2 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 (Galilei Galileo) | 意大利物理学家、数学家及哲学家 | 1564 ~ 1642

- `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，是便用 `True` 占位，否便用 `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 缺失值小传

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。处理缺失值是数据预处理中重要一环。

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。

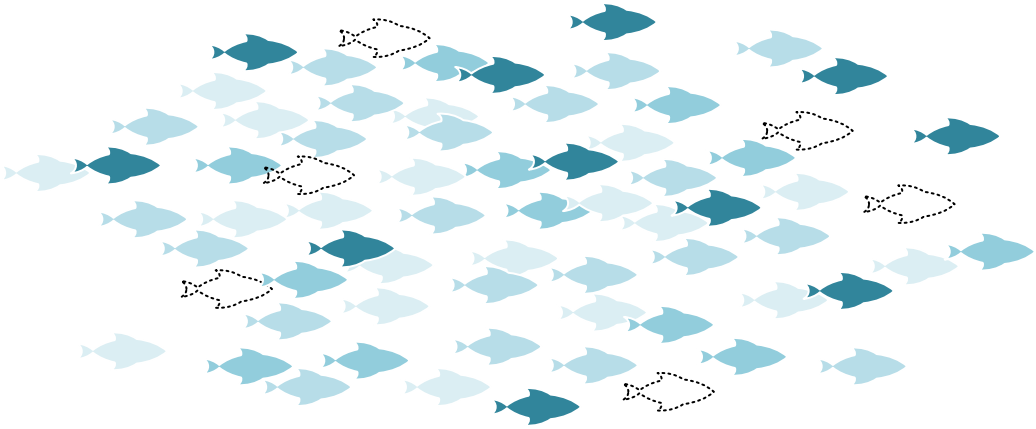

图 1. 缺失值

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，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本身值存在一定关系，比如高收入群体不希望透露它们的收入。

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 1 第二列 Numpy 函数遇到缺失值 NaN，会直接报错。而表 1 第三列函数，计算时忽略 NaN。

表 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 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。

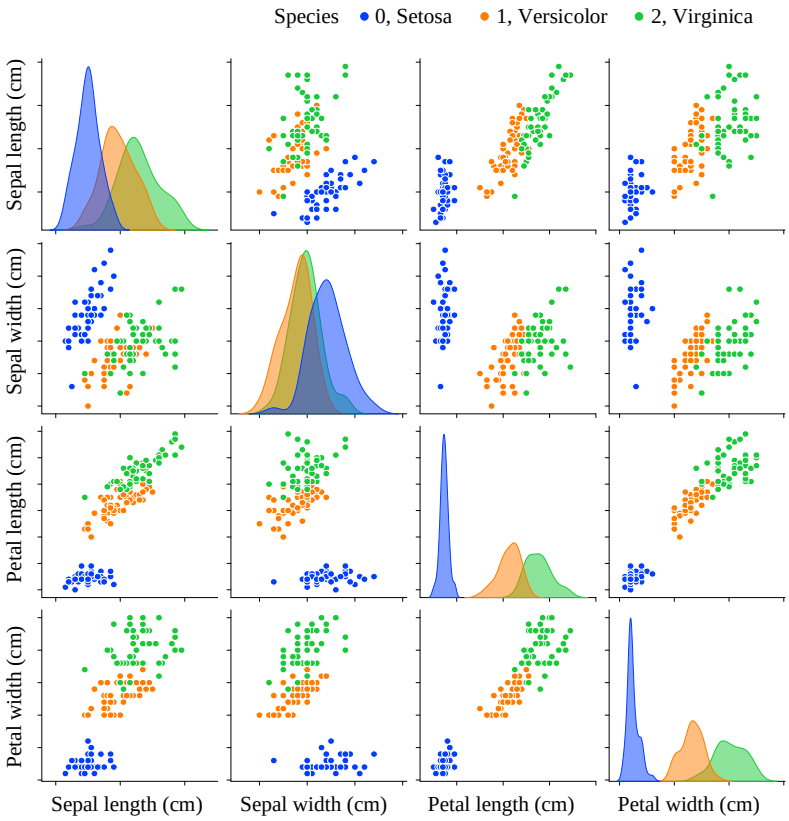

图 2. 鸢尾花原始数据，成对特征分析图

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 3 所示。图 4 所示为含有缺失值得鸢尾花可视化图像。

	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 3. 鸢尾花样本数据，随机引入缺失值

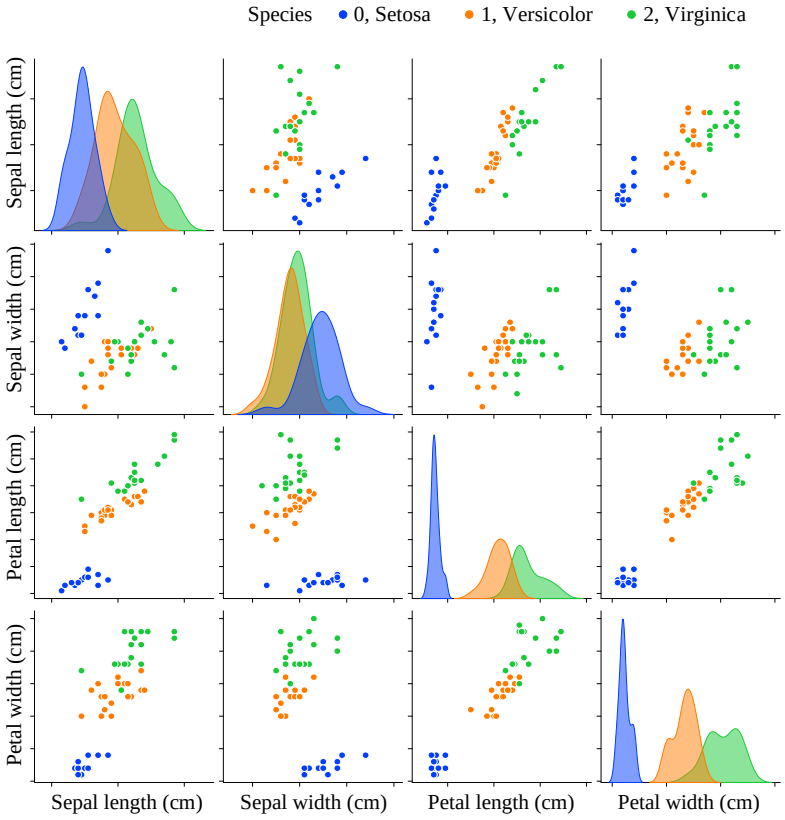

图 4. 鸢尾花数据可视化，引入缺失值

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 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 5. 判断数据是否为缺失值

图 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得到缺失值分布情况。

图 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。图 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 7. 判断数据是否为“非缺失值”

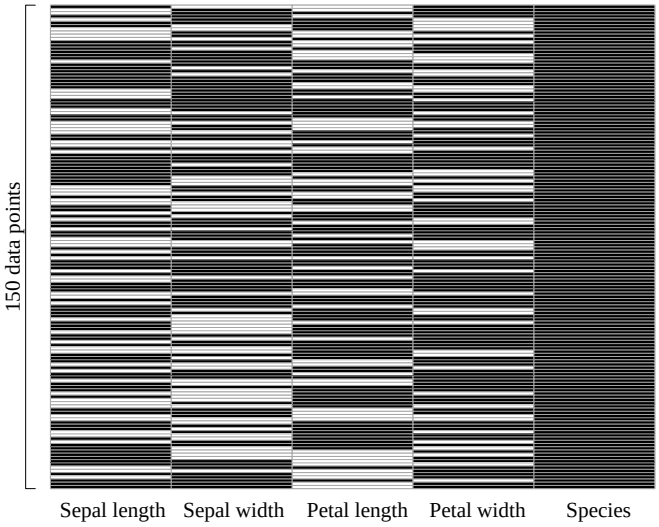

图 8. 缺失值可视化，每条白带代表缺失值

非缺失值变化线图

另外，可以安装 missingno，并调用 missingno.matrix() 绘制缺失值热图，具体如图 9 所示。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。

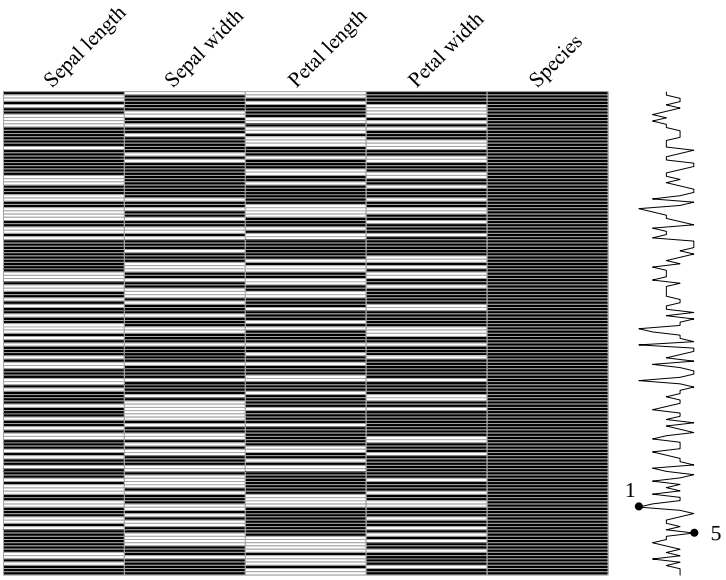

图 9. missingno.matrix()绘制缺失值热图，每条白带代表缺失值

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 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。

```
<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-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

图 10. pd.info() 总结样本数据特征

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。

2.3 处理缺失值：删除、插补

如图 11 所示，处理缺失值有两个主要办法：

- 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，要根据数据特点，采用合理的方法。

对于表格数据，一般情况，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(imputation)。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。

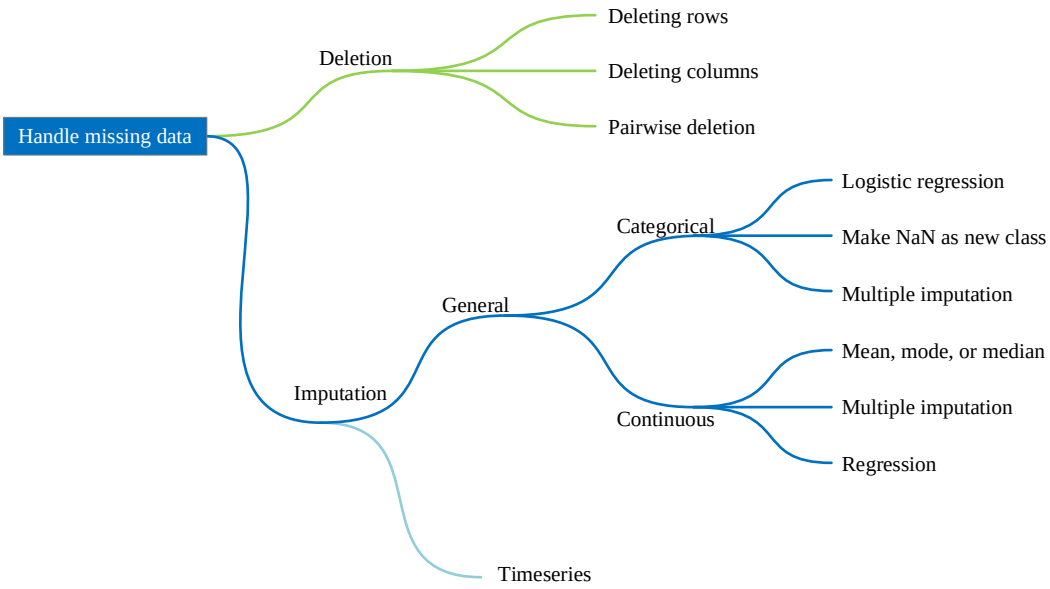

图 11. 处理缺失值的方法分类

2.4 删除：最基本方法

本节简单介绍 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。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 12 所示。

如图 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'。

```
df.dropna(axis=0, how='any')
```

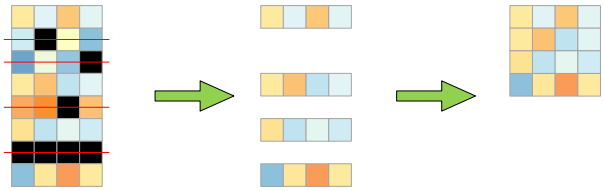

图 12. Pandas 数据帧中删除含有至少一个缺失值所在的行

```
df.dropna(axis=0, how='all')
```

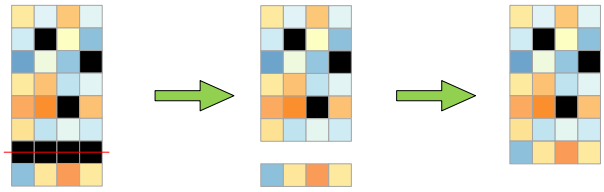

图 13. Pandas 数据帧中删除全为缺失值行

图 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 4，可以发现非缺失数据点明显减小。图 14 中所剩数据便是图 9 中最右侧线图值为 5 对应的数据点。

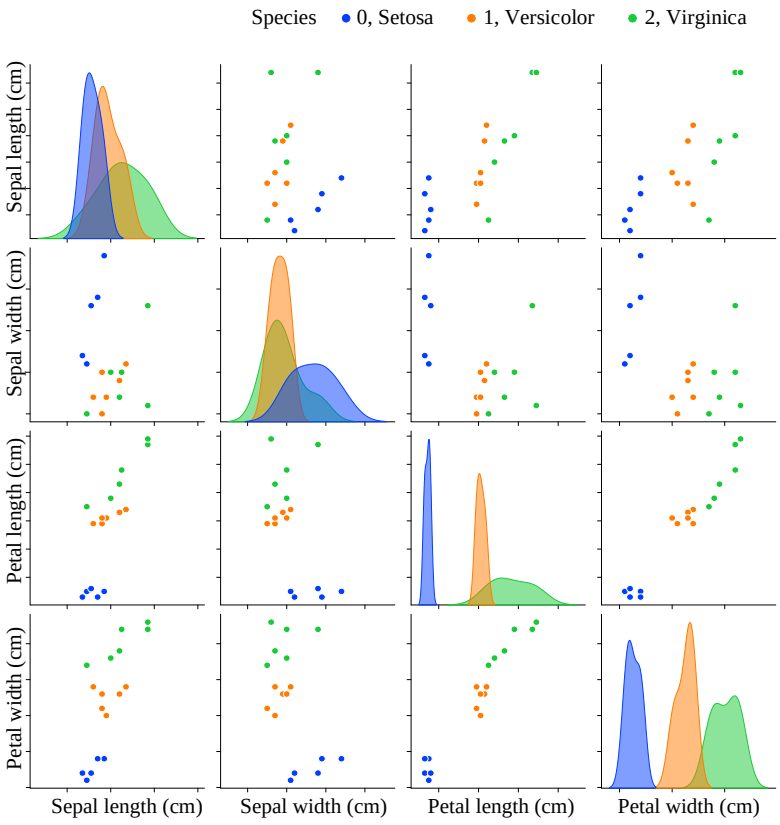

图 14. 鸢尾花数据，删除含有至少一个缺失值所在的行

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 (pairwise deletion) 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 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。

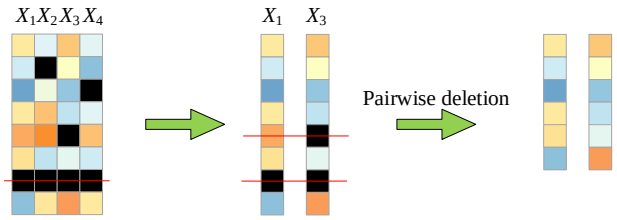

图 15. 成对删除

2.5 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)。如图 11 所示，分类数据和连续数据采用的方法也稍有差别。注意，选取采用插补方法要格外小心，如果填充方法不合理，会引入数据噪音，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'。

如果某个特征是是连续数据，可以根据在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 16 所示为采用中位数插补鸢尾花缺失值。观察图 16，可以发现插补得到的数据形成“十字”图案。

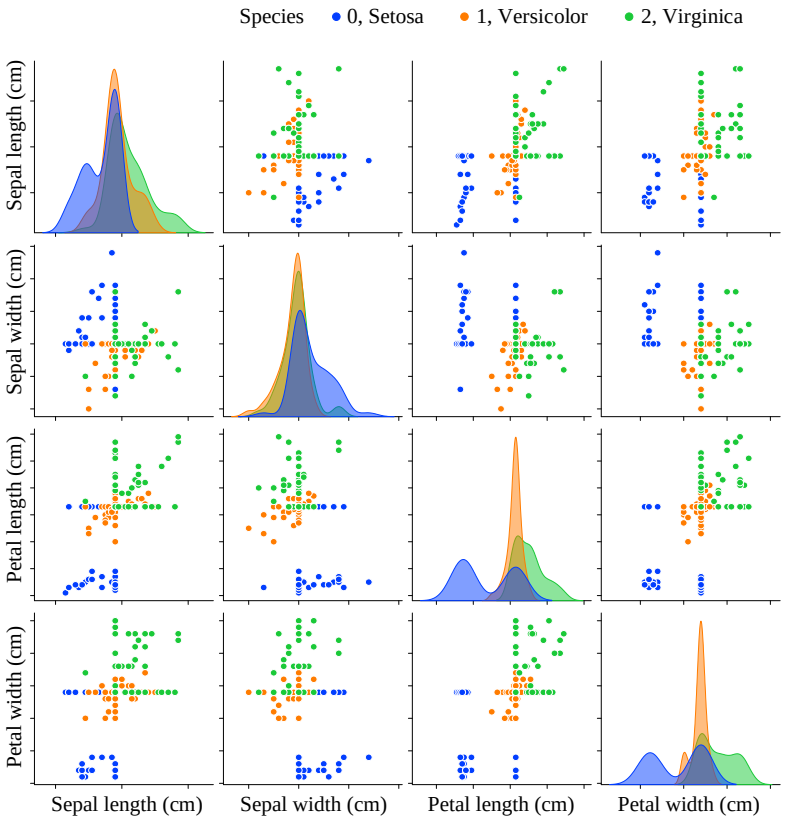

图 16. 鸢尾花数据，采用中位数插补缺失值

2.6 *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。*k*-NN 思路很简单——“近朱者赤，近墨者黑”。更准确地说，小范围投票，**少数服从多数** (majority rule)。

《机器学习》第 2 章专门介绍 *k* 近邻算法这种监督学习方法。

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 17 所示为采用 *k* 近邻插补鸢尾花数据结果。

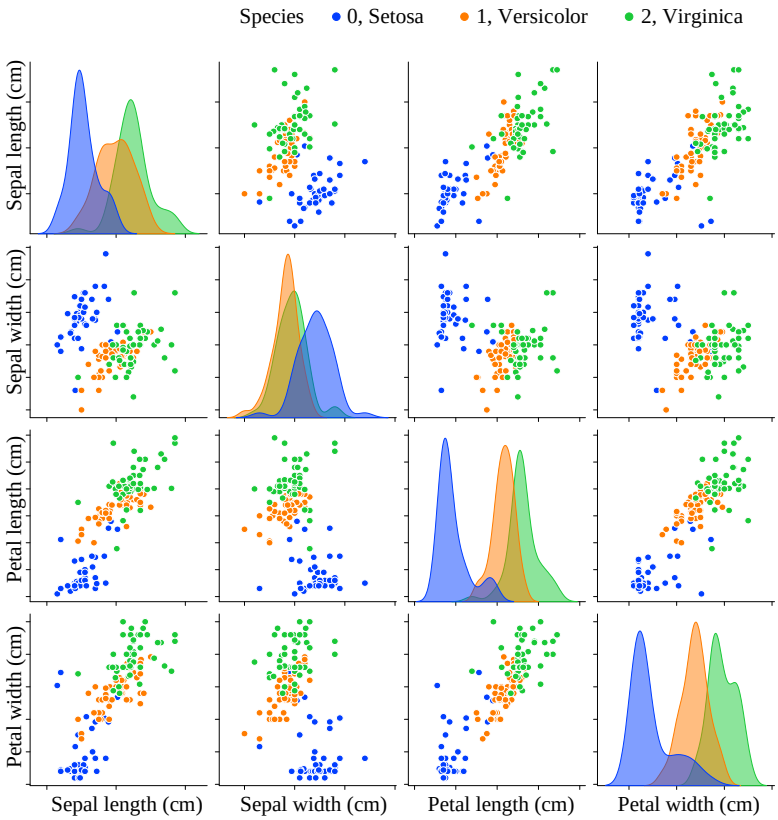

图 17. 鸢尾花数据，最近邻插补

2.7 多变量插补

多变量插补，利用其它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 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。

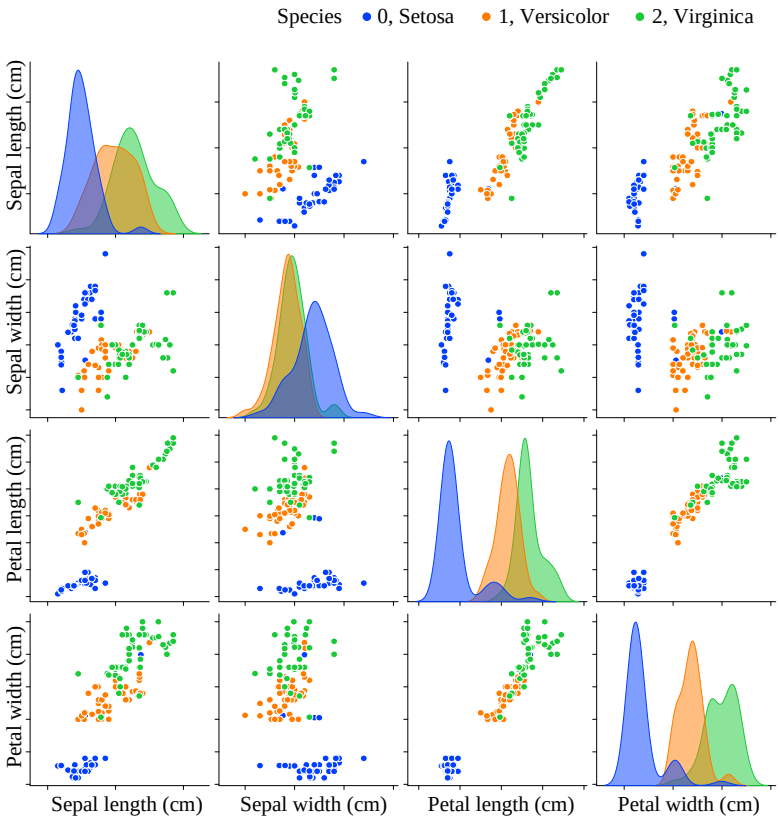

图 18. 鸢尾花数据，多变量插补

Bk6_Ch02_01.py 绘制本章大部分图像。

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如下链接：

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

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