder”为这一列的**标题** (header)。

一些NumPy函数在统计计算时，遇到缺失值会报错。表2.1第二列NumPy函数在遇到缺失值NaN时，会直接报错。而表2.1第三列函数在计算时，会忽略NaN。

表2.1　比较Numpy函数处理缺失值差异

	遇到NaN，报错	计算时，忽略NaN
均值	numpy.mean()	numpy.nanmean()
中位数	numpy.median()	numpy.nanmedian()
最大值	numpy.max()	numpy.nanmax()
最小值	numpy.min()	numpy.nanmin()
方差	numpy.var()	numpy.nanvar()
标准差	numpy.std()	numpy.nanstd()
分位	numpy.quantile()	numpy.nanquantile()
百分位	numpy.percentile()	numpy.nanpercentile()

原始数据中缺失值的样式没有特定标准，利用Pandas读取数据时，可以设置缺失值样式。比如read_csv() 读取CSV文件时，可以利用 na_values 设置缺失值样式，比如na_values = 'Null'，再如 na_values = '?'，等等。在Pandas数据帧中，也用NaT表达缺失值。

以鸢尾花数据为例

本章以鸢尾花数据讲解如何处理缺失值。图2.2所示为完整的鸢尾花数据成对特征分析图，其中有150个数据点。

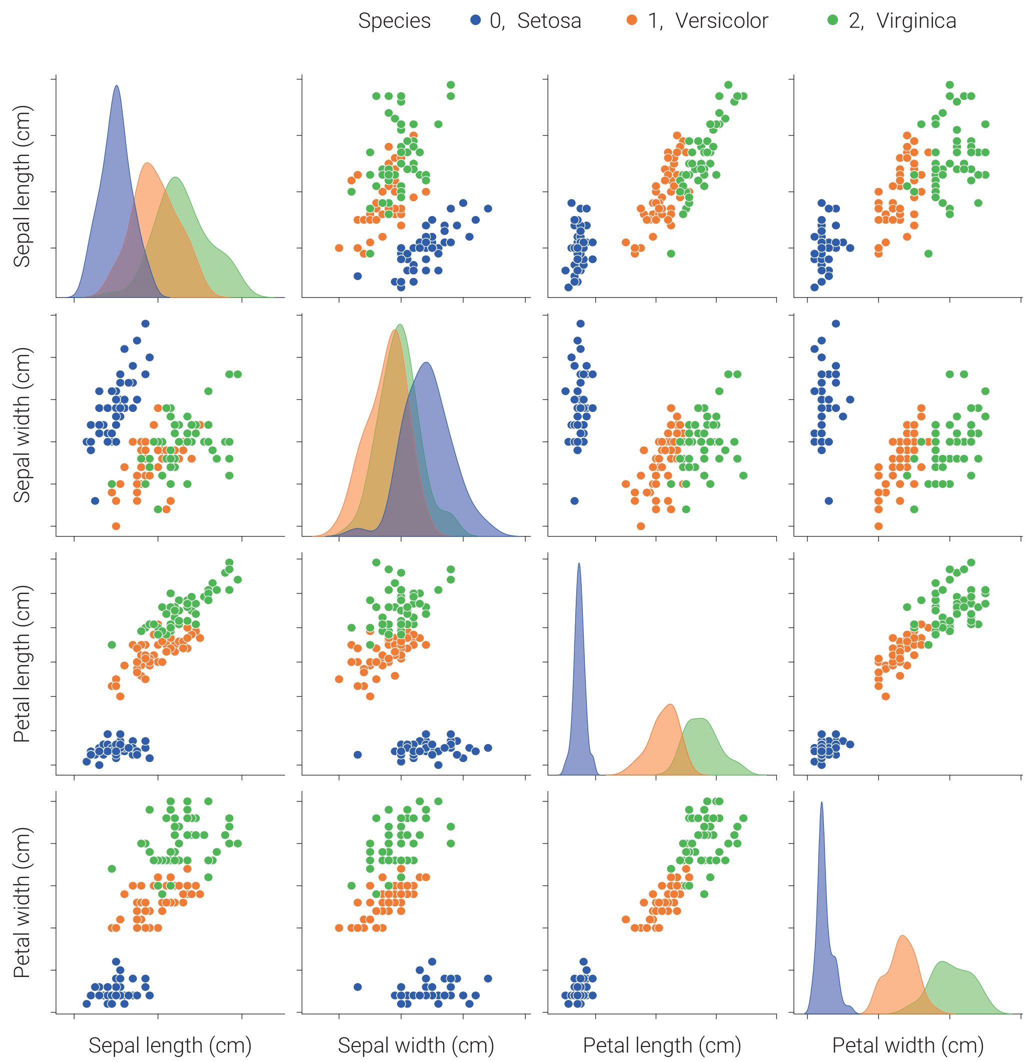


图2.2　鸢尾花原始数据，成对特征分析图

在鸢尾花原始数据中完全随机引入缺失值NaN，将数据存为iris_df_NaN，数据的形式如图2.3所示。图2.4所示为含有缺失值的鸢尾花可视化图像。

	sepal length(cm)	sepal width(cm)	petal length(cm)	petal width(cm)
0	5.1	NaN	NaN	0.2
1	NaN	NaN	1.4	0.2
2	4.7	3.2	1.3	0.2
3	NaN	NaN	NaN	NaN
4	NaN	NaN	1.4	NaN
...	...	...	...	...
145	6.7	NaN	5.2	2.3
146	6.3	2.5	5.0	NaN
147	6.5	3.0	5.2	NaN
148	6.2	NaN	NaN	2.3
149	5.9	3.0	NaN	1.8

图2.3　鸢尾花样本数据，随机引入缺失值

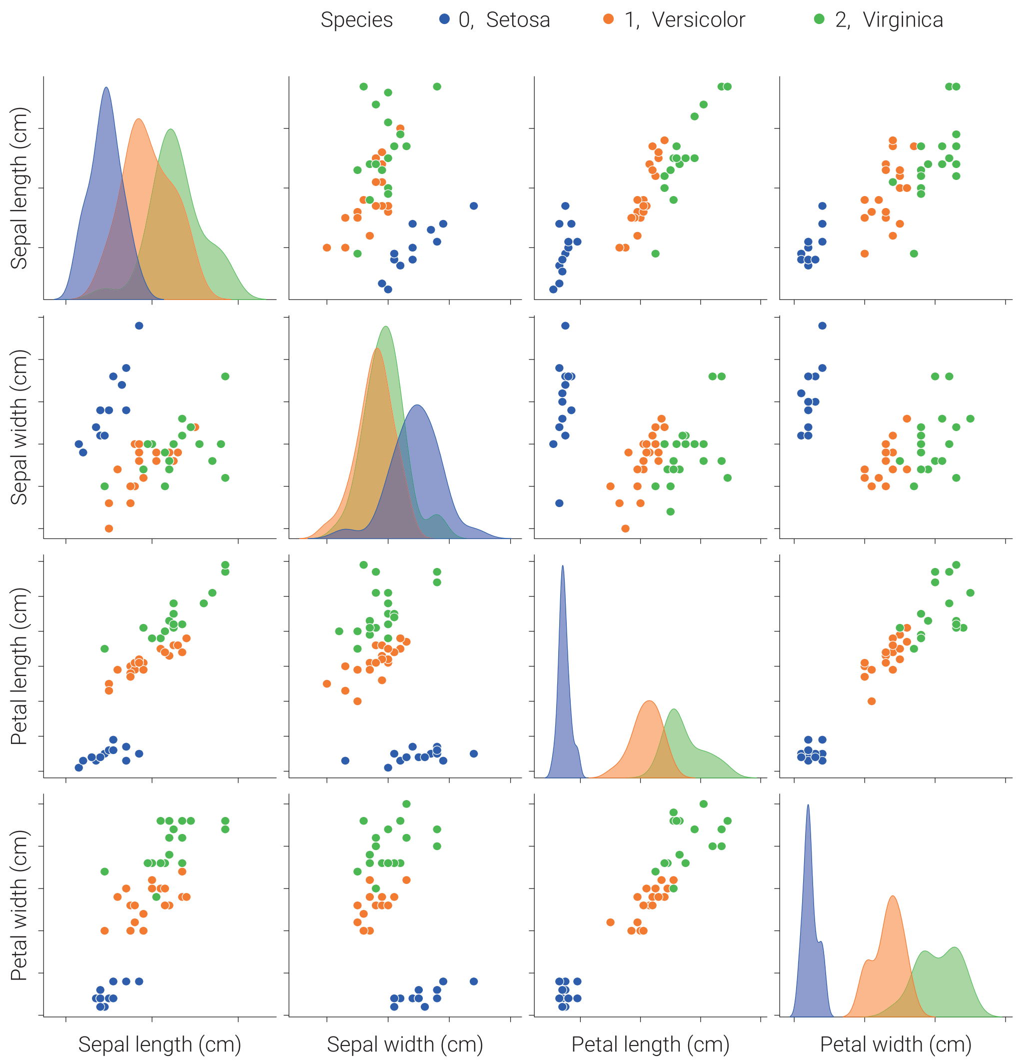


图2.4 鸢尾花数据可视化，引入缺失值

下面讲解代码2.1，这段代码在鸢尾花数据中随机插入了缺失值。

ⓐ使用sklearn.datasets.load_iris() 加载鸢尾花数据集，as_frame=True表示返回数据帧格式，return_X_y=True表示返回特征矩阵X和目标变量y。

ⓑ创建特征矩阵X的副本，存储在iris_df中。

ⓒ在数据帧中添加一列标签，表示鸢尾花的类别。

ⓓ使用Seaborn的pairplot函数绘制成对散点图，hue='species'表示按照鸢尾花类别标签着色，palette="bright"指定颜色主题为明亮色调。

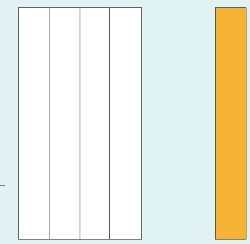
ⓔ生成一个与X_NaN相同形状的随机矩阵，数值满足0~1之间均匀分布。

ⓕ将小于等于0.4的元素设为True，形成一个与X_NaN相同形状的布尔矩阵，用作mask。

ⓖ将X_NaN中对应True位置的元素设置为缺失值NaN。

代码2.1　在鸢尾花数据中随机引入缺失值 | Bk6_Ch02_01.ipynb

```python
# 导入包
from sklearn.datasets import load_iris
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)
X.head()

# 副本
iris_df = X.copy()

# 增加一列标签
iris_df['species'] = y

# 用成对散点图可视化
sns.pairplot(iris_df, hue = 'species', palette = "bright")

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

# 再次用成对散点图可视化有缺失值的数据
iris_df_NaN = X_NaN.copy()
iris_df_NaN['species'] = y
sns.pairplot(iris_df_NaN, hue = 'species', palette = "bright")
```



2.2 可视化缺失值位置

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。

查找缺失值

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。图2.5所示为iris_df_NaN.isna() 结果。

```
     sepal length(cm)  sepal width(cm)  ... petal width(cm)  species
0               False             True  ...           False    False
1                True             True  ...           False    False
2               False            False  ...           False    False
3                True             True  ...            True    False
4                True             True  ...            True    False
...               ...              ...  ...             ...      ...
145             False             True  ...           False    False
146             False            False  ...            True    False
147             False            False  ...            True    False
148             False             True  ...           False    False
149             False            False  ...           False    False
```

图2.5　判断数据是否为缺失值

图2.6所示为采用seaborn.heatmap() 可视化数据缺失值，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到缺失值分布情况。

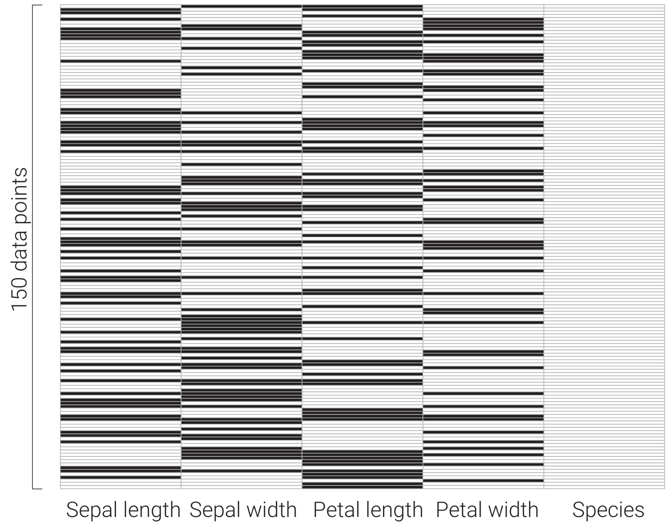


图2.6　缺失值可视化，每条黑带代表缺失值

查找非缺失值

方法notna()正好和isna()相反，iris_df_NaN.notna()判断数据是否为“非缺失值”。如果数据没有缺失，则为True；否则，为False。图2.7所示为iris_df_NaN.notna() 结果。

```
   sepal length(cm)   sepal width(cm)  ... petal width (cm)   species
0              True             False  ...             True      True
1             False             False  ...             True      True
2              True              True  ...             True      True
3             False             False  ...            False      True
4             False             False  ...            False      True
...             ...               ...  ...              ...       ...
145            True             False  ...             True      True
146            True              True  ...            False      True
147            True              True  ...            False      True
148            True             False  ...             True      True
149            True              True  ...             True      True
```

图2.7　判断数据是否为“非缺失值”

可视化数据缺失值如图2.8所示。

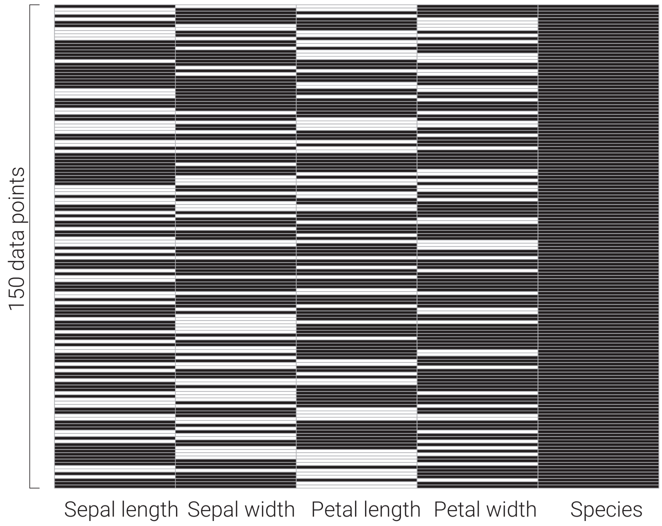


图2.8　缺失值可视化，每条白带代表缺失值

非缺失值变化线图

另外，可以安装missingno (pip install missingno)，并调用missingno.matrix() 绘制缺失值热图，具体如图2.9所示。

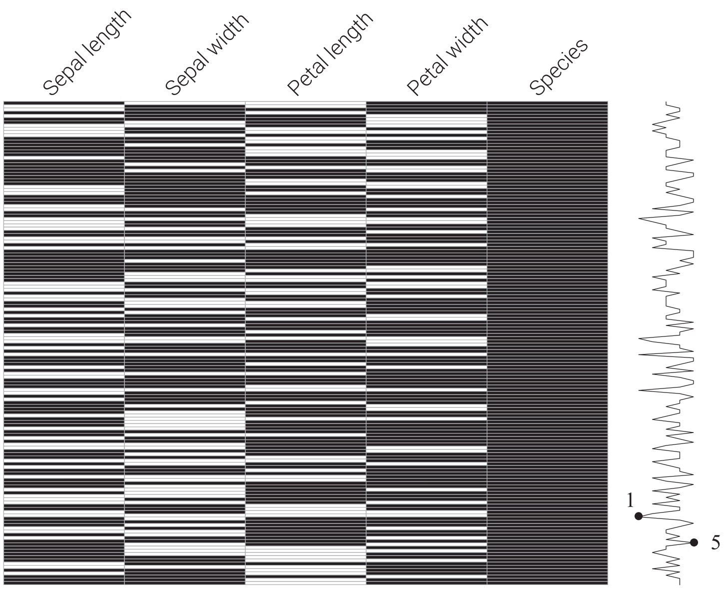


图2.9　missingno.matrix() 绘制缺失值热图，每条白带代表缺失值

这幅图最右侧还展示每行非缺失值数据数量的变化线图，线图最小取值为1，最大取值为5。取值为1时，每行只有一个非缺失值；取值为5时，该行不存在缺失值。观察这幅线图，可以帮助我们解读缺失值分布特征。

总结缺失值信息

对于Pandas数据帧，也可以采用info() 显示数据非缺失值数量和数据类型。图2.10所示为iris_df_NaN.info()结果。df.isnull().sum() * 100 / len(df)则计算每列缺失值的百分比。

也可以采用sklearn.impute.MissingIndicator() 函数将数据转换为相应的二进制矩阵 (True和False，相当于1和0)，以指示数据中缺失值的存在位置。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #  Column                        Non-Null Count  Dtype
---------                         --------------  -----
 0    sepal length (cm)           85 non-null     float64
 1    sepal width  (cm)           94 non-null     float64
 2    petal length (cm)           91 non-null     float64
 3    petal width  (cm)           84 non-null     float64
 4    species                     150 non - nullint32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

图2.10 pd.info() 总结样本数据特征

接着前文代码，代码2.2可视化缺失值。下面讲解其中关键语句。

a 使用isna()方法查找数据帧中缺失值。

b 使用Seaborn的heatmap函数绘制缺失值位置。cmap='gray_r'代表颜色映射为灰度逆序；cbar=False代表隐藏颜色条。

c 使用notna()方法查找数据帧中非缺失值。

d 使用Seaborn的heatmap函数绘制非缺失值位置。

e 使用missingno库的matrix()函数绘制缺失值的可视化矩阵。

f 统计每列中缺失值的总数。

g 计算每列中缺失值的百分比。

代码 2.2　可视化缺失值位置 | Bk6_Ch02_01.ipynb

```python
# 用isna()方法查找缺失值
is_NaN = iris_df_NaN.isna()                  # a
# print(is_NaN)

# 可视化缺失值位置
fig, ax = plt.subplots()
sns.heatmap(is_NaN,                          # b
            ax = ax,
            cmap = 'gray_r',
            cbar = False)

# 用notna()方法查找非缺失值
not_NaN = iris_df_NaN.notna()                # c
# sum_rows = not_NaN.sum(axis = 1)
# print(not_NaN)

# 可视化非缺失值位置
fig, ax = plt.subplots()
sns.heatmap(not_NaN,                         # d
            ax = ax,
            cmap = 'gray_r',
            cbar = False)

# 用missingno.matrix()可视化缺失值
import missingno as msno
```

```
# missingno has to be installed first
# pip install missingno

e  msno.matrix(iris_df_NaN)

# 总结缺失值

f  print("\nCount total NaN at each column:\n" ,
         X_NaN.isnull().sum())

g  print("\nPercentage of NaN at each column:\n" ,
         X_NaN.isnull().sum()/len(X_NaN)*100)
```

2.3 处理缺失值：删除

图2.11总结了处理缺失值常用方法。

- ◀删除：可以删除缺失值所在的行、列，或者**成对删除** (pairwise deletion)。
- ◀**插补** (imputation)：采用插补时，要根据数据特点，采用合理的方法。

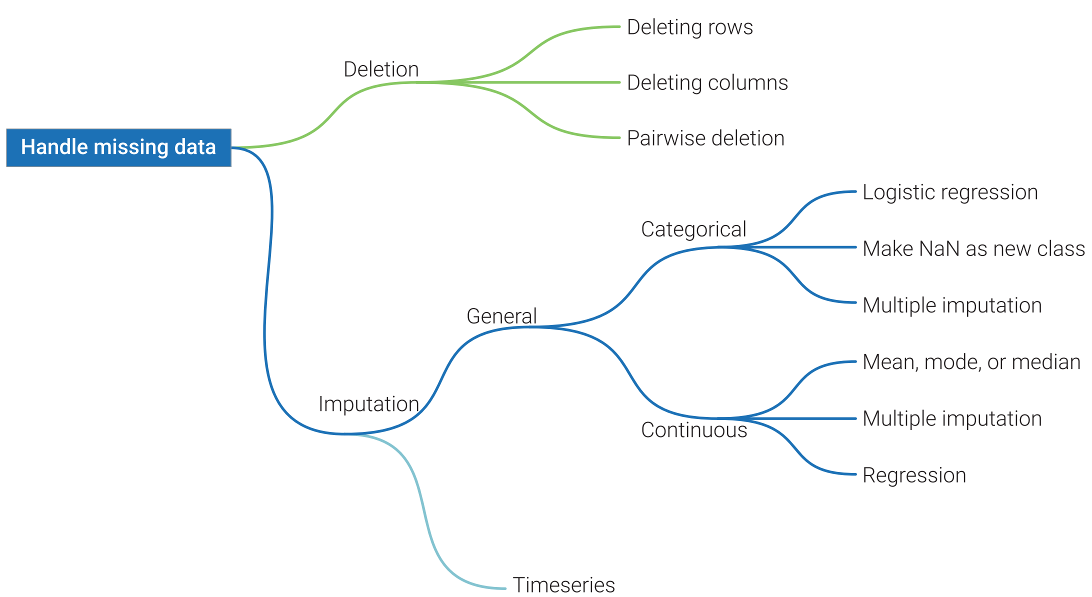


图2.11　处理缺失值的方法分类

对于表格数据，一般情况下，每一行代表一个样本数据，每一列代表一个特征。处理存在缺失值数据集的基本策略是舍弃包含缺失值的整行或整列。但是，这是以丢失可能有价值的数据为代价的。

更好的策略是估算缺失值，即从数据的已知部分推断出缺失值，这种方法统称插补。本章后续主要介绍连续数据的删除和插补方法。本书时间序列一章中将介绍时间序列数据的插补。

删除

下面简单介绍Pandas数据帧dropna() 方法。

对于某一个数据帧df，df.dropna(axis = 0, how = 'any') 中 axis = 0 为按行删除，设置 axis = 1表示按列删除。

参数how = 'any'时，表示某行或列只要有一个缺失值，就删除该行或列，如图2.12所示。

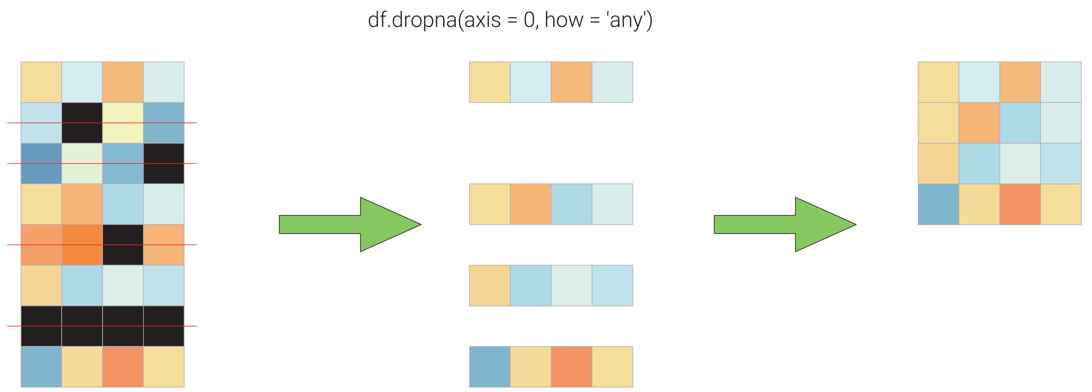


图2.12　Pandas数据帧中删除含有至少一个缺失值所在的行

如图2.13所示，当how = 'all'，表示该行或列全部都为缺失值时，才删除该行或列。dropna()方法默认设置为 axis = 0，how = 'any'。

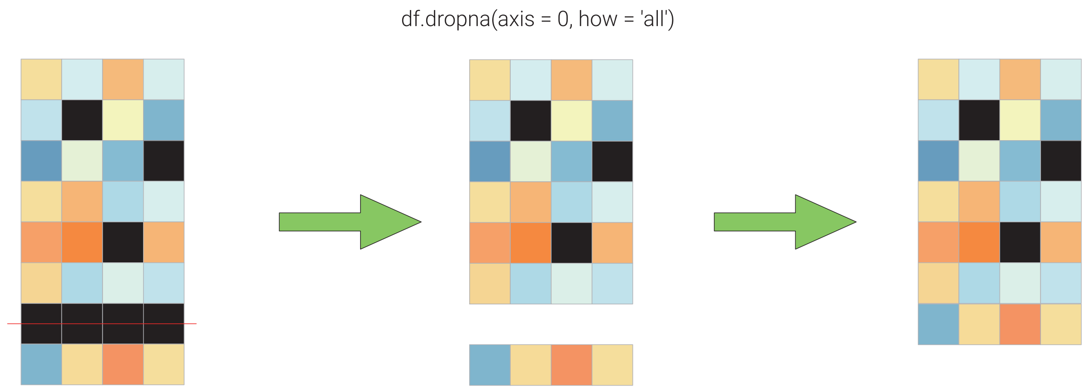


图2.13　Pandas数据帧中删除全为缺失值行

图2.14所示为删除缺失值后的鸢尾花数据，规则为删除含有至少一个缺失值所在的行。对比图2.4，可以发现非缺失数据点明显减小。图2.14中所剩数据便是图2.9中最右侧线图值为5对应的数据点。

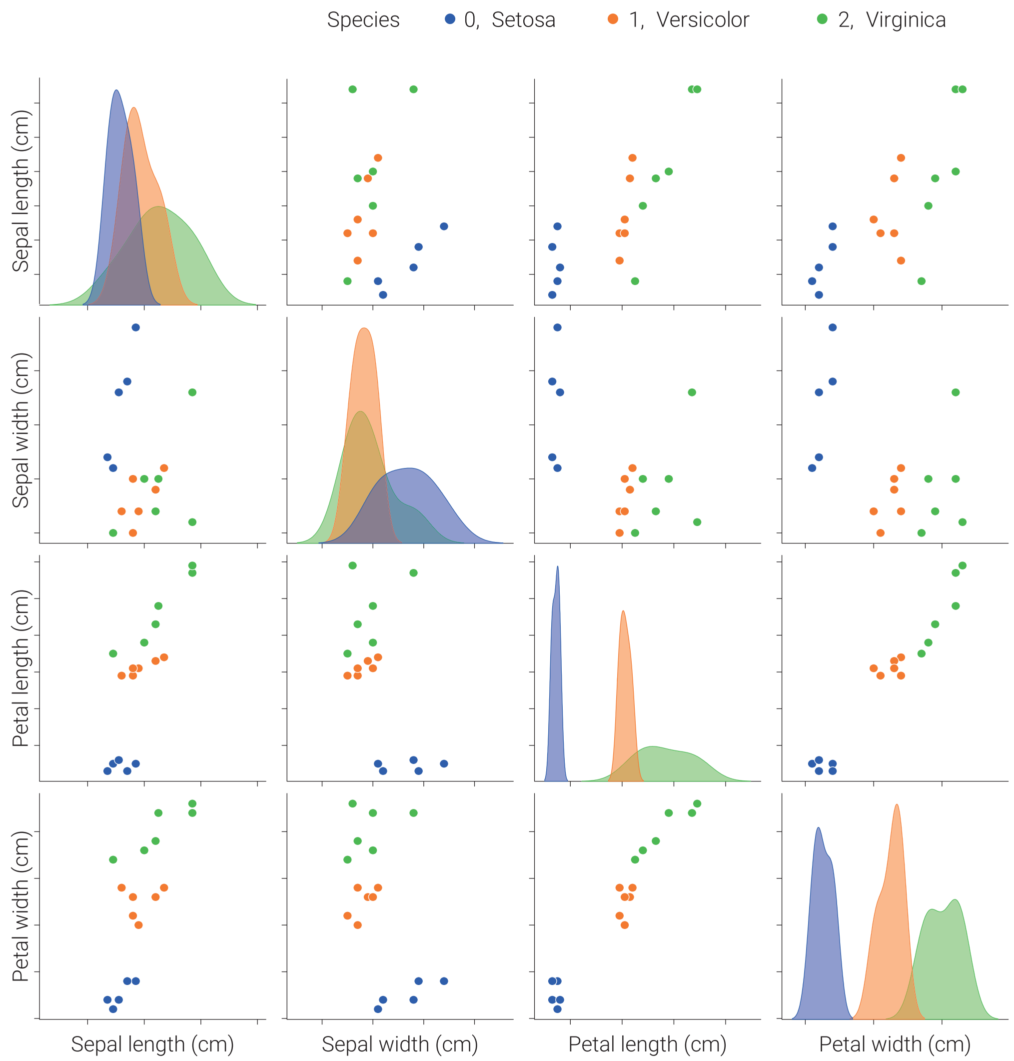


图2.14　鸢尾花数据，删除含有至少一个缺失值所在的行

一般情况下，每列数据代表一个特征，删除整列特征的情况也并不罕见。不管是删除缺失值所在的行或列，都会浪费大量有价值的信息。

成对删除

成对删除是一种特别的删除方式，进行多特征联立时，成对删除只删除掉需要执行运算特征包含的缺失数据；以估算方差协方差矩阵为例，如图2.15所示，计算X_1和X_3的相关性，只需要删除X_1和X_3中缺失值对应的数据点。

数据有道 5校 正文(1-11)(CS6).indd 36　　2024/8/26 19:26:32

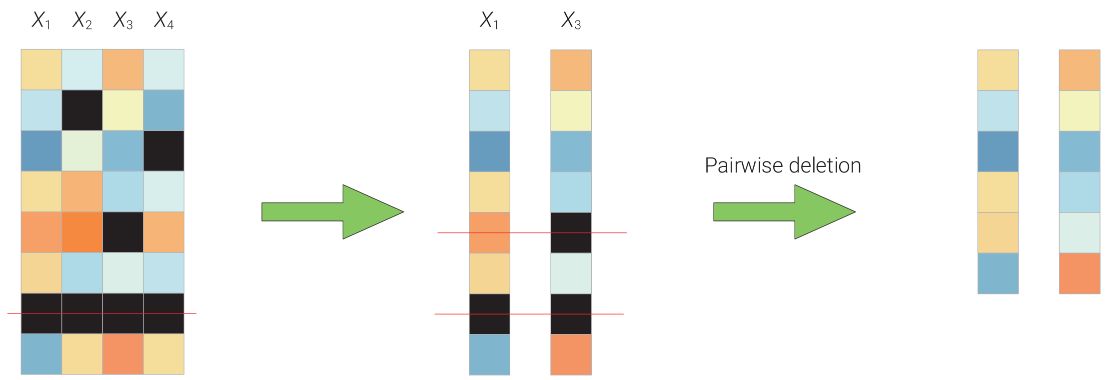


图2.15　成对删除

2.4 单变量插补

相对删除缺失值，更常用的方法是，采用一定的方法补全缺失值，我们称之为**插补** (imputation)。其中，分类数据和连续数据采用的方法也稍有差别。注意，采用插补方法时要格外小心，如果填充方法不合理，会引入数据噪声，并造成数据分析结果不准确。

时间数据采用的插补方法不同于一般数据。Pandas数据帧有基本插补功能，特别是对于时间数据，可以采用**插值** (interpolation)、向前填充、向后填充。这部分内容，我们将在本书插值和时间序列部分详细介绍。

单变量插补：统计插补

本节专门介绍单变量插补。单变量插补也称统计插补，仅使用第j个特征维度中的非缺失值插补该特征维度中的缺失值。本节采用的函数是sklearn.impute.SimpleImputer()。

SimpleImputer() 可以使用缺失值所在的行/列中的统计数据平均值 ('mean')、中位数 ('median') 或者众数 ('most_frequent') 来填充，也可以使用指定的常数 ('constant')。

如果某个特征是连续数据，可以利用其他所有非缺失值平均值或中位数来填充该缺失值。

如果某个特征是分类数据，则可以利用该特征非缺失值的众数，即出现频率最高的数值来补齐缺失值。

图2.16所示为采用中位数插补鸢尾花缺失值。观察图2.16，可以发现插补得到的数据形成“十字”图案。

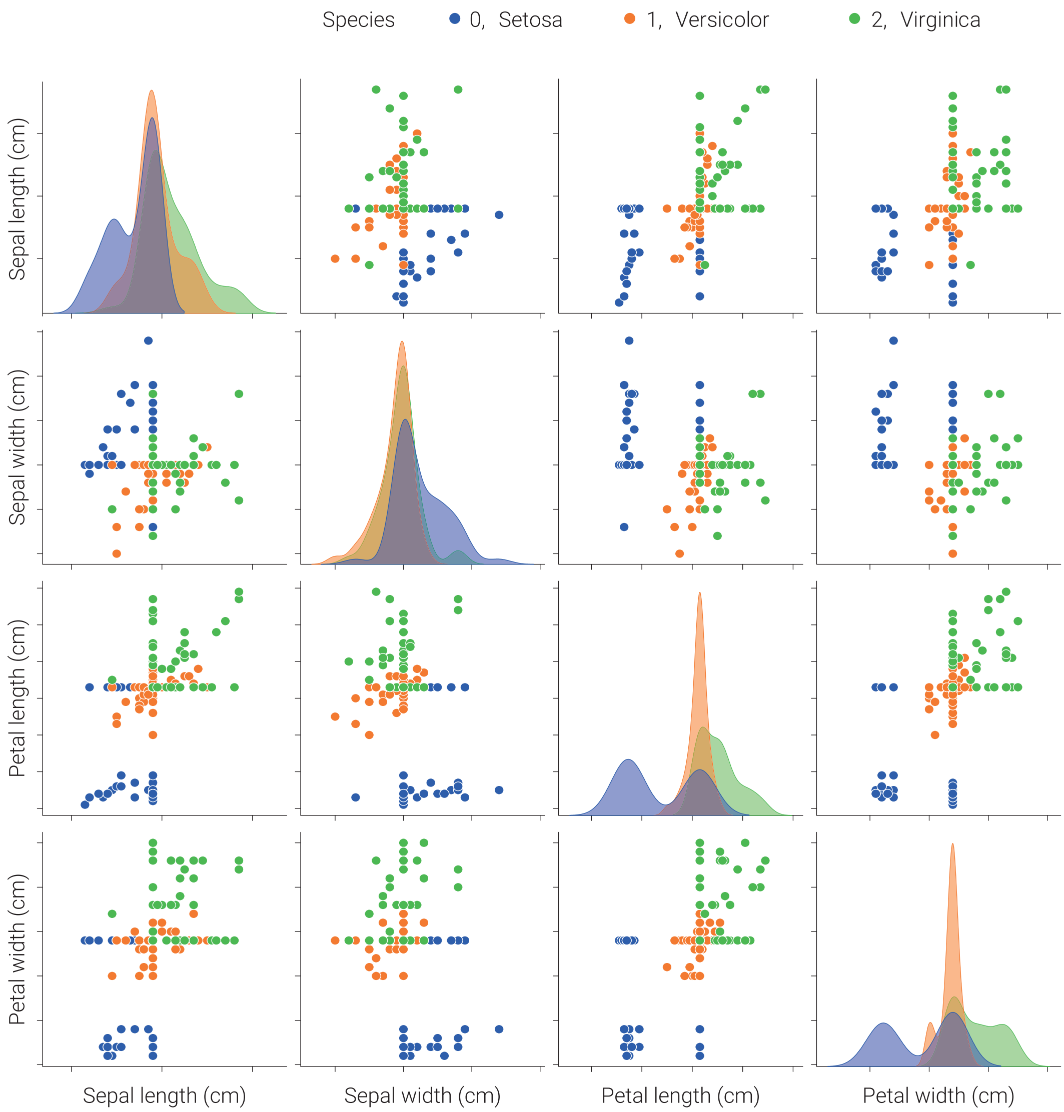


图2.16　鸢尾花数据，采用中位数插补缺失值

在前文代码基础上，代码2.3完成了单变量插补。

ⓐ调用sklearn.impute.SimpleImputer()完成单变量插补。

如果参数strategy选择“mean”，则用每列的均值替换缺失值。仅可用于数值数据。

如果选择“median”，则用每列的中位数替换缺失值。仅可用于数值数据。

如果选择“most_frequent”，则用每列的最频繁值替换缺失值。可用于字符串或数值数据。如果有多个最频繁值，则仅返回最小的一个。

如果选择“constant”，则用指定的fill_value替换缺失值。可用于字符串或数值数据。

ⓑ用fit_transform()完成单变量插补。

代码2.3　单变量插补 | Bk6_Ch02_01.ipynb

```
from sklearn.impute import SimpleImputer

si = SimpleImputer(strategy = 'median')
# impute training data
X_NaN_median = si.fit_transform(X_NaN)

iris_df_NaN_median  = pd.DataFrame(X_NaN_median,
                                   columns = X_NaN.columns,
                                   index = X_NaN.index)

iris_df_NaN_median['species'] = y
sns.pairplot(iris_df_NaN_median,
             hue = 'species', palette = "bright")
```

2.5 *k*近邻插补

本节介绍*k*近邻插补。***k*近邻算法** (*k*-nearest neighbors algorithm，*k*-NN) 是最基本监督学习方法之一，*k*-NN中的*k*指的是“近邻”的数量。简单来说，*k*-NN的思路就是“近朱者赤，近墨者黑”。

本节介绍*k*近邻插补的函数为sklearn.impute.KNNImputer()。利用KNNImputer插补缺失值时，先给定距离缺失值数据最近的*k*个样本，将这*k*个值等权重平均或加权平均来插补缺失值。图2.17所示为采用*k*近邻插补鸢尾花数据结果。

《机器学习》将专门介绍*k*近邻算法这种监督学习方法。

在前文代码基础上，代码2.4采用了最近邻插补。下面讲解其中关键语句。

代码先是从sklearn.impute库中导入KNNImputer类，该类用于使用KNN算法进行缺失值的插补。

a 创建KNNImputer对象knni，并指定邻居数量为5。

b 使用fit_transform方法对包含缺失值的数据X_NaN进行插补，得到插补后的数据X_NaN_kNN。

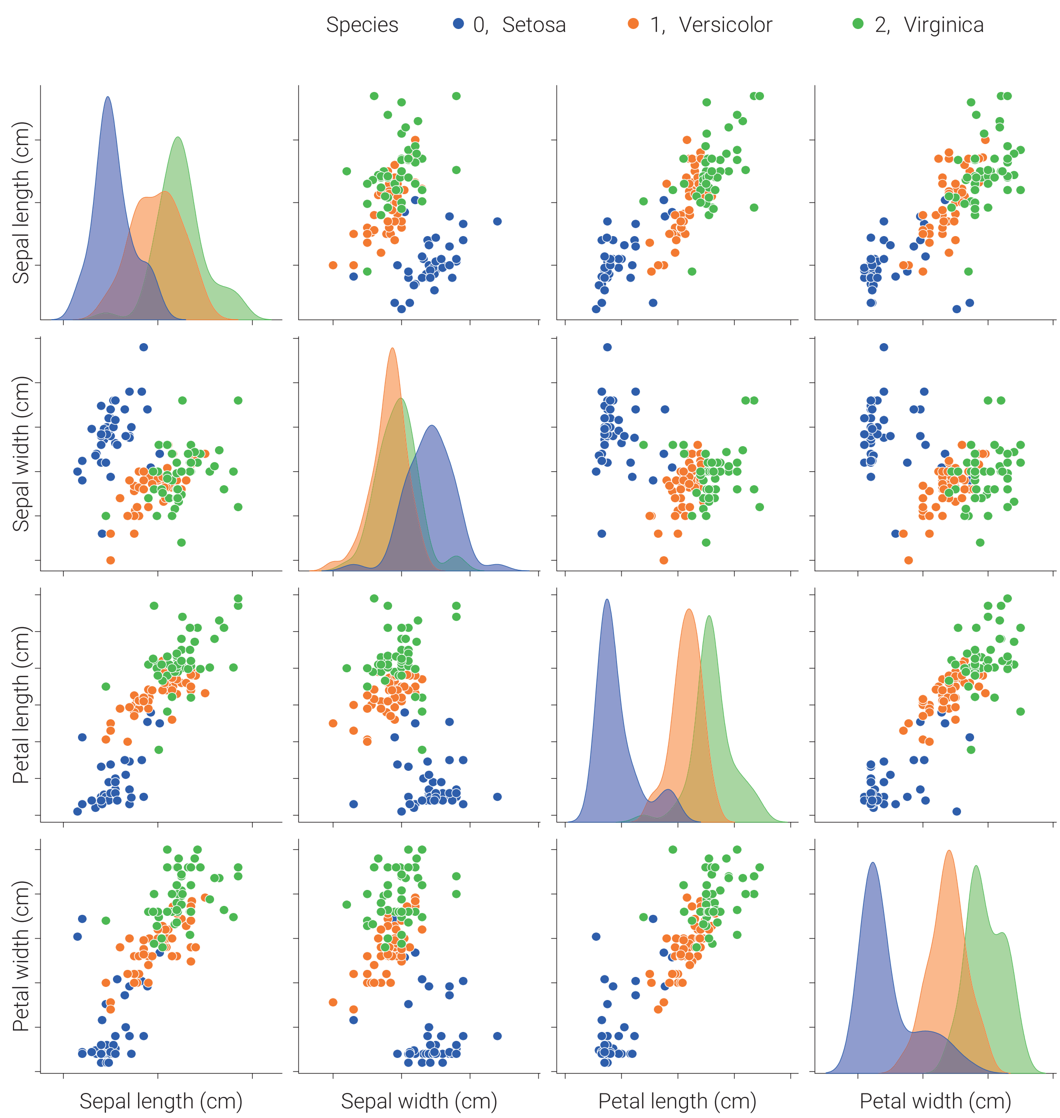


图2.17　鸢尾花数据，最近邻插补

代码2.4　k近邻插补 | Bk6_Ch02_01.ipynb

```
from sklearn.impute import KNNImputer

knni = KNNImputer(n_neighbors = 5)
X_NaN_kNN = knni.fit_transform(X_NaN)

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

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

2.6 多变量插补

多变量插补，利用其他特征数据来填充某个特征内的缺失值。多变量插补将缺失值建模为其他特征的函数，用该函数估算合理的数值，以填充缺失值。整个过程可以用迭代循环方式进行。

单变量插补一般仅考虑单一特征进行插补，而多变量插补考虑不同特征数据的联系。

图2.18所示为采用sklearn.impute.IterativeImputer() 函数完成多变量插补，补齐鸢尾花数据中缺失值。

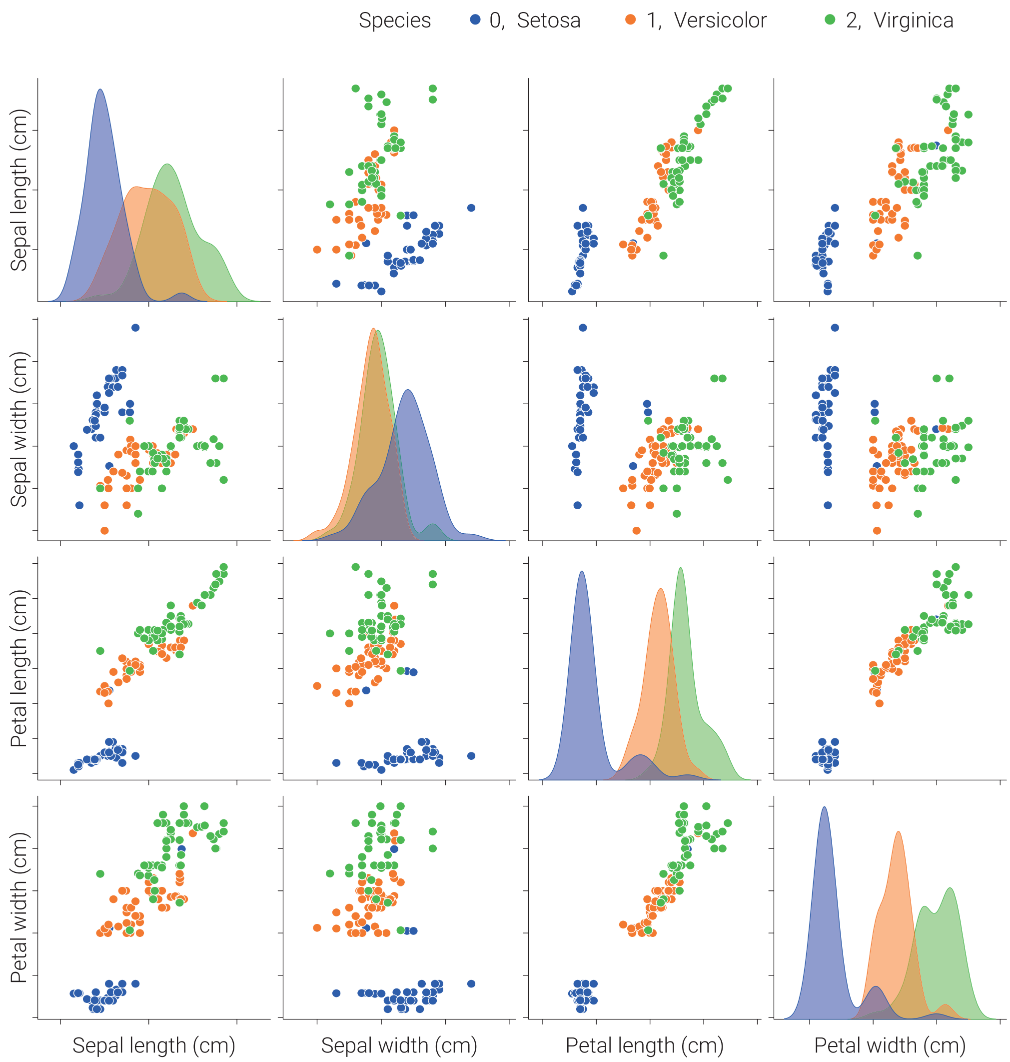


图2.18 鸢尾花数据，多变量插补

在前文代码基础上，代码2.5完成了多变量插补。

ⓐ创建IterativeImputer对象rf_imp，使用RandomForestRegressor作为估算器，并设置最大迭代次数为20。

ⓑ使用fit_transform方法对包含缺失值的数据X_NaN进行插补，得到插补后的数据X_NaN_RF。

代码2.5 多变量插补 | Bk6_Ch02_01.ipynb

```python
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestRegressor

rf_imp = IterativeImputer(estimator =
                          RandomForestRegressor(random_state = 0),
                          max_iter = 20)
X_NaN_RF = rf_imp.fit_transform(X_NaN)

iris_df_NaN_RF = pd.DataFrame(X_NaN_RF,
                              columns = X_NaN.columns,
                              index = X_NaN.index)
iris_df_NaN_RF['species'] = y
sns.pairplot(iris_df_NaN_RF, hue = 'species',
             palette = "bright")
```

总结一下，缺失值指的是在数据集中某些观测或特征的数值缺失或未记录。缺失值在机器学习中可能导致各种各样问题，因为模型需要完整的数据来进行有效的训练和预测。而失值可能导致模型性能下降，因为模型可能无法准确学习缺失值对应的模式或关系。此外，缺失值还可能引入偏见，导致对特定子集的预测不准确。

处理缺失值的方法有几种。一种是删除包含缺失值的行或列，但这可能会损失大量信息。另一种是填充缺失值，可以用均值、中位数或其他统计量代替缺失值。还有一些先进的技术，如插值方法或使用机器学习模型来预测缺失值。选择哪种方法取决于数据的性质和缺失值的模式。

有关数据帧处理缺失值，请大家参考：

◀ https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html

sklearn.impute.IterativeImputer() 函数非常灵活，可以和各种估算器联合使用，比如决策树回归、贝叶斯岭回归等等。感兴趣的读者可以参考：

◀ https://scikit-learn.org/stable/modules/impute.html
◀ https://scikit-learn.org/stable/auto_examples/impute/plot_iterative_imputer_variants_comparison

数据有道 5校 正文(1-11)(CS6).indd 42 2024/8/26 19:26:36

03 Detecting Outliers 离群值

利用统计方法和机器学习算法筛出离群值

数学领域，提出问题比解决问题，更珍贵。

In mathematics the art of proposing a question must be held of higher value than solving it.

—— 格奥尔格 • 康托尔 (Georg Cantor) | 德国数学家 | 1845 —1918年

- `numpy.percentile()` 计算百分位
- `pandas.DataFrame()` 构造 Pandas 数据帧
- `seaborn.boxplot()` 绘制箱型图
- `seaborn.histplot()` 绘制直方图
- `seaborn.kdeplot()` 绘制概率密度估计曲线
- `seaborn.pairplot()` 绘制成对分析图
- `seaborn.rugplot()` 绘制 rug 图像
- `seaborn.scatterplot()` 绘制散点图
- `sklearn.covariance.EllipticEnvelope()` 协方差椭圆法检测离群值
- `sklearn.ensemble.IsolationForest()` 孤立森林检测离群值
- `sklearn.svm.OneClassSVM()` 支持向量机检测离群值
- `stats.probplot()` 绘制 QQ 图