

## 23

## Basic Computations in Pandas DataFrame

## Pandas 常见运算

特别介绍 `groupby()`、`apply()` 方法

别弄乱了我的圆！

*Don't disturb my circles!*

—— 阿基米德 (Archimedes) | 数学家、发明家、物理学家 | 287 ~ 212 BC



- ◀ `pandas.DataFrame.apply()` 将一个自定义函数或者 `lambda` 函数应用到数据帧的行或列上，实现数据的转换和处理
- ◀ `pandas.DataFrame.corr()` 计算 `DataFrame` 中列之间 Pearson 相关系数（样本）
- ◀ `pandas.DataFrame.count()` 计算 `DataFrame` 每列的非缺失值的数量
- ◀ `pandas.DataFrame.cov()` 计算 `DataFrame` 中列之间的协方差矩阵（样本）
- ◀ `pandas.DataFrame.describe()` 计算 `DataFrame` 中数值列的基本描述统计信息，如平均值、标准差、分位数等
- ◀ `pandas.DataFrame.groupby()` 在分组后的数据上执行聚合、转换和其他操作，从而对数据进行更深入的分析 and 处理
- ◀ `pandas.DataFrame.kurt()` 计算 `DataFrame` 中列的峰度（四阶矩）
- ◀ `pandas.DataFrame.kurtosis()` 计算 `DataFrame` 中列的峰度（四阶矩）
- ◀ `pandas.DataFrame.max()` 计算 `DataFrame` 中每列的最大值
- ◀ `pandas.DataFrame.mean()` 计算 `DataFrame` 中每列的平均值
- ◀ `pandas.DataFrame.median()` 计算 `DataFrame` 中每列的中位数
- ◀ `pandas.DataFrame.min()` 计算 `DataFrame` 中每列的最小值
- ◀ `pandas.DataFrame.mode()` 计算 `DataFrame` 中每列的众数
- ◀ `pandas.DataFrame.nunique()` 计算 `DataFrame` 中每列中的唯一值数量
- ◀ `pandas.DataFrame.quantile()` 计算 `DataFrame` 中每列的指定分位数值，如四分位数、特定百分位等
- ◀ `pandas.DataFrame.rank()` 计算 `DataFrame` 中每列元素的排序排名
- ◀ `pandas.DataFrame.skew()` 计算 `DataFrame` 中列的偏度（三阶矩）
- ◀ `pandas.DataFrame.std()` 计算 `DataFrame` 中列的标准差（样本）
- ◀ `pandas.DataFrame.sum()` 计算 `DataFrame` 中每列元素的总和
- ◀ `pandas.DataFrame.var()` 计算 `DataFrame` 中列的方差（样本）



## 23.1 四则运算

在 Pandas 中，可以通过简单的语法实现各列之间的四则运算。以鸢尾花数据帧为例，图 1 中代码所示为鸢尾花数据帧花萼长度 ( $X_1$ )、花萼宽度 ( $X_2$ ) 两列之间的运算。

**a** 对花萼长度去均值 (demean)，即  $X_1 - E(X_1)$ 。其中，`X_df['X1'].mean()` 计算列均值。也可以用 `pandas.DataFrame.sub()` 完成减法运算。

**b** 对花萼宽度去均值，即  $X_2 - E(X_2)$ 。

**c** 计算花萼长度、宽度之和，即  $X_1 + X_2$ 。也可以用 `pandas.DataFrame.add()` 完成加法运算。

**d** 计算花萼长度、宽度之差，即  $X_1 - X_2$ 。

**e** 计算花萼长度、宽度乘积，即  $X_1 X_2$ 。也可以用 `pandas.DataFrame.mul()` 完成乘法运算。

**f** 计算花萼长度、宽度比例，即  $X_1/X_2$ 。也可以用 `pandas.DataFrame.div()` 完成除法运算。

```
import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧

X_df = iris_df.copy()
X_df.rename(columns = {'sepal_length': 'X1',
                      'sepal_width': 'X2'},
            inplace = True)
X_df_ = X_df[['X1', 'X2', 'species']]

# 数据转换
a X_df_['X1 - E(X1)'] = X_df_['X1'] - X_df_['X1'].mean()
b X_df_['X2 - E(X2)'] = X_df_['X2'] - X_df_['X2'].mean()
c X_df_['X1 + X2'] = X_df_['X1'] + X_df_['X2']
d X_df_['X1 - X2'] = X_df_['X1'] - X_df_['X2']
e X_df_['X1 * X2'] = X_df_['X1'] * X_df_['X2']
f X_df_['X1 / X2'] = X_df_['X1'] / X_df_['X2']
X_df_.drop(['X1', 'X2'], axis=1, inplace=True)

# 可视化
sns.pairplot(X_df_, corner=True, hue="species")
```

图 1. 鸢尾花数据帧花萼长度 ( $X_1$ )、花萼宽度 ( $X_2$ ) 两列之间的运算

图 2 所示为经过上述转换后用 `seaborn.pairplot()` 绘制的成对特征散点图。我们在鸢尾花书《统计至简》还会用到这幅图。

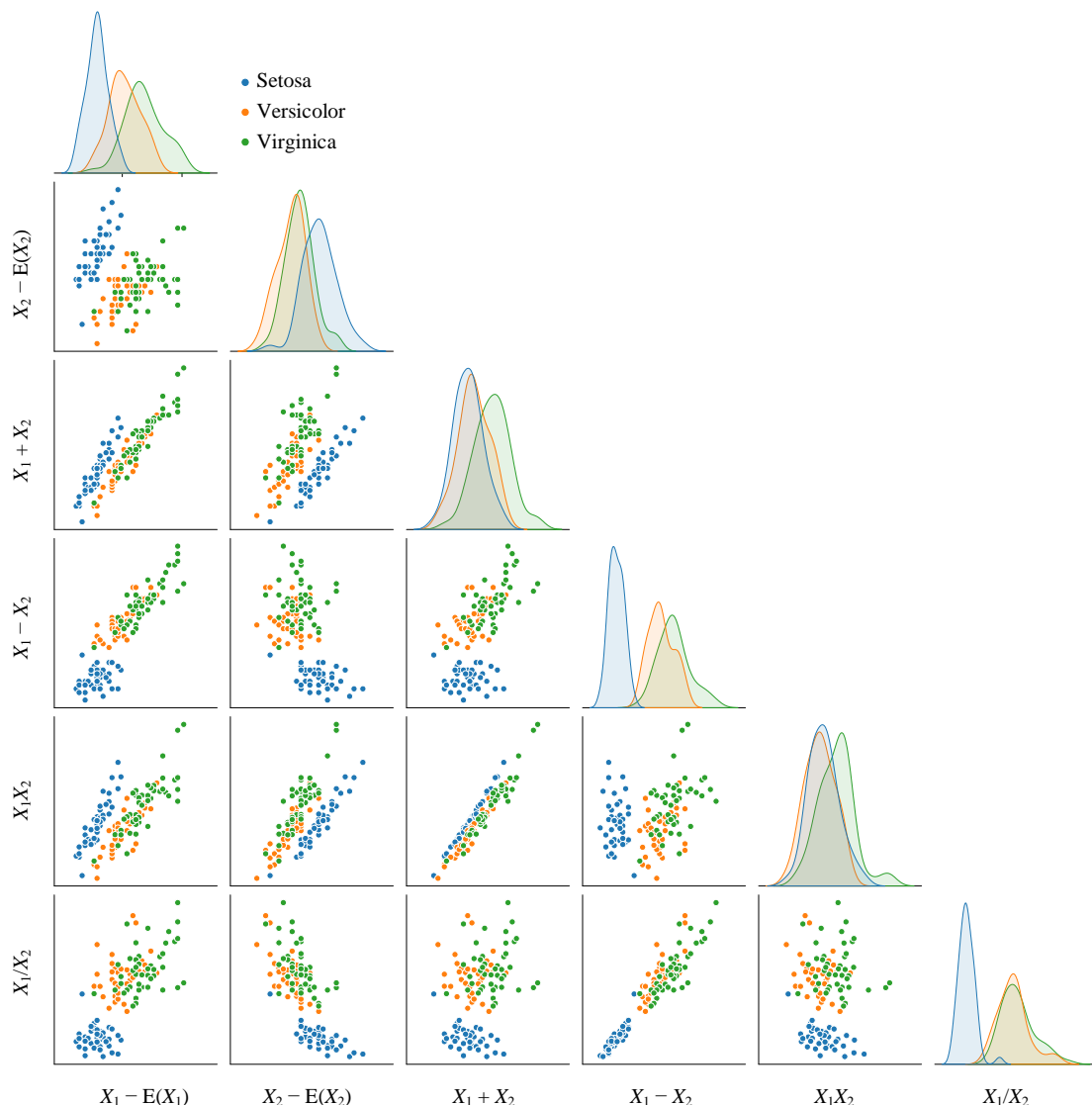


图 2. 鸢尾花花萼长度、宽度特征完成转换后的成对特征散点图

## 23.2 统计运算

Pandas 中还给出大量用于统计运算 (也叫聚合操作) 的方法, 表 1 总结常用的几种方法。

在数据分析中, 聚合操作 (aggregation) 通常用于从大量数据中提取出有意义的摘要信息, 以便更好地理解数据的特征和行为。

常见的聚合操作包括计算平均值、求和、计数、标准差、方差、相关性等。这些操作可以帮助我们了解数据的集中趋势、离散程度、相关性等特征, 从而做出更准确的分析和决策。

图 3 所示为 `pandas.DataFrame.cov()` 计算得到的鸢尾花前四列协方差矩阵热图。当然, 在计算协方差时, 我们也可以考虑到数据标签。图 5 所示为三个不同标签数据各自的协方差矩阵、相关性系数热图。

此外, `pandas.DataFrame.agg()` 方法用于对 `DataFrame` 中的数据进行自定义聚合操作。该方法按照指定的函数对数据进行聚合, 可以是内置的统计函数, 也可以是自定义的函数。

本 PDF 文件为作者草稿, 发布目的为方便读者在移动终端学习, 终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有, 请勿商用, 引用请注明出处。

代码及 PDF 文件下载: <https://github.com/Visualize-ML>

本书配套微视频频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教, 本书专属邮箱: [jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

比如，`iris_df.iloc[:,0:4].agg(['sum', 'min', 'max', 'std', 'var', 'mean'])` 对鸢尾花数据帧前四列进行各种统计计算。

表 1. Pandas 中常用统计运算方法

| 函数名称                                     | 描述                                       |
|--|--|
| <code>pandas.DataFrame.corr()</code>     | 计算 DataFrame 中列之间 Pearson 相关系数（样本）       |
| <code>pandas.DataFrame.count()</code>    | 计算 DataFrame 每列的非缺失值的数量                  |
| <code>pandas.DataFrame.cov()</code>      | 计算 DataFrame 中列之间的协方差矩阵（样本）              |
| <code>pandas.DataFrame.describe()</code> | 计算 DataFrame 中数值列的基本描述统计信息，如平均值、标准差、分位数等 |
| <code>pandas.DataFrame.kurt()</code>     | 计算 DataFrame 中列的峰度（四阶矩）                  |
| <code>pandas.DataFrame.kurtosis()</code> | 计算 DataFrame 中列的峰度（四阶矩）                  |
| <code>pandas.DataFrame.max()</code>      | 计算 DataFrame 中每列的最大值                     |
| <code>pandas.DataFrame.mean()</code>     | 计算 DataFrame 中每列的平均值                     |
| <code>pandas.DataFrame.median()</code>   | 计算 DataFrame 中每列的中位数                     |
| <code>pandas.DataFrame.min()</code>      | 计算 DataFrame 中每列的最小值                     |
| <code>pandas.DataFrame.mode()</code>     | 计算 DataFrame 中每列的众数                      |
| <code>pandas.DataFrame.quantile()</code> | 计算 DataFrame 中每列的指定分位数值，如四分位数、特定百分位等     |
| <code>pandas.DataFrame.rank()</code>     | 计算 DataFrame 中每列元素的排序排名                  |
| <code>pandas.DataFrame.skew()</code>     | 计算 DataFrame 中列的偏度（三阶矩）                  |
| <code>pandas.DataFrame.sum()</code>      | 计算 DataFrame 中每列元素的总和                    |
| <code>pandas.DataFrame.std()</code>      | 计算 DataFrame 中列的标准差（样本）                  |
| <code>pandas.DataFrame.var()</code>      | 计算 DataFrame 中列的方差（样本）                   |
| <code>pandas.DataFrame.nunique()</code>  | 计算 DataFrame 中每列中的唯一值数量                  |

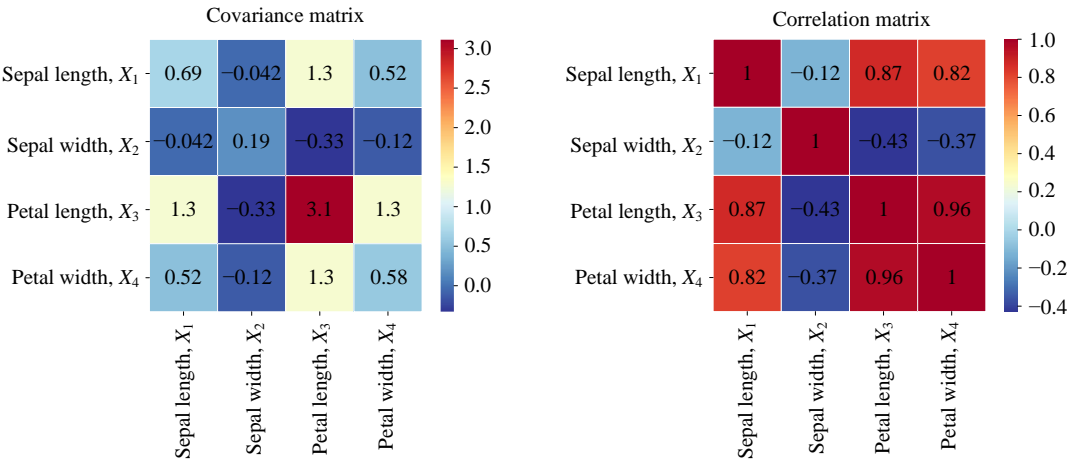


图 3. 鸢尾花数据协方差矩阵、相关性系数矩阵热图

## 22.3 分组聚合：groupby()

在 Pandas 中，`groupby()` 是一种非常实用的数据分组聚合计算方法。`groupby()` 按照某个或多个列的值对数据进行分组，然后对每个分组进行聚合操作。图 4 代码介绍如何使用 `groupby()` 方法，并结合 `mean()`、`std()`、`var()`、`cov()` 和 `corr()` 对分组后的数据进行聚合操作。

图 5、图 6 所示为考虑鸢尾花分类的协方差矩阵、相关性系数矩阵热图。其中，`groupby(['species']).cov()` 得到的数据帧为两层行索引。根据前文介绍的多层行索引数据帧切片方法，`groupby_cov.loc['setosa']` 提取鸢尾花类别为 'setosa' 的协方差矩阵。也可以用 `groupby_cov.xs('setosa')` 提取相同数据。此外，我们也可以用 `iris_df.loc[iris_df['species'] == 'setosa'].cov()` 专门计算鸢尾花类别为 'setosa' 的协方差矩阵。

```
import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 分组计算统计量
a groupby_mean = iris_df.groupby(['species']).mean()
b groupby_std = iris_df.groupby(['species']).std()
c groupby_var = iris_df.groupby(['species']).var()
d groupby_cov = iris_df.groupby(['species']).cov()
e groupby_corr = iris_df.groupby(['species']).corr()
```

图 4. 鸢尾花数据帧 `groupby(['species'])` 计算统计量

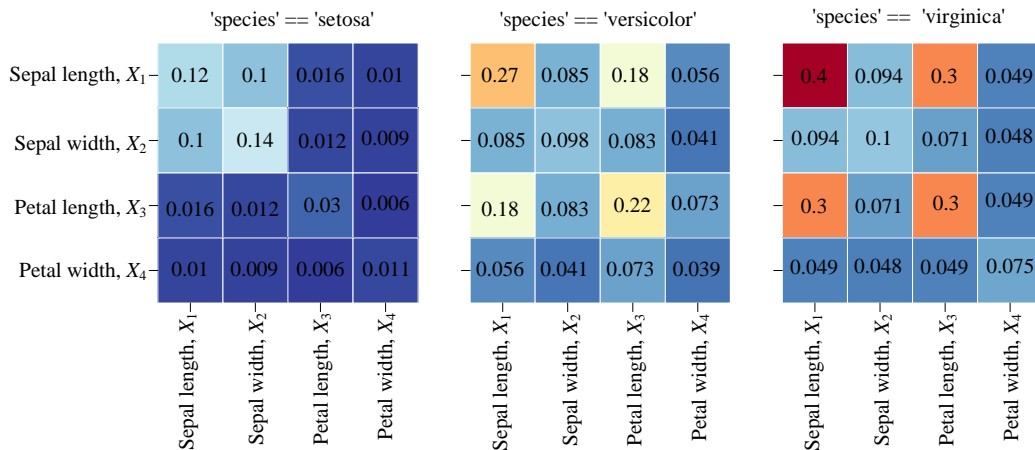


图 5. 协方差矩阵热图，考虑分类

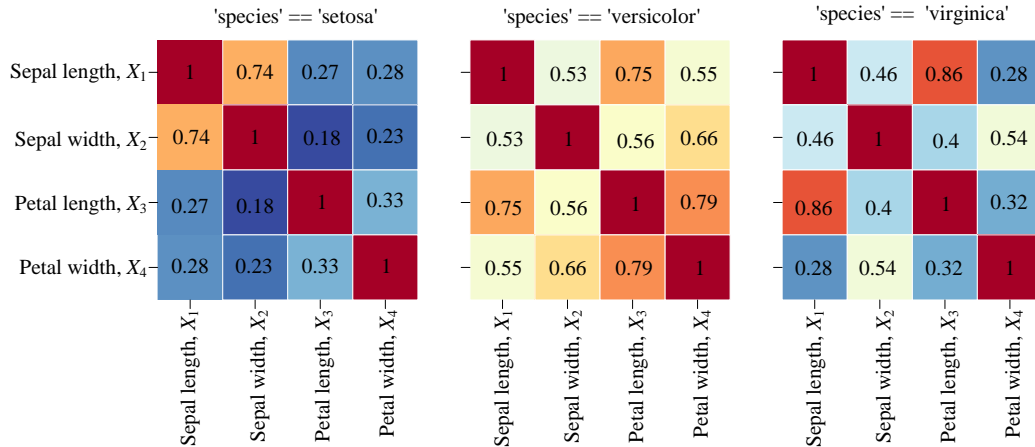


图 6. 相关性系数矩阵热图，考虑分类标签

还是用上一章的例子，给出如何用 `groupby()` 汇总学生成绩。

```
import pandas as pd
import numpy as np

# 创建班级、学号和科目的所有可能组合
classes = ['A', 'B']
stu_ids = [1, 2, 3, 4]
subjects = ['Art', 'Math', 'Science']

# 使用随机数生成成绩数据
np.random.seed(0)
length = len(classes) * len(stu_ids) * len(subjects)
data = np.random.randint(3, 6, size=(length))

# 创建多行标签数据帧
index = pd.MultiIndex.from_product(
    [classes, stu_ids, subjects],
    names=['Class', 'Student ID', 'Subject'])
df = pd.DataFrame(data, index=index, columns=['Score'])

# 1) 每个班级各个科目平均成绩
class_subject_avg = df.groupby(
    ['Class', 'Subject'])['Score'].mean()

# 2) 每个班级各个学生的平均成绩
class_student_avg = df.groupby(
    ['Class', 'Student ID'])['Score'].mean()

# 3) 两个班级放在一起各个科目平均成绩
both_class_avg = df.groupby(
    'Subject')['Score'].mean()

# 4) 两个班级每个同学总成绩，并排名
student_total_score = df.groupby(
    ['Class', 'Student ID'])['Score'].sum().sort_values(
    ascending=False)
```

图 7. 利用 `groupby()` 汇总学生成绩，代码

```
df.groupby(['Class', 'Subject'])['Score'].mean()
```

| Class | Student ID | Subject | Final |
|-------|------------|---------|-------|
| A     | 1          | Art     | 3     |
|       |            | Math    | 4     |
|       |            | Science | 3     |
|       | 2          | Art     | 4     |
|       |            | Math    | 4     |
|       |            | Science | 5     |
|       | 3          | Art     | 3     |
|       |            | Math    | 5     |
|       |            | Science | 3     |
| B     | 4          | Art     | 3     |
|       |            | Math    | 3     |
|       |            | Science | 5     |
|       | 1          | Art     | 4     |
|       |            | Math    | 5     |
|       |            | Science | 5     |
|       | 2          | Art     | 3     |
|       |            | Math    | 4     |
|       |            | Science | 4     |
|       | 3          | Art     | 4     |
|       |            | Math    | 4     |
|       |            | Science | 3     |
|       | 4          | Art     | 4     |
|       |            | Math    | 3     |
|       |            | Science | 3     |



| Class | Subject | Score |
|-------|---------|-------|
| A     | Art     | 3.25  |
|       | Math    | 4.00  |
|       | Science | 4.00  |
| B     | Art     | 4.00  |
|       | Math    | 3.75  |
|       | Science | 3.75  |

```
df.groupby(['Class', 'Student ID'])['Score'].mean()
```



| Class | Student ID | Score |
|-------|------------|-------|
| A     | 1          | 3.33  |
|       | 2          | 4.33  |
|       | 3          | 3.67  |
|       | 4          | 3.67  |
| B     | 1          | 4.67  |
|       | 2          | 3.67  |
|       | 3          | 3.67  |
|       | 4          | 3.33  |

```
df.groupby('Subject')['Score'].mean()
```



| Subject | Score |
|---------|-------|
| Art     | 3.50  |
| Math    | 4.00  |
| Science | 3.88  |

```
df.groupby(['Class', 'Student ID'])['Score'].sum().sort_values(ascending=False)
```



| Class | Student ID | Score |
|-------|------------|-------|
| B     | 1          | 3.33  |
| A     | 2          | 4.33  |
|       | 3          | 3.67  |
|       | 4          | 3.67  |
| B     | 2          | 4.67  |
|       | 3          | 3.67  |
| A     | 1          | 3.67  |
| B     | 4          | 3.33  |

图 8. 利用 groupby() 汇总学生成绩

## 22.4 自定义操作：apply()

在 Pandas 中，可以使用 `apply()` 方法对 `DataFrame` 的行或列进行自定义函数的运算。`apply()` 方法是 Pandas 中最重要和最有用的方法之一，它可以实现 `DataFrame` 数据的处理和转换，也可以实现计算和数据清洗等功能。

如图 9 代码所示，<sup>a</sup> 定义函数 `map_fnc()`，这个函数的目的是将花萼长度 `sepal_length` 转化为等级。转化的规则为，如果 `sepal_length < 5`，等级为 D；如果 `5 <= sepal_length < 6`，等级为 C；如果 `6 <= sepal_length < 7`，等级为 B；其余情况 (`sepal_length > 6`)，等级为 A。<sup>b</sup> 利用 `apply()` 将自定义函数用在数据帧 `iris_df['sepal_length']` 上。



```
import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧

# 定义函数将花萼长度映射为等级
def map_fnc(sepal_length):
    if sepal_length < 5:
        return 'D'
    elif 5 <= sepal_length < 6:
        return 'C'
    elif 6 <= sepal_length < 7:
        return 'B'
    else:
        return 'A'

# 使用 apply 函数将 sepal_length 映射为等级并添加新列
iris_df['ctg'] = iris_df['sepal_length'].apply(map_fnc)
```

图 9. 鸢尾花数据帧使用 `apply()` 自定义函数，对于特定一列

`apply()` 方法可以接受一个函数作为参数，这个函数将会被应用到 `DataFrame` 的每一行或每一列上。这个函数可以是 Pandas 中已经定义好的函数，可以是自定义函数，也可以是匿名 `lambda` 函数。

比如，图 10 代码使用 `apply()` 和 `lambda` 函数计算鸢尾花数据集中每个类别中最小的花瓣宽度。

<sup>a</sup> 等价于 `iris_df.groupby('species')['sepal_length'].min()`。

图 11 中 `apply()` 的输入先是匿名 `lambda` 函数，对象定义为 `row`，代表数据帧的每一行。而 `lambda` 函数调用自定义函数 `map_petal_width()`，这个函数有两个输入。






```
import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧

# 使用apply() 和lambda函数计算每个类别中最小的花瓣宽度
iris_df.groupby('species')['sepal_length'].apply(
    lambda x: x.min())
# iris_df.groupby('species')['sepal_length'].min()
```

图 10. 鸢尾花数据帧使用 apply() 匿名 lambda 函数，对于特定一列



```
import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 计算鸢尾花各类花瓣平均宽度
mean_X2_by_species = iris_df.groupby(
    'species')['petal_width'].mean()

# 定义映射函数
def map_petal_width(petal_width, species):
    if petal_width > mean_X2_by_species[species]:
        return "YES"
    else:
        return "NO"

# 使用 map 方法将花瓣宽度映射为是否超过平均值
iris_df['greater_than_mean'] = iris_df.apply(lambda
    row: map_petal_width(row['petal_width'],
        row['species']), axis=1)
```

图 11. 鸢尾花数据帧使用 apply() 匿名 lambda 函数，对于特定两列

此外，在 Pandas 中，可以使用 map() 方法对 Series 或 DataFrame 特定列进行自定义函数的运算。这个映射关系可以由用户自己定义，也可以使用 Pandas 中已经定义好的函数。

除了 apply() 和 map() 方法之外，Pandas DataFrame 还提供 applymap()、transform() 等方法，请大家自行学习使用。需要大家注意，applymap() 用于对 DataFrame 中的每个元素应用同一个函数，返回一个新的 DataFrame。



有关数据帧分组聚合操作，请大家继续阅读：

[https://pandas.pydata.org/docs/user\\_guide/groupby.html](https://pandas.pydata.org/docs/user_guide/groupby.html)