第5章 探索性数据分析

之前章节已经介绍了如何设计数据方案获取数据，以及如何使用 Python 导入数据。本章将介绍得到数据后，如何对数据进行初步分析，即探索性数据分析。

探索性数据分析(Exploratory data analysis，EDA)是我们获得数据以后的第一步工作，目的是初步了解数据集，验证一些简单假设，为形成后续的假设、构建模型提供基础和依据。本章将介绍基本的 EDA 流程与常用方法，以及一些数据清洗与处理手段。其间，还会对涉及的一些统计学概念和信号处理知识做必要的补充。

归纳来说，EDA 的流程包括数据检查、数据预处理，以及数据初步分析，如图 5-0-1 所示。其中，数据检查又涉及了解数据的意义及规模、了解特征的数据类型及意义，以及初步排除数据泄露等；数据预处理则包括缺失处理、异常处理和冗余处理等；数据初步分析则主要是对现有数据做最基本的描述性统计。

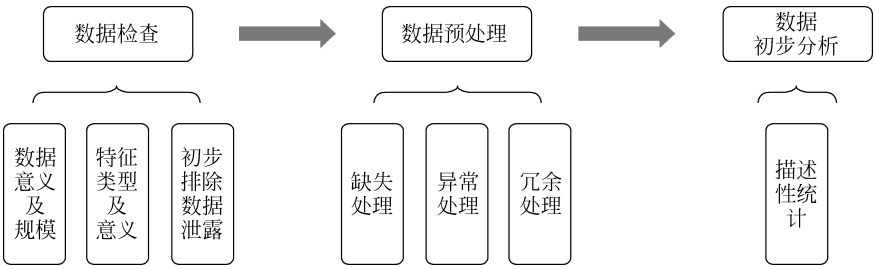


图 5-0-1　EDA 流程

5.1　数据检查

总体来说，数据检查主要包括了解数据的意义及规模、了解特征的数据类型及意义，初步排除数据泄露等。

5.1.1　数据的意义及规模

数据是什么数据？是结构化的还是非结构化的？如果是结构化的数据，即数据已经被组织成了二维表格的形式，其中的行、列分别对应什么实际意义？如果是非结构化的数据，能否转换为结构化的数据以方便后续处理？数据包含多少样本？每个样本由多少特征来描述？对于结构化的数据，数据包含多少行、多少列？如果样本被划分成不同类别，各类别的样本容量又分别是多少？

我们用一些具体的例子来进一步说明。

例 5-1-1　Titanic 数据集的数据意义和规模检查。

由 Titanic.csv 文件给出的 Titanic 数据集，根据数据提供方的描述，记录的是当年 Titanic 船难时，船上部分乘客的信息，已被组织成结构化数据，支持用 Excel 软件查看，也可以用 Python 导入读取。

```
import pandas as pd
my_data = pd.read_csv("Titanic.csv")
my_data
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

用 Python 命令 pandas.read_csv 导入 Titanic.csv 文件之后，通过屏幕输出读入的数据 my_data，可以看到这个数据集包含了 891 行×12 列，这就是数据规模。

继续查看数据，可以看出数据每一行代表一位乘客，即总共有 891 位乘客（样本容量 891）；每一列代表乘客的一个特征或属性，每位乘客由 12 个特征来描述，包括“乘客编号”“是否生还”“舱位等级”“姓名”“性别”“年龄”“同行平辈人数”“同行父母或子女人数”“票号”“船费”“舱名”和“登船港口”。

现在，请读者思考这样一个问题：例 5-1-1 中这 12 个特征可以被不加区别地用同样的方法（例如求算术平均）来分析吗？可能有读者注意到了，这 12 个特征不能被笼统地不加区别对待。为什么呢？这就涉及与特征实际意义紧密联系的特征的数据类型。

5.1.2 特征的数据类型及意义

数据的每一列（特征）在文件中的存储数据类型是什么？其对应的实际意义是什么？由此实际意义决定的实际数据类型又是什么？

进行数据分析时，我们常常将特征分为以下几种数据类型。

1. 数值型数据

数值型数据指具有数量上的意义，支持比较大小，同时还支持加减乘除、求算术平均等基本数学运算的那些数据。数值型数据在计算机中常常体现为整数或浮点数的存储形式。

2. 排序型数据

排序型数据也具有数量上的定义，所以支持比较大小，但不一定满足基本的数学运算。例如我们通常的排名数据，可以定义第一名最好，排名数字越大对应排名越差，所以排名是可以相互比较大小的。但是排名数据并不支持加法运算，例如第一名＋第二名并不等于第三名，第一名与第三名的算术平均也并不一定就等于第二名。所以通常的排名，尽管可能以整数体现，但并不能当成线性域定义的数值来对待。

排序型数据在计算机中可能出现整数和字符型的存储形式。

3. 类别型数据

类别型数据则没有数量上的对应性，因此没有大小的意义，仅仅作为一个标记符号，表示出一个类别与其他类别的不同，或者是一个个体与其他个体的不同。

类别型数据在计算机中最常见的是字符型的存储形式，但也不乏以整数存储的情况。当出现以整数存储的类别数据时，要特别注意，尽管对其做数学运算并不违背语法规则，但是没有实际意义的。

4. 逻辑型/布尔型数据

类别型数据中，如果只存在两种非此即彼的选择，则一般称为逻辑型数据或布尔型数据。

逻辑型数据在计算机中可以体现为布尔存储形式(True 或 False)，也可能体现为整数 0、1 的存储形式。逻辑型数据支持的运算与数学运算不同，是我们所说的逻辑运算。

例 5-1-2 Titanic 数据集中的特征检查。

结合各个特征的实际意义，我们会发现这 12 个特征呈现几种不同的数据类型。例如：

特征“年龄”“同行平辈人数”“同行父母或子女人数”“船费”是有数量上的意义的，不同的值之间能比较大小，如“人数”4 大于“人数”2，“年龄”52 大于“年龄”34，也支持基本的数学运算，例如加减法、求算术平均等，因此，这些都是通常所说的数值型数据。

特征“性别”“姓名”“票号”“登船港口”是以字符(串)型存储的类别型数据，不具备数量上的意义，不支持数学运算。

特征“是否生还”是什么类型？数据文件中各个样本在该特征上的取值是数字“0”或“1”，所以它是数值型数据吗？

判断一个数据是否是数值型的，关键是考察其是否具备数量的意义(能比较大小)，以及是否支持基本的数学运算。特征“是否生还”有两个取值，即 1(是)、0(否)，“1”比“0”大吗？这两种取值做加减法会有意义吗？显然这两个问题的答案都是否定的，所以它不是数值型数据。对于这种只有两个备选，且非此即彼的数据类型，通常称为逻辑型或者布尔型。

再来看特征“舱位等级”，它是数值型吗？如果“舱位等级”在定义时已经认定“1”代表最好的舱位，数字越大舱位等级越低，那么这个特征有数量上的定义，可以比较大小。但是，一个一等舱加上一个二等舱，并不等于一个三等舱，所以它也不是一个数值型数据。我们可以认为这是一种排序类型，是依据某个度量排序得到的，具备一定的量化意义。如果这里的数字 1～4 仅仅代表了不同的类别，不具备量的意义，即数字大或小都不代表等级高，这时只能认为它属于类别型数据。

最后来看特征“乘客编号”，这个特征也极具迷惑性，好像是数值型或排序型。但是这个编号明显不支持数学运算，所以不是数值型。如果其大小没有特殊的含义，既不是某种度量的排序，也不是排序型。该特征主要用来区别不同的乘客，所以我们依然可以

把它当成类别型来对待。

之前已经介绍过，pandas 的 read_csv 函数将 csv 文件中的数据导入一个 DataFrame 结构，DataFrame 封装的 info 函数能提供一些对数据的基本分析，例如数据的规模、每个特征有多少非空值，以及特征存储时采用的数据类型。要注意的是，这里的存储数据类型并不一定是特征的实际意义上的数据类型，但能给我们提供一定的参考。

```
1 print(my_data.info())   #这里可以看到，dataframe的info方法能返回对数据的一些总结
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
None
```

可见，对于特征，首先可依据“是否具备量化意义（即是否可比较大小），是否支持数学运算”的准则，区分出数值型数据和非数值型数据。对于数值型数据，后续可以用算术平均、标准差等量化统计量来分析；而对于非数值型数据，则主要依据它们来进行分组与筛选，整个判断流程可以参照图 5-1-1。

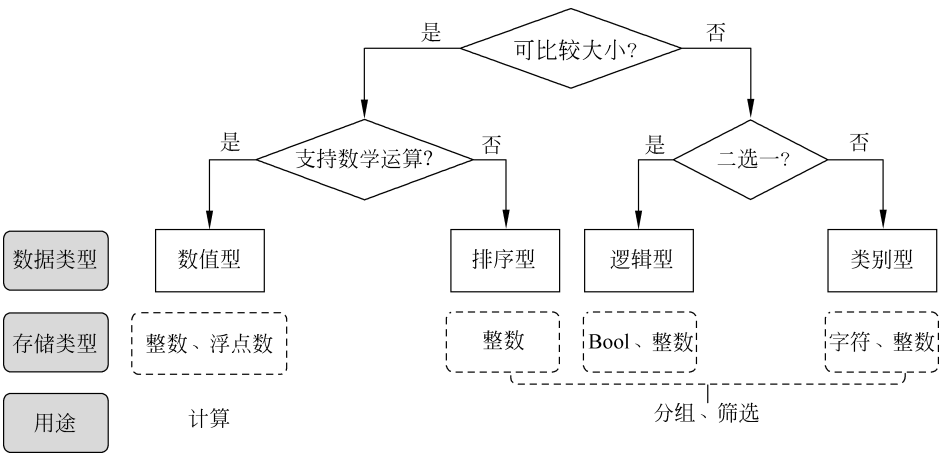


图 5-1-1　判断数据类型的流程

例如，我们对上述加载的 Titanic 数据，根据“舱位等级”进行分组，对“船费”进行算术平均统计，就可以获得每种“舱位等级”对应的平均“船费”了。

例 5-1-3 Titanic 数据集中不同舱位等级的船费统计。

```
import pandas as pd
my_data = pd.read_csv("Titanic.csv")
print('Table 1. Mean Fare of Group')
x = my_data.groupby(['PClass']).mean()
print(x['Fare'])
Table 1. Mean Fare of Group
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Fare, dtype: float64
```

5.1.3 初步排除数据泄露

除了清楚上述的特征类型之外，关于数据的特征，我们还有需要特别关注的方面：这些特征真的是我们所关心的某种性质的体现，还是仅仅人为制造？再面临新数据时，新数据也会具备这些特征并保持定义的一致性吗？

所以，EDA 中还有一项工作也不能忽略，即对数据泄露的排查，特别是后续如果要建立机器学习模型。这里的“数据泄露”并不是指常规意义上对于数据隐私的泄露，而是特指用来作为模型输入的特征中包含了泄露输出目标的信息，而这种特征在真实应用中却又是无法获得的。我们仍然通过具体的例子来进行说明。

鸢尾花数据集(iris.csv)总共包含 150 行(150 个样本)、5 列，即每个样本由 5 个属性描述。5 个属性中，前 4 个都是花萼或花瓣的尺寸信息，最后一个则是该样本所属的鸢尾花种属信息。假设我们要构建一个模型来实现鸢尾花种属的自动分类，也就是说希望模型能输出数据集的最后一列信息，那用什么作为模型的输入呢？我们注意到整个数据集中，前 50 个、中间 50 个和最后 50 个刚好分别属于三个不同的种属。所以我们决定构造一个样本序号特征，这个特征取值为 1～150，完全反映样本在数据集中的顺序位置。接下来我们可以保证，仅用这个构造出来的序号特征作为模型输入，就能完美区分这 150 个鸢尾花的种属，例如采用图 5-1-2 所示的决策树(或条件分支结构)。

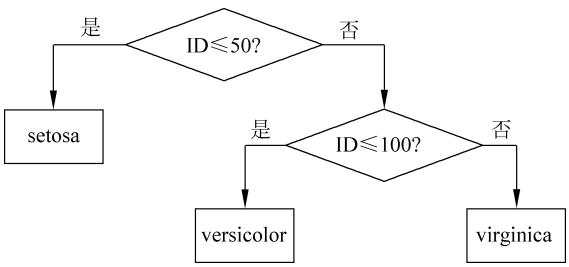


图 5-1-2 一个“完美”鸢尾花分类模型

请思考一下，图 5-1-2 这个“完美”模型是否有什么问题呢？更进一步，当我们要将这个模型应用于新采集来的数据，你预测模型判断鸢尾花种属正确的概率有多大？

这个模型中，我们构造的特征“样本序号”纯粹是我们看到了数据集中三个种属排序规律后人为制造出来的特征，与鸢尾花本身并无关系，但却如作弊一般泄露了用于建模的训练集中的分类目标信息，所以在这 150 个数据上显示出性能“完美”。但是，这种“完美”根本经不起新数据的考验，当面临新数据时，新数据的“样本序号”无疑会大于 150，在图 5-1-2 所示的模型下，会被全部判断为 virginica，毫无意义。这就是一种数据泄露。

现实情况下，人为规定的样本 ID 号常常是数据泄露的重要来源。例如以下这个例子：

在数据挖掘领域的国际知名 KDD 杯 2008 年度竞赛中，主办方提供了 1700 多位病患的医学图像数据作为训练集，目标是实现乳腺癌的诊断。有参赛者发现，数据集中的“病人 ID”存在着数据泄露，并给出了图 5-1-3，其中横轴代表病人 ID 号，纵轴代表用图像数据训练出来的支持向量机（SVM）模型的患乳腺癌风险打分，灰色点代表良性个体，黑色点代表恶性个体。图 5-1-3 显示仅仅用“病人 ID”就能达到比 SVM 模型更好的良性与恶性的区分度。但是很显然，用病人 ID 作为乳腺癌的诊断依据是很荒谬的，因为面临一个没有 ID 的新数据时，要如何诊断呢？为什么病人 ID 会泄露病患是否患乳腺癌的信息呢？一个可能的解释是，这些数据来源于不同的医疗机构，不同的机构对病人的编号方式不一样，这里连位数都不一样，所以导致了图 5-1-3 中 ID 完全分离的三个簇。而医疗机构本身可能又会泄露病患的疾病程度，例如一个保健型的医疗机构和一个专注于乳腺癌治疗的专科医疗机构，其中病人真正患乳腺癌的可能性是大不一样的。

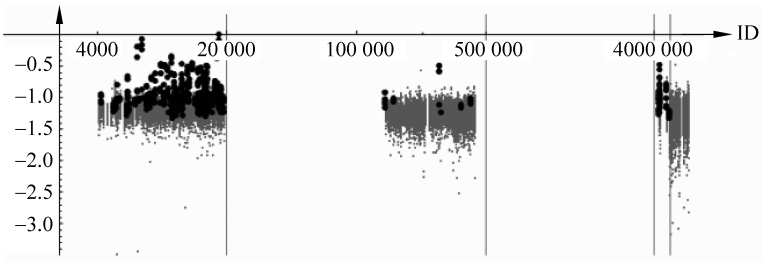


图 5-1-3　2008 年 KDD 杯某参赛队伍对比 ID 特征与 SVM 分类性能的图

（图片来源 https://doi.org/10.1145/2020408.2020496）

也许有读者会有疑问：数据泄露虽然是有作弊嫌疑，但只要结果好，不是也可以接受吗？请不要忘记，数据泄露带来的严重问题是，模型只在训练集（即建模数据）上表现得性能好，新数据因为目标是未知的，泄露特征将无法包含目标信息，因此再也没有获得好性能的理由了。其实这个现象（在训练集上表现很好，面临新数据却非常糟糕）本身就可以作为是否有数据泄露的一个重要判断依据。

除了上述例子中列举的形式，数据泄露还可能以更隐蔽的形式出现。从特征本身来看，建议主要做以下两点甄别：①特征真的是数据本身数量或性质的体现，还是某种人为

的定义；②新的数据是自动具备这个特征，还是需要人工定义或由目标派生。当两个问题的答案都是前者时，才能比较放心地使用特征。

5.2 数据预处理

数据检查过后就可以进行数据预处理。基本的数据预处理主要包括缺失处理、异常处理和冗余处理。

5.2.1 缺失处理

仍以 Titanic 数据集为例。

```
import pandas as pd
my_data = pd.read_csv("Titanic.csv")
my_data.head(15)
```

我们发现其中的“年龄”(Age)和“舱号”(Cabin)所在列，都出现了 NaN 符号(图 5-2-1)。NaN 是英文 Not a Number 的缩写，表示未定义数据。

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S

图 5-2-1 Python 导入 Titanic 数据时的 NaN 现象

如果对照看原始的电子表格文件，我们会发现凡是出现 NaN 的地方，其文件中对应的位置是空白的(图 5-2-2)，也就是说，数据缺失了。Python 读入表格文件时，正是用 NaN 来标记其中的数据缺失。

事实上，现实中获取的数据有缺失值的现象是非常常见的。对于随机缺失的情况，也就是数据的缺失并不针对特定的数据，是无目的或非故意的，其处理可以有两种方式：丢弃和填充。

如果样本容量本身足够大，其中有信息缺失的只占少数，此时我们可以选择直接丢弃有缺失的数据。DataFrame.dropna 函数可以用来丢弃含有 NaN 的数据。

Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, I	male	22	1	0	A/5 2117	7.25		S
2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkiner	female	26	0	0	STON/02.	7.925		S
4	1	1	Futrelle,	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mi	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mi	male		0	0	330877	8.4583		Q
7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson,	male	2	3	1	349909	21.075		S
9	1	3	Johnson,	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, I	female	14	1	0	237736	30.0708		C

图 5-2-2　Titanic 原始表格中的缺失现象

例 5-2-1(a)　对 Titanic 数据集的缺失丢弃处理——行丢弃。

```
my_fil_data1 = my_data.dropna(axis = 0)
my_fil_data1.head(7)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S

例 5-2-1(b)　对 Titanic 数据集的缺失丢弃处理——列丢弃。

```
my_fil_data2 = my_data.dropna(axis = 1)
my_fil_data2.head(7)
```

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	0	0	373450	8.0500
5	6	0	3	Moran, Mr. James	male	0	0	330877	8.4583
6	7	0	1	McCarthy, Mr. Timothy J	male	0	0	17463	51.8625

如例 5-2-1 的代码所示，当我们对 dropna 函数指定参数 axis＝0 时，返回的数据则丢弃了所有包含 NaN 的行，可以看到，处理后数据中的乘客编号(Passengerid)不再连续，有些数据行(乘客)被直接丢掉了；当指定参数 axis＝1 时，返回的数据则丢掉了所有包含 NaN 的列，可以看到，处理后的数据已经没有年龄、舱号等存在缺失值的特征。

如前所述，当样本容量本身足够大，而其中有信息缺失的只占少数时，直接丢弃有缺失的数据，对样本的影响不会太大，我们尚可接受。但是，如果样本量本身就不是很大呢？由于某项特征的数据缺失，就丢弃整行或整列数据的做法无疑会让我们进一步损失更多有效信息，所以，很多情况下，我们会采用修补的做法，也就是对缺失项进行填充。

DataFrame.fillna 方法可以对缺失项进行填充。我们可以为填充内容专门定义填充字典。

例 5-2-2(a) 对 Titanic 数据集的缺失填充处理 —— 字典填充。

```
mean_Age = int(my_data[['Age']].mean()[0])
my_dict = {'Age':mean_Age,'Cabin':'haha'}
my_fil_data3 = my_data.fillna(my_dict)
my_fil_data3.head(7)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	haha	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	haha	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	haha	S
5	6	0	3	Moran, Mr. James	male	29.0	0	0	330877	8.4583	haha	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S

在例 5-2-2(a)的代码中,我们专门定义了一个名为 my_dict 的字典,以此来规定针对不同的列(特征),用不同的值填充。比如,对“年龄”,我们用所有样本年龄的平均值来填充;对于“舱号”,用指定的字符“haha”来填充。运行代码后可以看到,填充后的表格数据,在原来年龄缺失的地方(表格第 6 行)有了数值 29.0,这其实是所有样本的平均年龄,而舱号所在的列也出现了很多新的字符串“haha”,这正是我们在替代字典 my_dict 中所指定的。

如果我们并不想用固定的值来填充缺失项,fillna 还提供用邻近值来填充的选择。

例 5-2-2(b) 对 Titanic 数据集的缺失填充处理——邻近值填充。

```
my_fil_data4 = my_data.fillna(method = 'ffill')
my_fil_data4.head(7)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	C85	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	C123	S
5	6	0	3	Moran, Mr. James	male	35.0	0	0	330877	8.4583	C123	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S

当我们设置 fillna 中参数 method=ffill,程序会用缺失值之前最邻近的有效值来填充,如例 5-2-2(b)所示,可以看到填充后的表格其第 6 行,原本缺失的年龄用之前一行的年龄 35 来填充了;“舱号”所在列,除了第 1 行仍为 NaN 外,其他原本有缺失的行都有了与其前数据相同的取值。

而当设置 fillna 中参数 method=bfill 时,则是用缺失值之后最邻近的有效值来填充,读者不妨自己试一下。

注意:需要清醒地认识到,对于随机缺失,无论我们采取哪种缺失值填充,都不能还原本来因数据缺失而丢失的信息。但是,通过对缺失值的填充,我们可以将原本不完整数据行中的其他信息利用起来,从而避免了有效信息的进一步损失。所以缺失值填充是

一种止损手段，而不是加分手段。那么在选择具体的填充内容或方法时，只要不严重偏离样本原来的性质就可以了。

此外，现在的模型其实也大多支持直接的缺失值输入，亦即不丢弃、不填充，直接将NaN当作一种特征取值。其实对于非随机性缺失，直接保留NaN常常是一种很有效的做法，因为此时缺失这个现象本身就有一定内涵，是包含了一定信息的。

5.2.2 异常处理

除了内容缺失外，数据还有可能混杂了噪声、干扰，甚至有可能根本就是错误的。对于这样的坏数据，我们也必须在建模之前就将其甄别出来并给予相应的处理。对于信号的去噪、去干扰技术，其本身就构成了一个内涵丰富的信号处理领域，本书中不做介绍。本节利用基本的统计学方法，对一些异常处理常用手段做相关介绍，主要包括：结合特征实际意义和有效值范围的判断、基于正态分布 z-score 的判断，以及无正态分布约束的四分位距(Interquartile Range，IQR)判断。

关于错误数据的甄别，首先要基于数据的实际意义，利用常识或专业领域知识进行判断。例如年龄不可能为负数，也一般不会有超过100的数，船费也不可能为负数，等等。我们通常会为每个特征基于其实际意义，定义一个有效的取值范围，超出范围就会判定数据是错误数据，并予以丢弃或修正，修正的方式，可以参考之前缺失值填充的做法。

然后，我们会基于统计学特征做进一步判断，一般又分为两种情况：数据服从正态分布的情况，以及数据不服从正态分布的情况。

如果先验知识告诉我们数据服从正态分布，我们还可以结合统计学中的 z-score 进行异常值判别。z-score 的定义如下：

$$z=\frac{x-\bar{x}}{s} \tag{5-2-1}$$

式中，$\bar{x}$ 为统计均值；$s=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})^2}$ 为样本集的标准差。可以看出，z-score 其实质是以标准差为单位衡量的个体偏离统计均值的程度。结合正态分布，可以快速查出样本落入偏离均值3倍标准差以外的概率是非常小的(约为0.0026)，因此我们一般认为 z-score 绝对值大于3时，数据就可怀疑为异常。z-score 是个无量纲的数，因而不受数据本身的单位和取值范围的影响。

如果不能肯定数据是否服从正态分布，则可以用四分位距作为异常值的判断标准。四分位距是借助四分位数来定义的。假设样本容量为 N，将所有样本按所考察特征(数值型的)的取值从小到大排列，排序在其中 $N\times\frac{1}{4}$，$N\times\frac{2}{4}$，$N\times\frac{3}{4}$ 的特征值就分别称为第1、第2、第3个四分位数。所谓四分位距，是指第3个四分位数(记为 q_3)和第1个四分位数(记为 q_1)之间的差距，即 IQR=q_3-q_1。特征在某个个体上的取值小于该特征的 $q_1-1.5\times$IQR 或大于 $q_3+1.5\times$IQR 的样本，被认为是异常值；而小于 $q_1-3\times$IQR 或大于 $q_3+3\times$

IQR 的样本，则被认为是极端异常值，异常值或极端异常值又常被称为离群值(outlier)。

对于按上述方法判断为异常的值，如果确定是出错的数据，则丢弃和替换都是可以的；但是如果不能肯定是错误，则最好增加样本容量，既能确保样本尽量真实反映总体，同时又能减小异常值对于最终分析结果的影响。

5.2.3 冗余处理

所谓数据有冗余，是指数据中含有重复信息。冗余信息让我们在存储和计算中消耗更多的资源，却并不能提升性能，因此最好去除。

冗余的形式可能表现为二维表格中内容完全重复的行或列。对于重复行，DataFrame 中可以用 duplicated 函数来判断，还可以用 drop_duplicates 函数直接删除重复行，或在指定列上数据重复的行。

例 5-2-3 简单重复行的去除举例。

```
import pandas as pd
student_scores = pd.DataFrame({'姓名':['张三'] * 3 + ['李四'] * 3 + ['王五'] * 3,
                               '成绩':[10,10,10,8,8,8,5,5,5]})
student_scores
```

	姓名	成绩
0	张三	10
1	张三	10
2	张三	10
3	李四	8
4	李四	8
5	李四	8
6	王五	5
7	王五	5
8	王五	5

```
student_scores.duplicated()
0    False
1     True
2     True
3    False
4     True
5     True
6    False
7     True
8     True
dtype: bool
my_fil_data5 = student_scores.drop_duplicates()
my_fil_data5
```

	姓名	成绩
0	张三	10
3	李四	8
6	王五	5

例 5-2-3 中，我们首先人为构造了一个包含重复行的二维表格（数据框），通过 duplicated 函数可以看到每行是否是重复行，在调用 drop_duplicates 函数之后，新的表格则删除了所有的重复行。

对于数据列，其冗余出现的形式会有多种可能性：简单重复、一元线性依赖和多元线性依赖。

简单重复的情况，也就是列或特征直接出现了重复，是很容易被发现和处理的，读取数据时直接通过列名称或特征名筛选即可。

我们更应该警惕的是不同特征也可能出现冗余，即特征间线性依赖的情况。例如，在贷款申请客户的数据中，有可能同时出现的“月平均收入”和“年收入”，如图 5-2-3 所示。两个特征名称不同，同一样本在两列上的具体数据值也不相同。但是，年收入却始终是月平均收入的 12 倍，那么这两个特征包含的信息本质上就是重复的，应选择只保留一个。

	月平均收入	年收入
客户1	x12	
客户2		

图 5-2-3　以贷款申请客户数据为例的非简单列冗余举例

更一般地说，如果一个特征（记为 C_2）可以通过将另一个特征（记为 C_1）线性变换得到，也就是满足

$$C_2 = kC_1 + b \tag{5-2-2}$$

其中，k 和 b 都是常数，那么可以说 C_2 线性依赖于 C_1，这两个特征包含的是重复的信息，两者中可以去掉一个。

判断如式（5-2-2）呈简单线性关系的冗余特征的常用方法是线性相关分析。DataFrame 中有 corr 函数，当其参数 method 设置为 pearson 时可以直接用来求线性相关系数，如果两个特征的线性相关系数接近 1 或－1，则说明两个特征存在强的线性相关或反相关，有着较大的冗余；如果等于 0，则说明两个特征间没有线性相关性。

例 5-2-4　应用 DataFrame.corr 求线性相关系数举例。

```
import pandas as pd
my_data = pd.read_csv("iris.csv", header = None, names = ['sepal_length', 'sepal_width',
          'petal_length', 'petal_width','target'])
print(my_data.corr(method = 'pearson'))
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      1.000000    -0.109369      0.871754     0.817954
sepal_width      -0.109369     1.000000     -0.420516    -0.356544
petal_length      0.871754    -0.420516      1.000000     0.962757
petal_width       0.817954    -0.356544      0.962757     1.000000
```

例 5-2-4 中，我们对鸢尾花数据的 4 个特征每两个之间求相关系数，如示例代码所示。结果显示，petal_length 与 petal_width 两个特征的相关系数为 0.96，即存在着很强的正线性相关。而 sepal_length 和 sepal_width 之间则没有太大的线性关系。所以，如果

要压缩维度，可以考虑在 petal_length 和 petal_width 中只保留一个，与剩下的 sepal_length 和 sepal_width 这三个特征来作为样本的描述即可。

前述主要针对两特征间的冗余，即一个特征完全线性依赖于另一个特征。更复杂地，在某些场合，尽管某个特征不能由另一个特征完全描述，但却可以被另外几个特征的线性组合来表达，即多元线性依赖，如：

$$C_t = \sum_{i \neq t} k_i C_i + b_i \tag{5-2-3}$$

此时，我们通常不再局限于考察两两线性相关，而会直接采用主成分分析（Principle Component Analysis，PCA）的方法来进行去冗余操作。事实上，在涉及特征数目较多的情况时，用 PCA 来去冗余，常常是必要的一步。

先简单介绍一下 PCA 的原理。

以最简单的二维情况为例。假设每个样本数据以特征 C_1 的取值为 x 轴坐标，特征 C_2 的取值为 y 轴坐标，描记为 x-y 平面上的点，如图 5-2-4 中的蓝色点，图中比较明显地呈现出样本的 y 与 x 之间有相互依赖关系。此时当然可以采取直接丢掉一个特征（譬如 y 轴代表的 C_2 特征）的处理方法来去冗余，但更常用的方法是，将现在的坐标系进行旋转，例如将 x 轴旋转到红色的 x' 轴方向，相应地 y 轴则旋转至图中红色的 y' 轴方向。轴 x'、y' 的方向要如何确定呢？PCA 中采取的原则是：x' 的方向的选择，要使得数据集在其上的投影具备最大的方差。二维情况下，x' 的方向找到了，y' 也就找到了，即与 x' 垂直的方向。

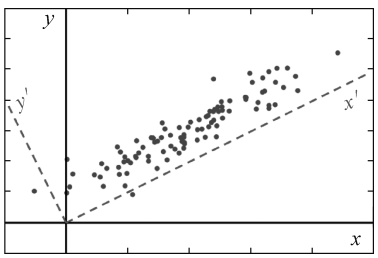


图 5-2-4　二维情况下的 PCA 示意图

接下来，我们有这样一个假设：在 x' 方向上，由于数据集在该方向的投影方差最大，所以最大程度地保存了原始数据的信息；在 y' 方向上，数据集在该方向的投影最小（因为数据的总方差是一定的，x' 方向上方差最大了，剩下的 y' 方向上方差就只能最小了），很有可能只是一些随机扰动，或者对于数据本质不重要的信息。基于这样的假设，我们可以选择保留原始数据在 x' 方向上（不妨称为主方向）的投影（坐标），并称其为主成分（Principle Component），同时忽略掉数据在 y' 方向（可称为次方向）上的波动，这样就实现了用一个主成分来描述原本用 C_1、C_2 两个特征描述的数据。

推广到 n 维，在找到第一个主方向后，将数据中的主成分（记为 PC_1）去掉，在剩下的部分中寻找与第一个主方向垂直，且其上投影方差最大的方向，即第二主方向，并以数据在第二主方向上的投影为第二主成分（记为 PC_2）；类似的操作可以一直进行下去，获得第三主成分（PC_3）、第四主成分（PC_4）、……直到剩下的方差残余足够小，或者已分解得到了 n 个方向，无可再分。随着分解的逐次进行，原始数据集在各分解方向上的投影方差也逐步减小，即满足

$$\mathrm{Var}(PC_1) > \mathrm{Var}(PC_2) > \mathrm{Var}(PC_3) > \cdots > \mathrm{Var}(PC_k) > \varepsilon$$
$$\geqslant \mathrm{Var}(PC_{k+1}) > \cdots > \mathrm{Var}(PC_m)$$

其中，$m \leqslant n$，ε 是我们设定的一个阈值。如果我们认为投影方差小于 ε 的那些方向上的分量都不重要或者是扰动造成(可以转而称呼它们为次成分(Minor Components))，从而只保留 $\mathrm{Var}(\mathrm{PC}_i) > \varepsilon$ 的前 k 个主成分，这就是主成分分析。

主成分分析将原始含冗余的 n 个特征去冗余，形成了更少的 k 个特征。如何理解这里的“去冗余”呢？从线性代数的角度来看，忽略投影方差足够小的次成分，相当于原始数据构成的矩阵不满秩，亦即原有的 n 个维度(特征)间不是线性无关的，有的特征能被其他特征的线性组合来表示，这正是“冗余”的意思。

主成分分析被广泛应用于数据预处理，我们可以理解为去冗余或降维，这对于特征繁多时避免维数灾难具有重要意义。同时，PCA 也可以从另一个角度来理解为一种特征提取：我们获得的主成分往往不是原来的某一个单一特征，而是由多个特征线性组合而来的复合特征，在这一复合特征的描述下，数据体现出比原来在单一特征上更大的方差。

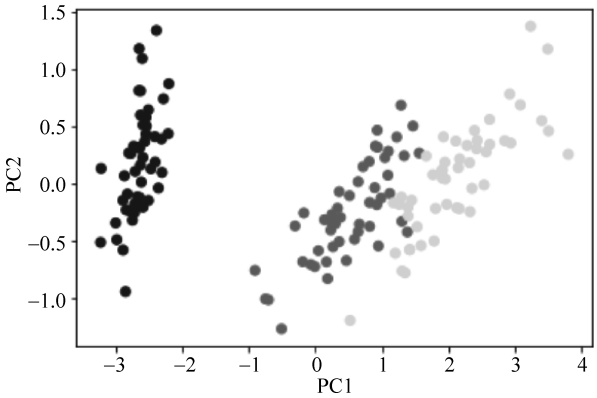
sklearn 库中 decomposition 模块有 PCA 对象可用来做主成分分析。

例 5-2-5 PCA 应用举例。

```
import numpy as np
from sklearn import datasets
from sklearn.decomposition import PCA
from matplotlib import pyplot as plt

my_iris = datasets.load_iris()
print(type(my_iris.data))

my_pca = PCA(n_components = 0.95)
post_proc = my_pca.fit_transform(my_iris.data)
print('原始数据的尺寸是：',my_iris.data.shape,
      '\n 即包含',my_iris.data.shape[1],'维特征度\n')
print('PCA 后留下的主成分数据尺寸是：',post_proc.shape,
      '\n 即满足要求的主成分维度为：',post_proc.shape[1],'\n')
print('PC1 所占总方差的比例为：',my_pca.explained_variance_ratio_[0])
print('PC2 所占总方差的比例为：',my_pca.explained_variance_ratio_[1])
print('两个主成分所占总方差的比例为：',my_pca.explained_variance_ratio_.sum())
print('两个主方向矢量为：')
print(my_pca.components_)
plt.scatter(post_proc[:,0],post_proc[:,1],c = my_iris.target)
plt.gca().set_xlabel('PC1')
plt.gca().set_ylabel('PC2')
plt.show
```

```
<class 'numpy.ndarray'>
原始数据的尺寸是： (150, 4)
即包含 4 维特征度

PCA后留下的主成分数据尺寸是： (150, 2)
即满足要求的主成分维度为： 2

PC1所占总方差的比例为： 0.9246187232017271
```

```
PC2所占总方差的比例为： 0.053066483117067825
两个主成分所占总方差的比例为： 0.977685206318795
两个主方向矢量为：
[[ 0.36138659 -0.08452251  0.85667061  0.3582892 ]
 [ 0.65658877  0.73016143 -0.17337266 -0.07548102]]
```



例 5-2-5 中，我们尝试对 sklearn 库 datasets 中自带的鸢尾花数据进行了 PCA。具体是先生成一个名为 my_pca 的 PCA 对象实例，其中设置参数 n_ components＝0.95，意为只保留投影方差累积刚超过总体 95％的前 k 个分量。当然，也可以把 n_ components 直接设置为想保留的主成分的个数。然后，调用 PCA 对象中内置的 fit_transform 函数对数据进行主成分分解，并将数据在主方向上的投影，亦即主分量，返回到我们命名为 post _proc 的数组中。我们检查一下处理前的数据和处理后的数据尺寸，输出区结果显示，原始数据包含 4 个特征，而执行 PCA 后只保留两个主分量了，即特征维度从 4 降为 2。程序中又接着检查了主分量的方差，在 PCA 的属性 explained_variance_ratio_中，输出区结果显示第 1 主分量方差占总方差的约 92％，第 2 主分量方差占总方差的约 5％，两者之和正好超过我们在 n_ components 设置的 0.95，这也解释了为什么返回的主分量正好是两维，而非其他数目。我们新的坐标轴，即两个主方向（由原始坐标系下的 4 维矢量表示）则保存在 PCA 的属性 components_中。例程序的最后，我们以 PC_1、PC_2 分别为坐标，画出了 150 个鸢尾花数据在新坐标系下的散点图。可以将输出区的结果图与后面的图 5-2-5（即以鸢尾花原始 4 个特征两两配对画出的共计 6 幅图）做对比。可以看到，基于 PC_1、PC_2 的散点图与图 5-2-5 中任何一幅都不相同。这是因为 PC_1、PC_2 已经不是任何一个单一特征，而是 4 个原始特征的某种线性组合，而且，在 PC_1 上，150 个数据能获得最大的方差。

经过 PCA 去除冗余后，我们就可以把返回的 k 维主分量数据直接作为新的（复合）特征，送给后续的模型进行处理了。

本节主要介绍了缺失处理、异常处理和冗余处理几种基本的预处理方法，对于其中涉及的一些专业名词（如正态分布、四分位数、散点图等）并未深入解释，在接下来的描述性统计中将给出更详细的介绍。

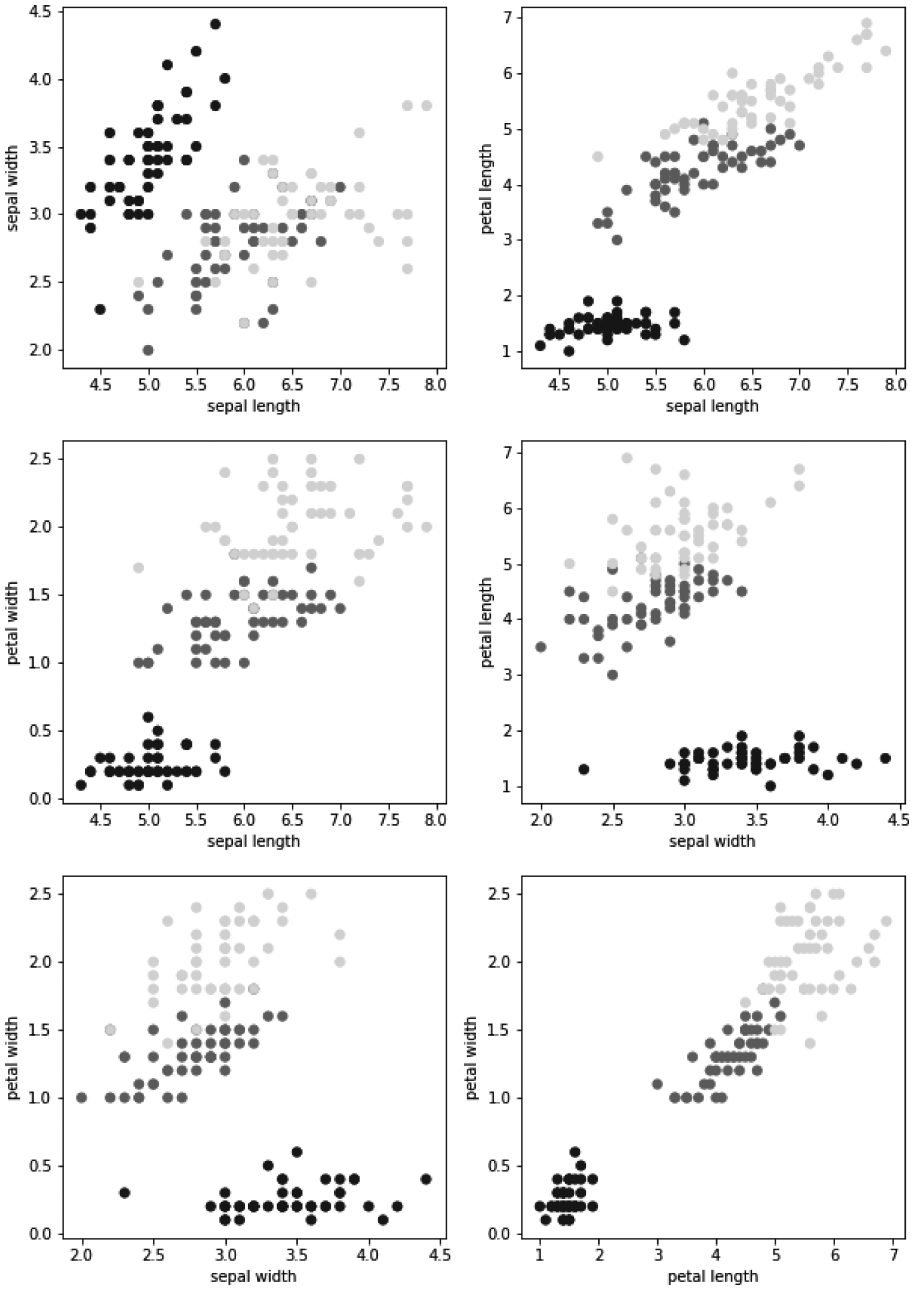


图 5-2-5　鸢尾花基于两两特征的散点图

5.3　描述性统计

在对数据进行了基本的预处理之后，就可以进一步分析了。在 EDA 中，我们主要对数据进行描述性统计，包括位置性测度、离散性测度的计算，以及图形化描述。其中位置

性测度和离散性测度都只能针对数值型数据计算，图形化描述则既有适用于数值型数据的，也有适用于非数值型数据的。

5.3.1 位置性测度

常用的位置性测度有算术平均、中位数、p 百分位数、众数等。

算术平均是对所有考察的样本值求统计平均，即

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i \tag{5-3-1}$$

算术平均是最常用的统计量，但是容易受样本中极端值的影响。

例 5-3-1 算术平均计算举例。

10 个样本的某数值型特征分别为：2、3、1、2.5、3.3、4、4.5、2、3、100，其算术平均 $\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i=\frac{2+3+1+2.5+3.3+4+4.5+2+3+100}{n}=12.53$。

例 5-3-1 中的 10 个样本，有 9 个都处于 1～5 之间，但第 10 个取值为 100(极端值)，则它们的统计平均值会大大高于前 9 个样本的真实水平。这也是每年各地统计局发布年度薪酬数据时，很多人感觉自己“被加薪”的原因之一。

中位数是将所有样本按数值从小到大或从大到小排序以后，最中间位置的一个数(当样本容量为奇数时)，或者两个数的平均(当样本容量为偶数时)。

例 5-3-2(a) 中位数计算举例。

10 个样本的某数值型特征分别为：2、3、1、2.5、3.3、4、4.5、2、3、100，对其从小到大排序为：1、2、2、2.5、3、3、3.3、4、4.5、100，其最中间是排序第 5、6 的两个数，即 3、3，两数平均值也为 3，所以这 10 个样本在该特征的中位数为 3。

中位数对极端值不敏感，但是对于中位数以外的所有值也都不敏感。

例 5-3-2(b) 中位数局限举例。

10 个样本的某数值型特征对其从小到大排序为：－10、－10、－10、－5、3、3、7、8、9、10，其最中间是排序第 5、6 的两个数，即 3、3，两数平均值也为 3，所以这 10 个样本在该特征的中位数为 3。

例 5-3-2(b)中，我们把除了正中间的数以外其他数值全修改了，整个集合变化很大，但中位数却并没有改变。可见，只考察中位数也是不够的。更细致地，可考察第 p 百分位数。

第 p 百分位数，是将所有样本值按从小到大的顺序排好，排序在第 $p\%$ 的样本取值。如果将第 p 百分位数记为 V_p，那么样本中有且仅有 $p\%$ 的样本观察值小于或等于 V_p。常用的 p 百分位数有第 10 百分位数、第 25 百分位数(又分别称为第 1 四分位数)、第 75 百分位数(又称第 3 四分位数)、第 90 百分位数，等等。其实中位数就是第 50 百分位数。当样本容量足够大以后，第 p 百分位数是相对稳定的，并且不会受样本容量的影响。

众数是指样本集中出现次数最多的那个值。一个样本集中，众数可能只有一个，此

时可称数据是单峰分布；也可能出现两个甚至更多的众数，此时可称数据为双峰/多峰分布。尽管对数值型特征可以求众数，但对于在连续区间上取值的浮点数求众数通常意义不大，因为在理论的无限精度下，样本浮点型观测值完全相等的情况是很少的，反而是对非连续取值，甚至是类别型数据，求众数要更具现实意义。

Pandas 的数据框结构中提供现成的函数，可以很方便地求取各种位置性测度。

例 5-3-3 Pandas 数据框求位置性测度举例。

```
import pandas as pd
import numpy as np
my_data = pd.read_csv("Titanic.csv")
print('对 Fare 的位置性测度统计结果：')
print('均值：\t\t',my_data[['Fare']].mean()[0])
print('中位数：\t',my_data[['Fare']].median()[0])
print('第 25 百分位数：',my_data[['Fare']].quantile(q=0.25)[0])
print('众数：\t\t',my_data[['Fare']].mode().values[0,0])
```

```
对Fare的位置性测度统计结果：
均值：                32.204207968574636
中位数：    14.4542
第25百分位数：    7.9104
众数：                8.05
```

如例 5-3-3 所示，对于我们之前看过的 Titanic 数据，其中明确是数值数据的“船费”，我们来试着求均值等统计量。read_csv 命令读入的 my_data 是一个 DataFrame 结构，我们对 my_data 进行切取，取出其中的“船费”列，注意这里使用两重中括号形式的切取，返回依然是 DataFrame 结构。所以，可以就用 DataFrame 结构中封装的 mean、median、quantile、和 mode 函数。其中 quantile 函数需指定参数 q，q 取值 0～1，代表要求的具体百分位数，例如这里设置 $q=0.25$，就是求第 25 百分位数了。这里，mean、median、quantile 返回的都是 Pandas 的 series 序列结构，可以直接用方括号加序号来访问；而 mode 返回的是 DataFrame 结构，要获取其中的值，可读取 values 属性，values 本身是 ndarray 结构，所以用访问二维数组的方式来访问。运行一下代码，可以看到 cell 下的输出区显示出了对 Fare 的统计结果。

位置性测度主要用来反映样本集合的中心成员或特定成员在所考察的数域或空间中的位置。但是显然，样本集作为一个集合，只靠少数成员来描述其位置是不够的，众多成员相对于这些特定位置是集中还是分散，也是数据集的重要特征。所以，同时还要关注所谓的离散性测度。

5.3.2 离散性测度

常用的离散性测度有极差、方差或标准差、变异系数等。

极差是指集合中最大值与最小值之间的差异。显然极差只由分处两端的值决定，因此对极端值非常敏感。

方差是对集合中所有样本值相对于均值的偏差的平方求近似平均，常记为 Var，方差

的平方根称为标准差，常用符号 s 表示。即

$$\mathrm{Var}=\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1},\quad s=\sqrt{\mathrm{Var}} \tag{5-3-2}$$

方差与标准差可以总体衡量集合中数据偏离均值的程度，而且标准差与数据量纲相同。一般而言，方差或标准差越大，代表样本的离散性或样本间差异性越大。

但是，在不同的物理量之间，方差或标准差是不好直接拿来做比较的，因为不同的物理量单位不同，其常规取值的数量级也可能不同。为消除物理量自身单位（量纲）的影响，还会引入一个变异系数的概念。

变异系数是标准差除以均值再乘以 100%，即

$$\mathrm{CV}=\frac{s}{\bar{x}}\times 100\% \tag{5-3-3}$$

可见，变异系数是一个无量纲或者说无单位的数，能尽量消除量纲及均值的绝对位置带来的影响。

我们可以在例 5-3-3 的基础上，来试试利用 DataFrame 的封装函数计算船费的离散性测度。

例 5-3-4 Pandas 数据框求离散性测度举例。

```
print('对 Fare 的离散性测度统计结果：')
print('变化范围：\t [', my_data[['Fare']].min()[0], '\t', my_data[['Fare']].max()[0], ']')
print('极差：\t\t',my_data[['Fare']].max()[0] - my_data[['Fare']].min()[0])
print('方差：\t\t',my_data[['Fare']].var()[0])
print('标准差：\t',my_data[['Fare']].std()[0])
print('变异系数：\t',my_data[['Fare']].std()[0]/my_data[['Fare']].mean()[0])
```

```
对Fare的离散性测度统计结果：
变化范围：          [ 0.0   512.3292 ]
极差：              512.3292
方差：              2469.436845743116
标准差：    49.6934285971809
变异系数：          1.5430725278408497
```

```
my_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

DataFrame 没有封装专门的极差函数，但是我们可以通过其求最大、最小值的 max 和 min 两个函数计算获得。var 函数和 std 函数则可直接用来求方差和标准差，变异系

数没有对应函数，但我们用 std 除以 mean 就可以得到了。

DataFrame 中还有一个好用的 describe 函数，可以对 DataFrame 中所有用数值保存的特征（无论是整数还是浮点数），一次性计算多个常用的描述性统计量，包括前面介绍的均值、标准差、百分位数等；此外，还会对该列所有非空元素计数并将结果返回在 count 行中。

5.3.3 图形化描述统计

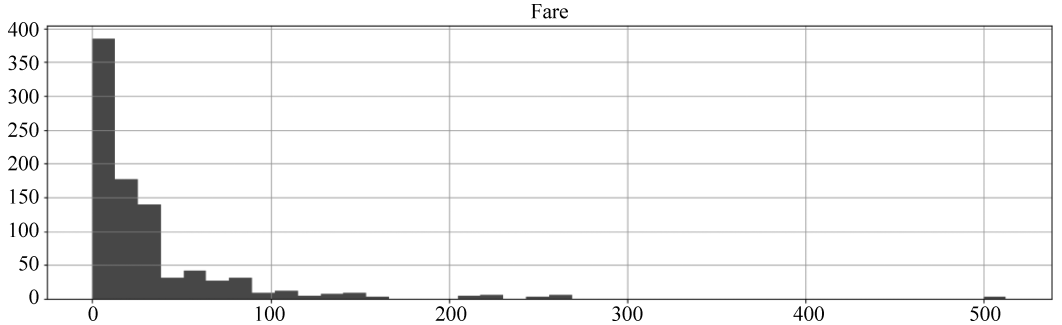
除了采用前述的统计量，我们还可以用图形化的方式（例如直方图、箱型图等）来对数据进行更为细致、直观的观察。

直方图是一种反映数据分布的柱状统计图。做直方图时，我们通常先对数据进行分组，将不同的组对应于画到不同的 x 轴位置的条柱，然后，将落入各组内的样本个数作为条柱的高度 y。

在前面代码的基础上，仍以 Titanic 数据中的船费特征为例做直方图，可以直接使用 Dataframe.hist 函数。

例 5-3-5 直方图绘制举例。

```
my_data[['Fare']].hist(bins = 40,figsize = (18,5),xlabelsize = 16,ylabelsize = 16)
```



指定参数 bins＝40，代表着我们将船费的取值范围（0～513）分成了均等的 40 个区间，例如第 1 个区间是 0 到大约 12.8，第 2 个区间是 12.8 到大约 25.6，依此类推。绘制直方图时，船费落入同一个区间的样本会被作为同一个组，然后，对每一组统计其中的样本个数，作为对应位置上条柱的高度画出来。从结果图上来看，尽管最贵的船费是约 513 元，可能是总统套房的价格，但是大部分人的船费其实并未超过 25.6 元，并且买最便宜船票（落入第一组），也就是船费 0～12.8 区间的人在 40 个分组里面是最多的。

箱型图是另一种刻画分布的常用图形化方法（图 5-3-1）。箱型图中，并不像直方图中画出所有样本，而是通过几个重要的百分位数来界定数据的主要分布。

如图 5-3-1 所示，箱型图中箱子的下、上边缘分别由第 1 四分位数 q_1 和第 3 四分位数 q_3 来决定，箱子中的线则代表中位数。箱子的高度 a_3-q_1，暂记为 Δ，就是常说的四分位距（IQR）。箱子两端伸出的虚线用来刻画数据的极差，向下延伸到数据集中大于等于 $q_1-1.5\Delta$ 的所有数据中的最小值，以下边缘线作为结束；向上延伸到数据集中小于

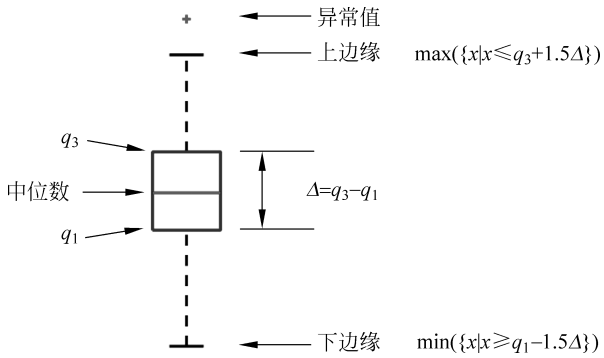
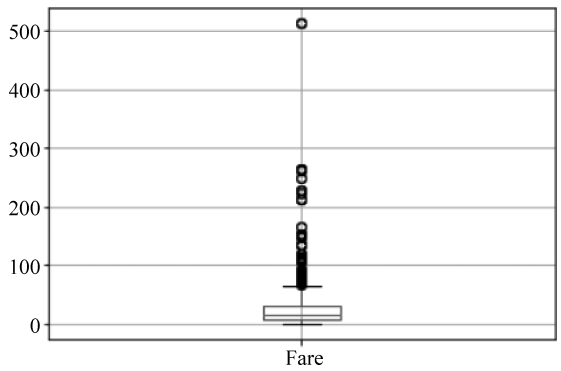


图 5-3-1 箱型图示意

或等于 $q_3+1.5\Delta$ 的所有数据中的最大值，以上边缘线作为结束。超出上、下边缘线的数据，被当作异常值，或称为离群点(outlier)。

例 5-3-6 箱型图绘制举例。

```
my_data[['Fare']].boxplot()
```



例 5-3-6 中，我们依然针对船费用 DataFrame 的 boxplot 函数画出其箱型图。由结果图可以看到，由于船费的分布严重偏歪，所以上边缘线之外有很多离群点。这与我们之前看到的统计量和直方图所反映的信息是一致的。

迄今，我们都是在介绍对数值型特征的图形化描述统计。其实图形化描述不仅仅限于数值型特征分析，对非数值型特征也是适用的。具体如何分析呢？

非数值型的特征，我们主要应用来分组，分组后，可以对各组进行频次统计，绘制与直方图类似的柱状图；还可以基于分组，对其他的数值型特征进行更为细致的分组统计。

例 5-3-7 利用非数值型数据分组绘图举例。

```
#查看数据在不同舱位等级分组的分布
import pandas as pd

my_data = pd.read_csv("Titanic.csv")
```

```
my_plot_data = my_data[['Pclass']].groupby(['Pclass']).size()
print(my_plot_data)
my_plot_data.plot(kind = 'bar')

#分组统计
print('表 1. 按舱位等级分组求船费、年龄、同行平辈人数、同行父母和子女人数的均值')
print(my_data[['Fare','Age','SibSp','Parch','Pclass']].groupby(['Pclass']).mean())

print('\n\n 表 2. 按舱位等级分组求船费、年龄、同行平辈人数、同行父母和子女人数的标准差')
print(my_data[['Fare','Age','SibSp','Parch','Pclass']].groupby(['Pclass']).std())
 Pclass
 1    216
 2    184
 3    491
 dtype: int64

<matplotlib.axes._subplots.AxesSubplot at 0x194437b8>
```

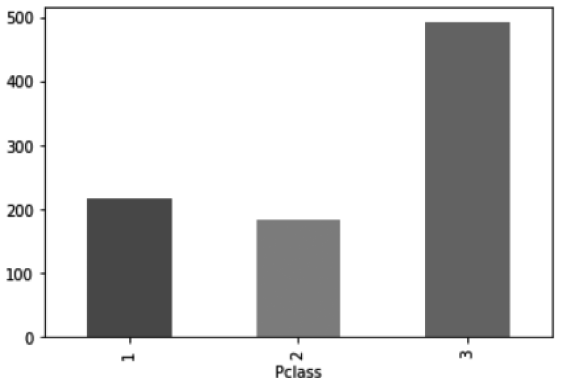


表 1. 按舱位等级分组求船费、年龄、同行平辈人数、同行父母和子女人数的均值

Pclass	Fare	Age	SibSp	Parch
1	84.154687	38.233441	0.416667	0.356481
2	20.662183	29.877630	0.402174	0.380435
3	13.675550	25.140620	0.615071	0.393075

表 2. 按舱位等级分组求船费、年龄、同行平辈人数、同行父母和子女人数的标准差

Pclass	Fare	Age	SibSp	Parch
1	78.380373	14.802856	0.611898	0.693997
2	13.417399	14.001077	0.601633	0.690963
3	11.778142	12.495398	1.374883	0.888861

在例 5-3-7 中，我们依据舱位等级的不同，利用 DataFrame 中的 groupby 函数对数据进行分组，先通过调用 groupby 返回的 DataFrameGroupBy 对象的 size 方法统计各舱位等级的样本个数，然后对 size 方法返回的 Series 对象 my_plot_data，利用其支持的 plot 方法，设置参数 kind＝bar，就可以绘制出反映乘客舱位等级分布的柱状图。其中，pandas 中 Series 结构支持的 plot，其实就是 matplotlib 库中定义的 plot 方法，所以其使用方法基本一致。

绘图结果显示，选择三等舱的人是最多的。这种柱状图可直观反映数据在不同组上的分布，其作用与数值型特征的直方图是类似的。

接着，我们可以借助于舱位等级，对船费等数值化特征进行更细致的描述性统计。例如，我们对所有有量化意义的数值型特征，都根据舱位等级分组后求平均值或标准差，这只要对 groupby 返回的 DataFrameGroupBy 对象调用其 mean 函数或 std 函数即可实现。

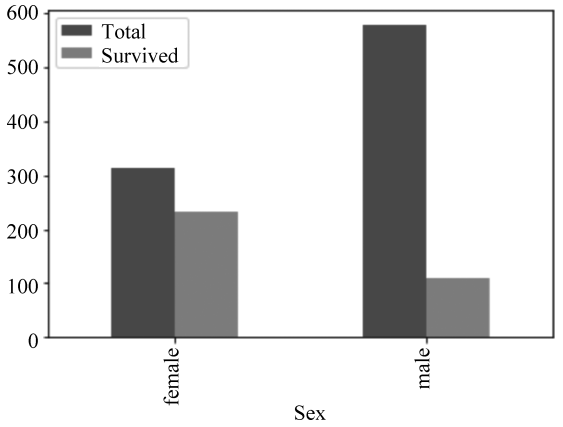
其实，在数据分析中，分组是非常重要的一步。在第 3 章介绍过，对混杂因素实施匹配分组能有效避免辛普森悖论，以及在 A/B Testing 中寻找组与组之间的差异或联系。所以数据中那些非数值型的特征，尽管不具备量化意义，似乎无法进行很多统计量的计算，其实却对分析有着重要作用。我们再来看一个图形化的分组对比例子。

Titanic 数据集是对当年“泰坦尼克号”上的部分人员的信息记录。以前有一位学生对于 Titanic 数据中幸存者的男女性别分布产生了兴趣，想要通过数据看看情况是否与报道吻合。他是这样做的：

例 5-3-8 对 Titanic 数据集中性别分布研究举例。

```
import pandas as pd
my_data = pd.read_csv("Titanic.csv")
#借助分组、筛选的图形化描述
gender_dst_org = my_data[['Sex']].groupby(['Sex']).size()
print('数据文件中全部非空数据的性别情况')
print(gender_dst_org,'\n')
my_filter = my_data[my_data.Survived == 1]                    #DataFrame 的数据筛选
gender_dst_srv = my_filter[['Sex']].groupby(['Sex']).size()
print('数据文件中幸存者的性别情况')
print(gender_dst_srv,'\n')
my_tmp = pd.concat([gender_dst_org,gender_dst_srv],axis = 1)    #数据连接,axis = 1 表示增加列
my_plot_data = my_tmp.rename(columns = {0:'Total',1:'Survived'})    #列重命名
print(my_plot_data)
my_plot_data.plot(kind = 'bar')
```

```
数据文件中全部非空数据的性别情况
Sex
female    314
male      577
dtype: int64

数据文件中幸存者的性别情况
Sex
female    233
male      109
dtype: int64

        Total  Survived
Sex
female    314       233
male      577       109
```



例 5-3-8 中，他首先对全部数据根据性别分组，得到了一个性别分布；然后，他专门筛选出幸存者(Survived＝＝1)的数据，并对幸存者根据性别分组，获得了幸存者的性别分布；最后，他用柱状图将两组分布绘制在同一张图上，方便对比。这也是一个典型的利用非数值化特征进行筛选、分组，然后图形化描述的示例。

到这里，不知道读者是否注意到，一开始，对于数据的研究是在分立的维度，例如，计算船费的各种统计量；但是，当结合了非数值特征的分组以后，实际上就扩展到两个维度了，也就是开始将两个特征结合在一起考虑，例如不同舱位等级的船费，既有船费，又有舱位等级的考察；再如研究幸存者的性别分布时，除了性别本身，其实还考察了幸存与否的影响。事实上，不仅非数值型特征可以结合数值型特征做两个维度的考察，两个数值型特征也是可以结合起来考虑的，基本的图形化方法就是散点图。

二维散点图，是用样本在一个特征，例如特征 1 上的取值作为横轴，在另一个要关联考察的特征，例如特征 2 上的取值作为纵轴，这样在二维平面上确定下该样本的位置，描绘出一个样本点。所有的样本都依据同样的方法在二维平面上以点的形式绘出，如图 5-3-2 所示，这样就画出了样本集在特征 2-特征 1 平面中的二维散点图。类似地，当扩展到 3 个不同特征时，我们还可以做出三维特征空间中的散点图。借助于散点图，我们能快速直观地观察到样本在两个或三个特征上的分布，以及特征两两之间有没有相互依赖的关系。

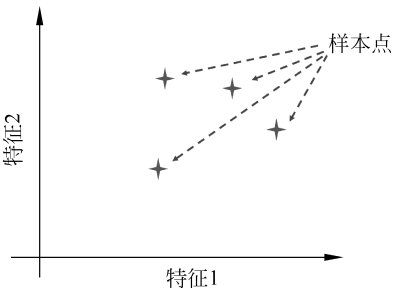


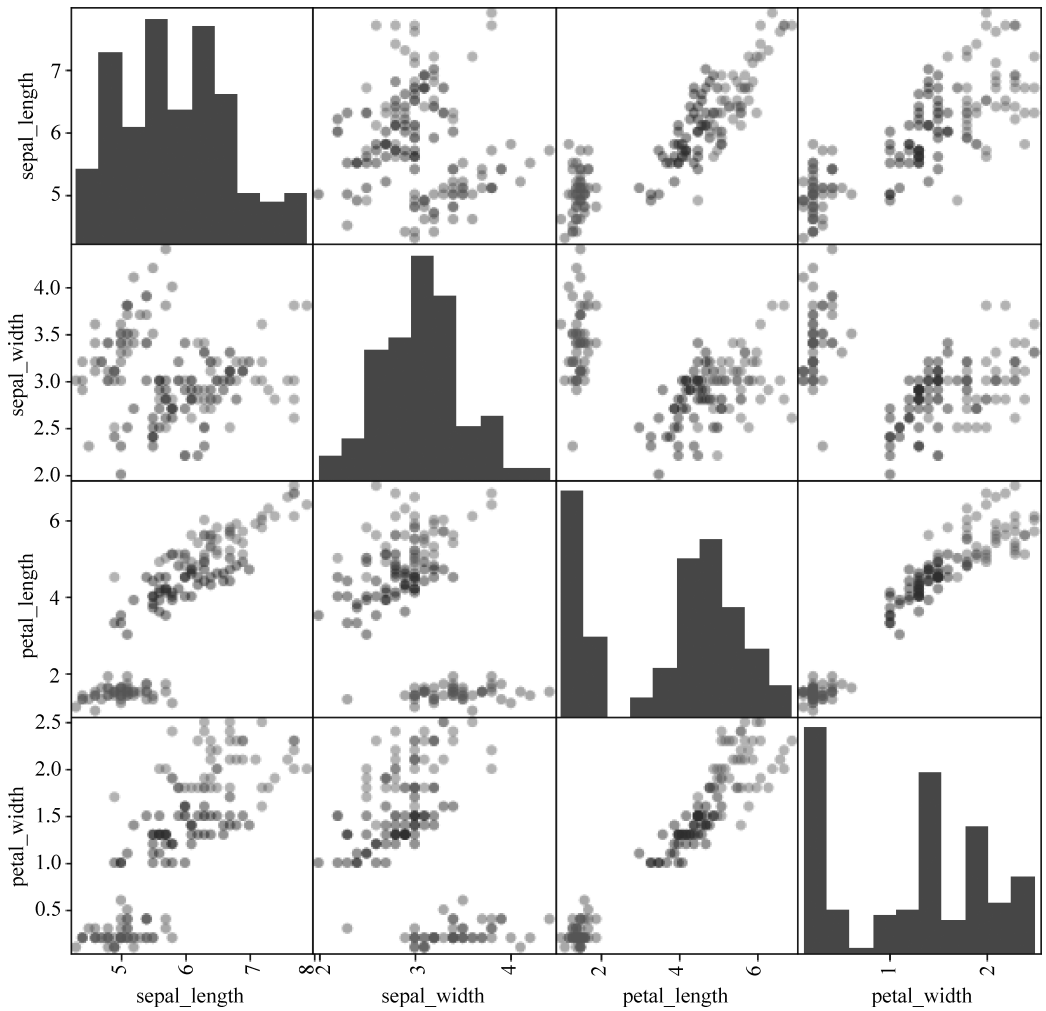
图 5-3-2　二维散点图示意

我们仍以鸢尾花数据集为例，来看一下 Python 中的散点图绘制。

例 5-3-9　散点图绘制举例。

```
import pandas as pd
my_data = pd.read_csv("iris.csv",header = None,
                      names = ['sepal_length','sepal_width','petal_length',
                               'petal_width','target'])
my_set = set(my_data['target'])          #创建一个类别名集合
```

```
my_set_list = list(my_set)                    #set 不能直接访问其元素,转换成 list 后可以访问
colors = list()
palette = {my_set_list[0]:"red",my_set_list[1]:"green",my_set_list[2]:"blue"} #字典,给
#三种类别对应散点图中的三种 marker_color
for n,row in enumerate(my_data['target']): #根据类别为每个样本设置绘图颜色
    colors.append(palette[my_data['target'][n]])

#对 my_data 中的数值型数据,每两个特征绘制散点图
scatterplot = pd.plotting.scatter_matrix(my_data,alpha = 0.3,
                                         figsize = (10,10), diagonal = 'hist',color = colors,
                                         marker = 'o',grid = True)
```



例 5-3-9 中,我们采用的是 pandas.plotting 中的 scatter_matrix 方法。这个函数可以对数据框中的所有数值型的特征,按每两个特征绘制一幅二维散点图。鸢尾花数据共有 4 个数值型特征,所以,可以绘制出 4×3 共计 12 幅两不同特征的二维散点图。大图是一个 4×4 矩阵形式,可以看到,对角线之外,就是上述 12 幅散点图。大图对角线的位

置，本来应该是以同一特征分别为横轴和纵轴坐标来画样本点，但如果那样作图，散点图会是什么样子呢？是的，会成为一条 $y=x$ 的直线，没有太多意义。所以我们通过设置 scatter_matrix 方法中的参数 diagonal=hist 选择显示对应特征的直方图，来观察样本在单个特征上的数据分布。例 5-3-9 的示例程序中，还可以注意一下我们利用字典（自定义的字典结构对象 palette）、枚举（enumerate 函数）和 for 循环的方法为不同类别设置不同画图颜色的方法。其中 for 循环，其意义是对集合中的每一个鸢尾花样本，根据其标签的类别，赋以在字典 palette 中所指定的对应颜色。

本节主要介绍了基本的描述性统计。描述性统计不对数据做任何预先的猜想，只是实事求是地告诉我们样本数据是怎样的。而我们则需要在描述性统计结果的基础上进行思考，形成一些初步结论或假设。例如，根据本节例题中对 Titanic 数据分组统计均值的结果，你有什么想法吗？再如，根据其中性别分布的图形化描述，你有没有得出什么初步的结论？

有读者说，头等舱的船费高于二等舱，二等舱的船费又高于三等舱；还有读者说，随着舱位等级变化，乘客的平均年龄也在变化，头等舱的乘客最年长，三等舱的乘客最年轻，二等舱则正好处于前两者之间；更眼尖的读者则注意到，三等舱的乘客，相较头等舱和二等舱乘客，会有更多的出行同伴，二等舱乘客的同行平辈人数最少，头等舱的同行父母子女人数最少。而根据对性别的分布对比，我们似乎可以非常肯定，全部数据中男性更多，而幸存者中女性更多，所以逃生时确实是女士优先的。

真的是这样的吗？事实上，上述所有的初步结论都是需要后续进一步验证的，因为这些都是我们从现有数据上获得的直接描述。但是，现有数据并不是 Titanic 乘客的全体，当年船上有超过 2000 人。那么这些描述能代表船上所有人的情况吗？其实，用样本的统计量反映总体的过程，称为统计推断。统计推断都要经过相应的统计检验，我们会在后续的统计建模中做进一步介绍。现在，暂且先记下你对于描述性统计结果做出的初步论断吧。

到这里，EDA 的任务就基本可算完成了。接下来，我们将会进入到建模阶段，首先我们会介绍统计学建模的方法。

思考题

5-1　针对例 5-3-9 得出的散点图，你能提出哪些后续供验证的假设？或者说，你获得了进一步要做什么的提示？