【机器学习实战】科学处理鸢尾花数据集

原创 AI阿聪 2020-04-10 16:02:39 883 收藏 8 原力计划 版权

分类专栏： 机器学习 文章标签： 机器学习 鸢尾花

目录

一、案例分析

假设我们要创建一个智能手机应用程序，从智能手机拍摄的照片中自动识别花的种类。 我们需创建一个演示机器学习模型，测量花的萼片长度 (sepal length)，萼片宽度 (sepal width)，花瓣长度 (petal length) 和花瓣宽度 (petal width) 四个变量，并根据这些测量识别物种。

如图，花的萼片和花瓣。（萼片是花的最外一环）

三种类型的鸢尾花，如图所示：

根据当地研究人员测量的每种鸢尾花的四个数据 (萼片长/宽和花瓣长/宽)，我们最终目的是想正确的分类这三种花。

二、数据处理

2.1 回答问题

任何数据分析项目的第一步就是提出想要解决的问题，并为成功解决该问题而定义一个度量 。一些常见的问题如下：

（1）在查看数据之前是否了解数据分析问题的类型，是回归，分类还是聚类问题？(明晰问题本质)

这是个根据萼片长度，萼片宽度，花瓣长度和花瓣宽度四个测量指标的**分类**问题

（2）是否在一开始就定义了成功的度量？(设定量化指标)

因为是分类问题，所以可以使用**查准率**，即正确分类花的百分比，来量化模型的表现。我们的数据主管告诉我们应该实现 90% 的准确性。

（3）现有数据是否解决分类问题？(了解数据局限性)

我们目前的数据集只有三种类型的鸢尾花。从这个数据集建立的模型将只适用于那些鸢尾花，未来创建一个通用的花分类器**需要更多的数据**。

（注意：思考这些问题执行有效数据分析的重要一步，不可忽略哦。）

2.2 检查数据

在花费太多时间分析数据之前，提早检查并修正这些数据错误能节省大量时间。一般来说，我们希望回答以下问题：

1. 数据格式有什么问题吗?
2. 数据数值有什么问题吗?
3. 数据需要修复或删除吗?

首先引进 python 里面的几个包，numpy 是为了做数学运算，pandas 是为了处理数据，matplotlib 是为了画图，seaborn 是为了画高级图。代码如下：

```
import numpy as np     # 用来做数学运算
import pandas as pd    # 用来处理数据表
import seaborn as sns   # 用来画高级统计图

# 将所有图都在 Notebook 里显示
%matplotlib inline
import matplotlib.pyplot as plt  # 用来画图

from sklearn.model_selection import train_test_split     # 做交叉验证，划分训练集和测试集
from sklearn.tree import DecisionTreeClassifier           # 用决策树来分类
```

检查点 1. 数据格式 (format)

首先用 pandas 读取 csv 文件并将数据存成数据表 (data frame) 格式。

```
iris_data = pd.read_csv('data/iris-data.csv', na_values=['NA'])   # 从名为iris_data的csv文件读数据存成数据表
#第二个参数用来把 csv 里面空白处用 NaN 代替
iris_data.head(5).append(iris_data.tail())    # 展示数据表前5个和后5个数据
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	cl
0	5.1	3.5	1.4	0.2	Iri
1	4.9	3.0	1.4	0.2	Iri
2	4.7	3.2	1.3	0.2	Iri
3	4.6	3.1	1.5	0.2	Iri
4	5.0	3.6	1.4	0.2	Iri
145	6.7	3.0	5.2	2.3	Iri
146	6.3	2.5	5.0	2.3	Iri
147	6.5	3.0	5.2	2.0	Iri
148	6.2	3.4	5.4	2.3	Iri
149	5.9	3.0	5.1	1.8	Iri

检查点 2. 数据统计 (statistics)

接下来，检查数据的分布可以识别异常值。我们从数据集的汇总统计数据开始。

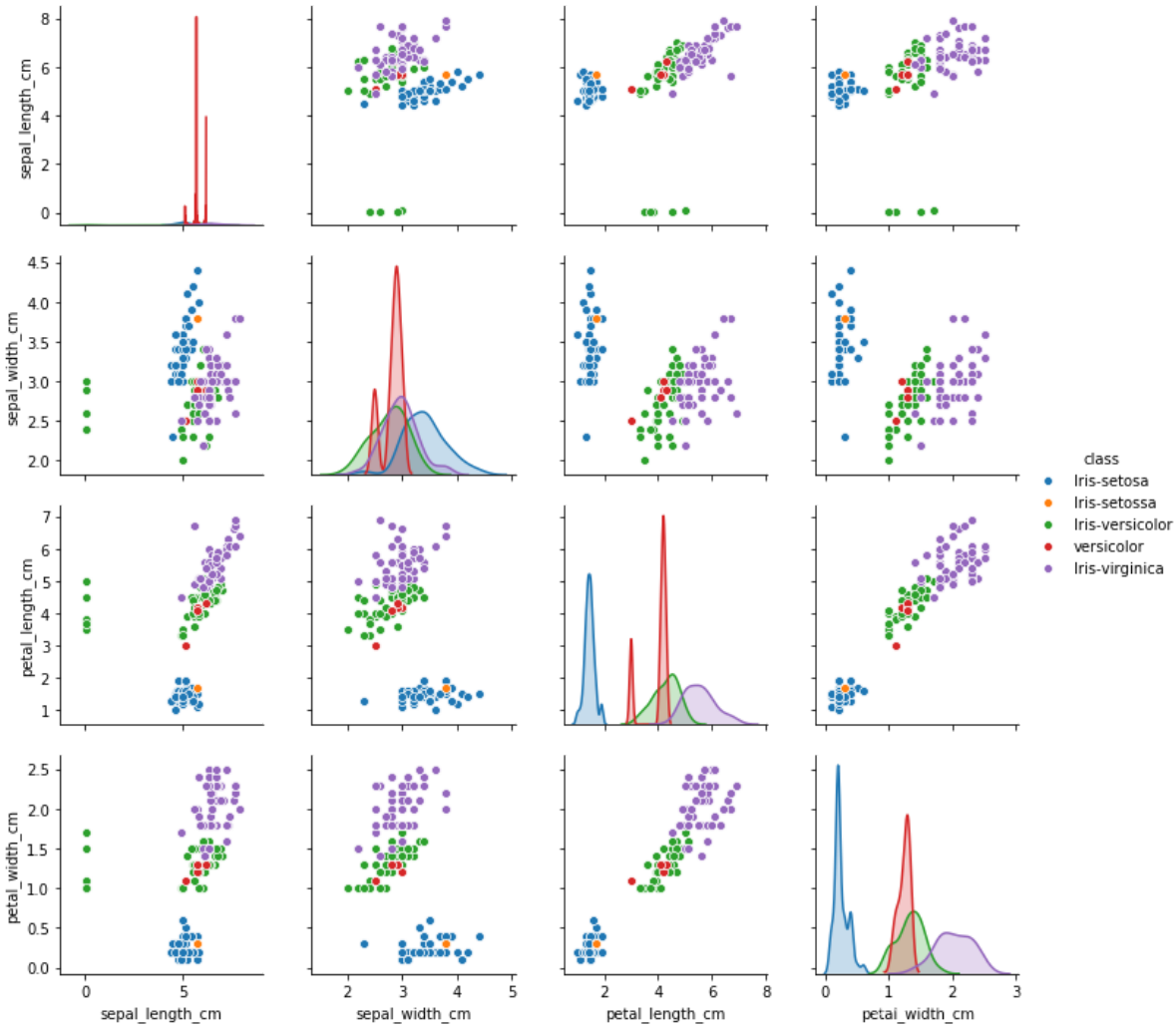
```
iris_data.describe()    # 检查四列数据的个数，平均数，标准差，最小值，最大值和25，50，75的百分位数
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cr
count	150.000000	150.000000	150.000000	145.000000
mean	5.644627	3.054667	3.758667	1.236552
std	1.312781	0.433123	1.764420	0.755058
min	0.055000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.400000
50%	5.700000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

从该表中看到几个有用的值。 例如，我们看到缺少 5 条花瓣宽度的数据（表里 count 那一行的萼片长度，萼片宽度和花瓣长度的个数都是 150 个，唯独花瓣宽度是 145 个）。此外，这样的表给不了太多有用信息，除非我们知道数据应该在一个特定的范围 （如萼片长度的最小值是 0.055，和它其他指标如均值和几个百分位数都不是一个数量级的，很有可能是测量错误）。

比起一串枯燥的数值，我们可能更喜欢绚烂的绘图。接下来可视化数据，它能使异常值立即脱颖而出。

```
sns.pairplot(iris_data.dropna(), hue='class')    # 画散点矩阵图
```



- 第一个参数 iris_data.dropna() 就是除去 NaN 的数据表，这么做原因很简单，图里不可能显示的出 NaN 值的；
- 第二个参数 hue = 'class' 就是根据类 (class) 下不同的值赋予不同的颜色 (hue 就是色彩的意思) 。

散点矩阵图绘制前四列变量（萼片长/宽和花瓣长/宽）的相关系数图，而且用不同颜色区分不同的类下面的这四个变量。 从上图可知，横轴纵轴都有四个变量，那么总共可以画出 16 (4*4) 张小图。

- 对角线上的 4 张都是某个变量和自己本身的关系，由于自己和自己的相关系数永远是 1，画出相关系数图意义不大。
- 非对角线的 12 张就是某个变量和另一个变量的关系。比如第一行第二列的图描述的就是萼片长度 (看纵轴第一个 sepal_length_cm 字样) 和萼片宽度 (看横轴第二个 sepal_width_cm 字样)。

从散点矩阵图中，我们可以迅速看出数据集的一些问题：

（1）图的右侧标注这五个类 (Iris-setosa, Iris-setossa, Iris-versicolor, versicolor, Iris-virginica)，但原本要分类的花只有三类 (Iris-setosa, Iris-versicolor, Iris-virginica)。这意味着在记录数据时可能会犯下一些错误。

（2）在测量中有一些明显的异常值可能是错误的。

- 例如第一行后三张小图，对于 Iris-setosa (山鸢尾花，蓝点)，一个萼片宽度值落在其正常范围之外；
- 例如第二行第一，三，四张小图，对于 Iris-versicolor (变色鸢尾花，红点) ，几个萼片长度值都接近零。

下一步我们的任务是要处理错误的数据。

2.3 清理数据

修正 1. 数据类（class）

问题：按理应该只有三个类，图中却显示五个。

原因是：标记数据时忘记在 Iris-versicolor 之前添加 Iris-。另一个类 Iris-setossa 他们只是多打了一个 s。让我们使用代码来修复这些错误。

```
iris_data['class'].unique()    # 查看数据表的类的不重复值，有5个，但按理说只有3个
```

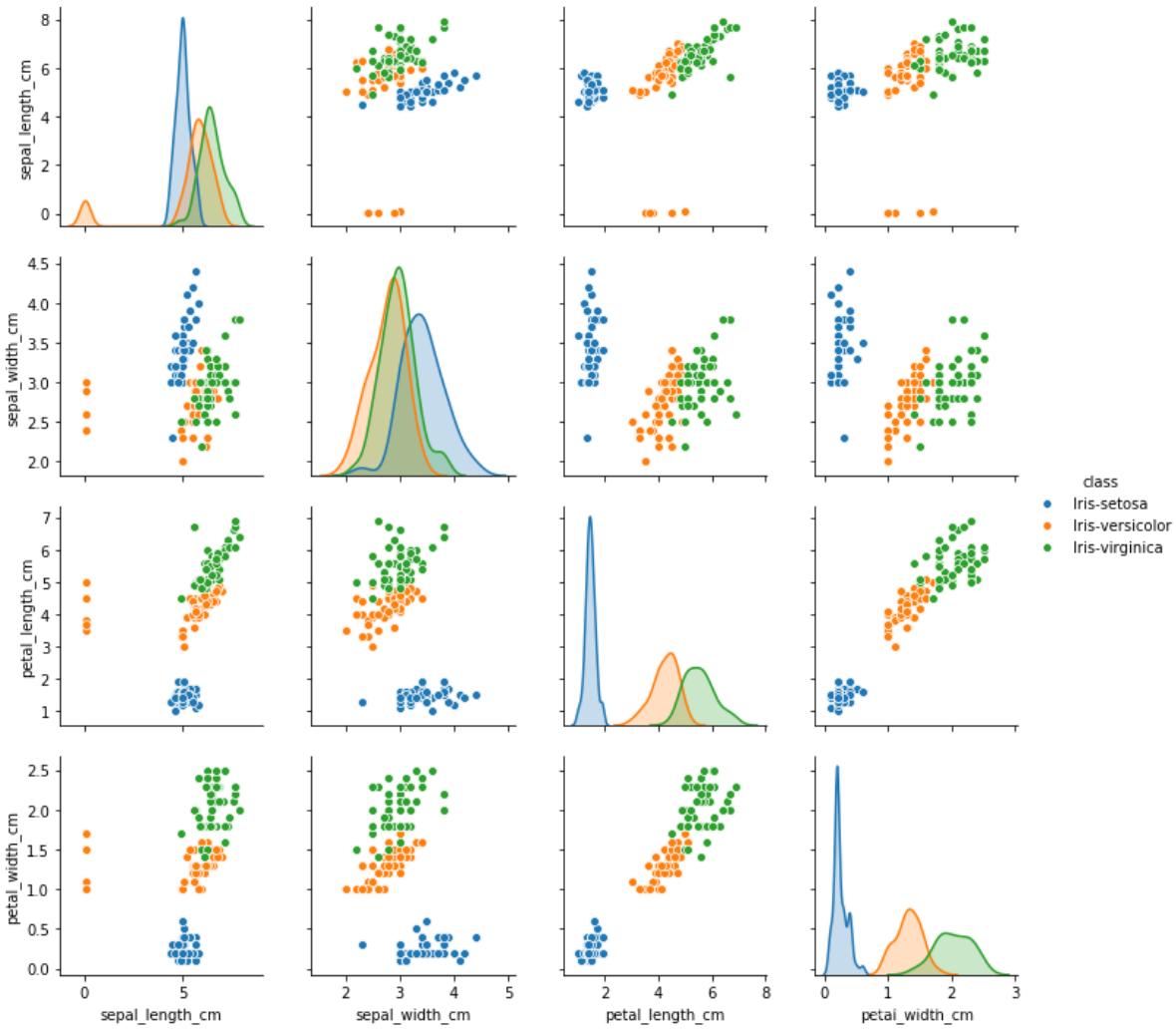
```
array(['Iris-setosa', 'Iris-setossa', 'Iris-versicolor', 'versicolor',
       'Iris-virginica'], dtype=object)
```

```
iris_data.loc[iris_data['class'] == 'versicolor', 'class'] = 'Iris-versicolor'   # 将 versicolor 改为 Iris-versic
iris_data.loc[iris_data['class'] == 'Iris-setossa', 'class'] = 'Iris-setosa'     # 将 Iris-setossa 改为 Iris-setc

iris_data['class'].unique()    # 再查看数据表的类的不重复值，现在只有3个
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

更新后的散点矩阵图如下：

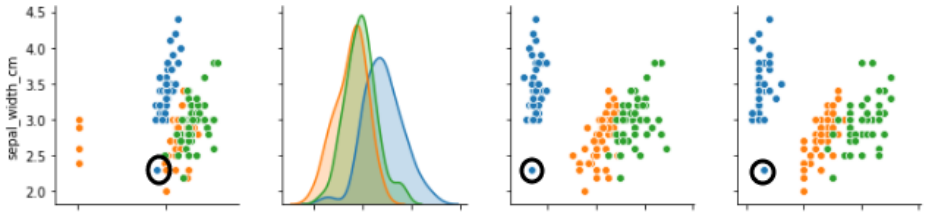


现在只有三个类而分别是 Iris-setosa, Iris-versicolor 和 Iris-virginica。

修正点 2. 异常数值 (outliers)

修复异常值是一件棘手的事情。因为我们很难判断异常值是否由测量误差引起，或者是不正确的单位记录数据，或者是真正的异常。如果我们决定排除任何数据，需要记录排除的数据并提供排除该数据的充分理由。由上节所知，我们有两种类型的异常值。

问题：山鸢尾花的一个萼片宽度值落在其正常范围之外 (黑色圆框)。

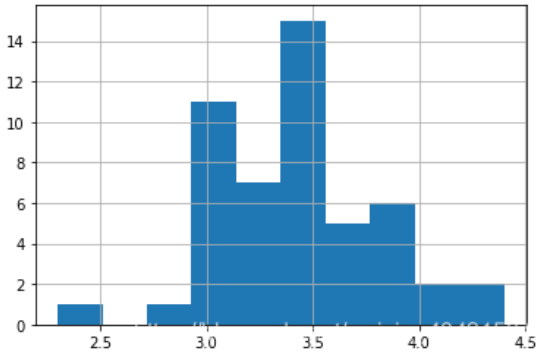


我们发现，山鸢尾花 (Iris-setosa) 的萼片宽度 (sepal_width_cm) 不可能低于 2.5 厘米。显然，这个记录是错误的，这种情况下最有效的方法是删除它而不是花时间查找原因。但是，我们仍需要知道有多少个类似这样的错误数据，如果很少删除它没有问题，如果很多我们需要查明原因。

```
# 查看 Iris-setosa 里萼片宽度小于2.5厘米的数据，只有一个
iris_data.loc[(iris_data['class'] == 'Iris-setosa') & (iris_data['sepal_width_cm'] < 2.5)]
```

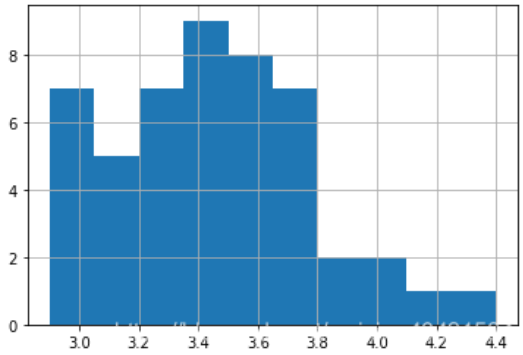
	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
41	4.5	2.3	1.3	0.3

```
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'sepal_width_cm'].hist()
```



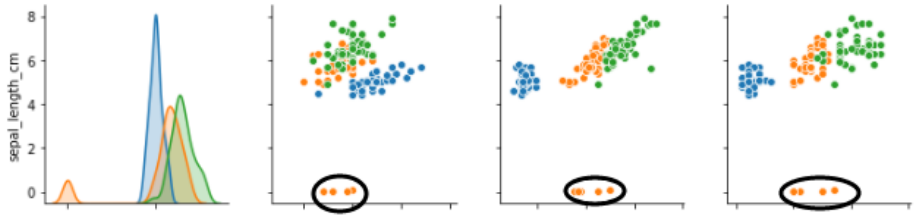
第一行代码是用数据表里的 loc[] 函数来找到类值为 Iris-setoa (因为是蓝点) 并且 sepal width 小于 2.5 的所有行。最后发现只有一个这样的数据，而上图的条形图也确认了这样的异常值只有一个。因此可以直接删除此数据。

```
# 去掉 Iris-setosa 里萼片宽度大于2.5厘米的数据，然后画出其条形图
iris_data = iris_data.loc[(iris_data['class'] != 'Iris-setosa') | (iris_data['sepal_width_cm'] >= 2.5)]
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'sepal_width_cm'].hist()
```



第一行代码将类值为 Iris-setosa 并且 sepal width 大于 2.5 的所有数据都新存到 iris_data 中。从上面条形图也看到了再没有这个异常值。现在所有的山鸢尾花的萼片宽度都大于 2.5 厘米。

问题：变色鸢尾花的几个萼片长度值接近与零 (黑色椭圆框)。

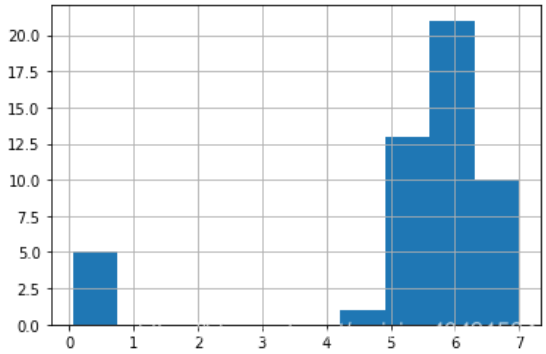


我们发现，所有这些接近零的 sepal_length_cm 似乎错位了两个数量级。

```
# 查看 Iris-versicolor 里的萼片长度接近于零的所有数据
iris_data.loc[(iris_data['class'] == 'Iris-versicolor') & (iris_data['sepal_length_cm'] < 1.0)]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	clas
77	0.067	3.0	5.0	1.7	Iris-v
78	0.060	2.9	4.5	1.5	Iris-v
79	0.057	2.6	3.5	1.0	Iris-v
80	0.055	2.4	3.8	1.1	Iris-v
81	0.055	2.4	3.7	1.0	Iris-v

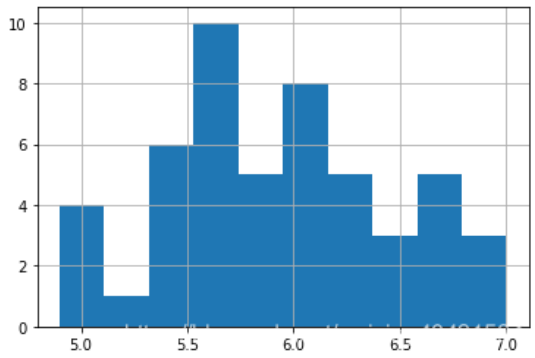
```
# 画出其条形图
iris_data.loc[iris_data['class'] == 'Iris-versicolor', 'sepal_length_cm'].hist()
```



第一行代码是用数据表里的 loc[] 函数来找到类值为 Iris-versicolor (因为是红点) 并且 sepal length 接近零的所有行，发现有五个数据，而条形图最左边显示的数据个数也确认了是五个。

```
# 将萼片长度乘以100倍，从单位米换成单位厘米
iris_data.loc[(iris_data['class'] == 'Iris-versicolor') &
              (iris_data['sepal_length_cm'] < 1.0),
              'sepal_length_cm'] *= 100.0

iris_data.loc[iris_data['class'] == 'Iris-versicolor', 'sepal_length_cm'].hist()
```



修正点 3. 缺失数值 (missing value)

我们还有些 NaN 数据。通常我们有两种方式来处理这类数据。

1. 删除 (deletion)
2. 插补 (imputation)

在本例中删除不是理想的做法，特别是考虑到它们都在 Iris-setosa 下，如图

5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	NaN	Iris-setosa
8	4.4	2.9	1.4	NaN	Iris-setosa
9	4.9	3.1	1.5	NaN	Iris-setosa
10	5.4	3.7	1.5	NaN	Iris-setosa
11	4.8	3.4	1.6	NaN	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	5.7	3.0	1.1	0.1	Iris-setosa

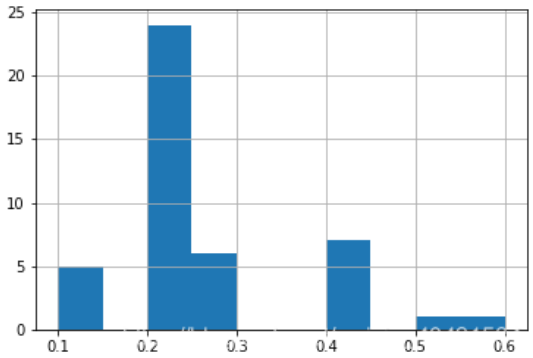
所有缺失的值都属于 Iris-setosa类，直接删除可能会对日后数据分析带来偏差。此外，可以用插补方法，其最常见的方法平均插补 (mean imputation)。其做法就是“假设知道测量的值落在一定范围内，就可以用该测量的平均值填充空值”。

```
# 查看所有有NaN值的行数据，发现只有花瓣宽度 (petal_width_cm) 列下才有
iris_data.loc[(iris_data['sepal_length_cm'].isnull()) |
              (iris_data['sepal_width_cm'].isnull()) |
              (iris_data['petal_length_cm'].isnull()) |
              (iris_data['petal_width_cm'].isnull())]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
7	5.0	3.4	1.5	NaN
8	4.4	2.9	1.4	NaN

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
9	4.9	3.1	1.5	NaN
10	5.4	3.7	1.5	NaN
11	4.8	3.4	1.6	NaN

```
# 画出其条形图
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'petal_width_cm'].hist()
```



接下来用 hist() 函数画出 Iris-setosa 花瓣宽度的条形图，可以清楚看到大多数宽度在 0.25 左右。

```
# 用平均值来代替NaN值
average_petal_width = iris_data.loc[iris_data['class'] == 'Iris-setosa', 'petal_width_cm'].mean()

iris_data.loc[(iris_data['class'] == 'Iris-setosa') &
              (iris_data['petal_width_cm'].isnull()),
              'petal_width_cm'] = average_petal_width

iris_data.loc[(iris_data['class'] == 'Iris-setosa') &
              (iris_data['petal_width_cm'] == average_petal_width)]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
7	5.0	3.4	1.5	0.25
8	4.4	2.9	1.4	0.25
9	4.9	3.1	1.5	0.25
10	5.4	3.7	1.5	0.25
11	4.8	3.4	1.6	0.25

然后用 mean() 准确求出其宽度的平均值，将其 NaN 值全部用平均值代替，最后打出那 5 行插补后的数据表。

```
# 确保所有NaN值都已被更新
iris_data.loc[(iris_data['sepal_length_cm'].isnull()) |
              (iris_data['sepal_width_cm'].isnull()) |
              (iris_data['petal_length_cm'].isnull()) |
              (iris_data['petal_width_cm'].isnull())]
```

为了确保所有 NaN 值已被替换，再次用 iris_data[A].isnull() 语句来查看，出来的结果是一个只有列标题的空数据表。这表示表内已经没有 NaN 值了。

2.4 测试数据

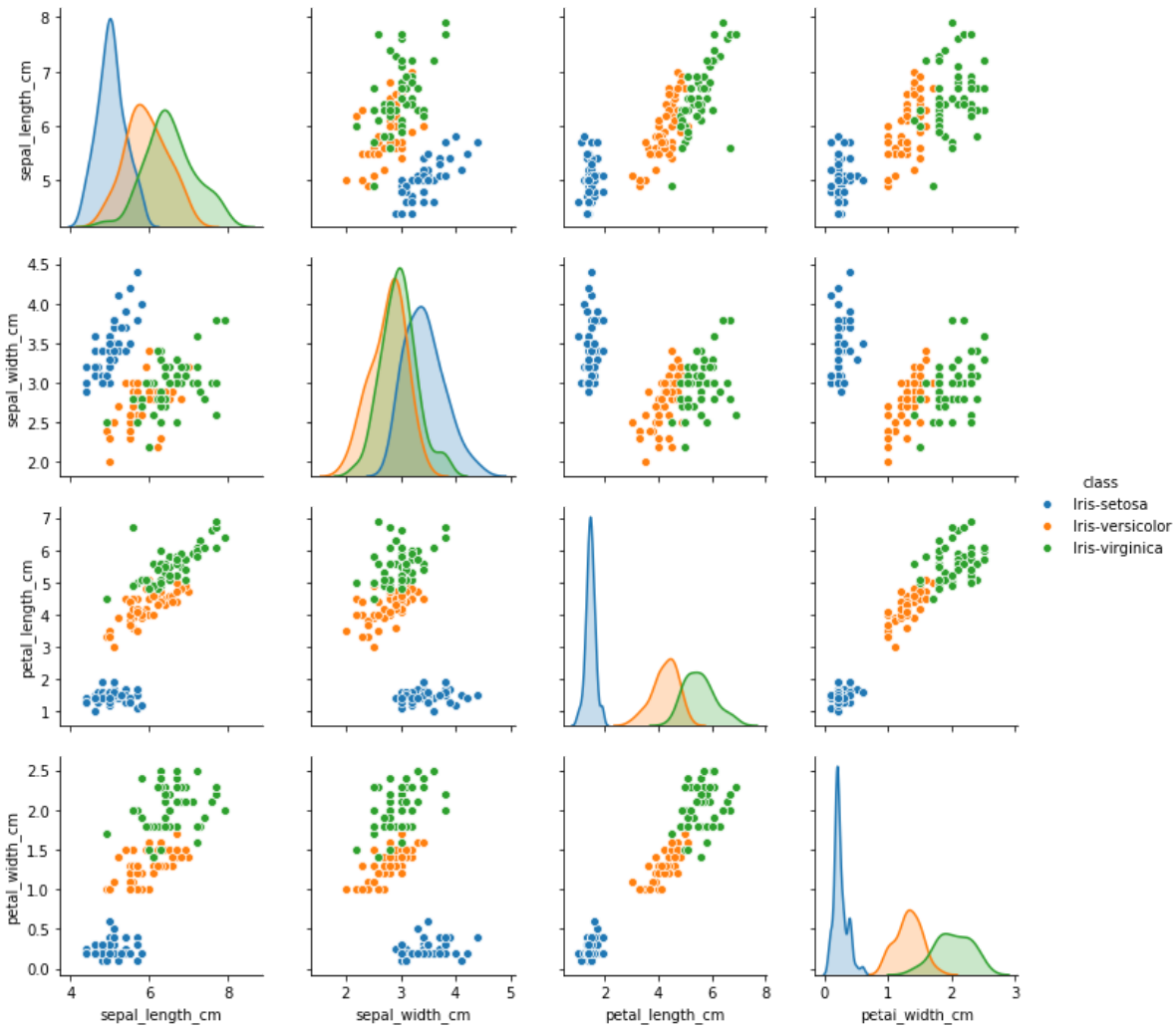
1. 存储数据（save data）

```
iris_data.to_csv('data/iris-data-clean.csv', index = False)    # 将整理好的数据写到一个名叫iris-data-clean的csv文件里,

iris_data_clean = pd.read_csv('data/iris-data-clean.csv')      # 重新从iris-data-clean的csv文件里读数据存成iris_data
```

让我们再看看基于干净数据画的散点矩阵图吧。

```
sns.pairplot(iris_data_clean, hue='class')  # 查看散点矩阵图
```



从上图可看到：

1. 五个类变成三个类；
2. 异常值全部删除或修正了。

2. 声明数据（assert data）

为了防止一些数据问题没有解决，我们可以用 assert 语句来做声明。该语句好处是，在运行时如果声明语句为真，没有任何事发生，反之会报错而警告我们有哪些错误数据需要注意且修正。

```
# 声明花只有三种类型
assert len(iris_data_clean['class'].unique()) == 3

# 声明变色鸢尾花的萼片长度应该大于 2.5 厘米
assert iris_data_clean.loc[iris_data_clean['class'] == 'Iris-versicolor', 'sepal_length_cm'].min() >= 2.5

# 数据不应该有缺失
assert len(iris_data_clean.loc[(iris_data_clean['sepal_length_cm'].isnull()) |
                               (iris_data_clean['sepal_width_cm'].isnull()) |
                               (iris_data_clean['petal_length_cm'].isnull()) |
                               (iris_data_clean['petal_width_cm'].isnull())]) == 0
```

如果任何声明被违反，我们应该立即停止分析，而回到整理阶段。

三、用 scikit-learn 来预测数据

3.1 选出特征 (输入变量) 和标记 (输出变量)

```
# 读数据
iris_data_clean = pd.read_csv('data/iris-data-clean.csv')

# 选出特征
all_inputs = iris_data_clean[['sepal_length_cm', 'sepal_width_cm',
                              'petal_length_cm', 'petal_width_cm']].values

# 选出标记
all_classes = iris_data_clean['class'].values
```

3.2 划分训练集和测试集

```
(training_inputs, test_inputs, training_classes,
 test_classes) = train_test_split(all_inputs, all_classes, train_size=0.75, random_state=1)
```

用 75% 和 25% 的比例来划分训练集和测试集，设一个 random_state 是为了在机器学习中每次出的结果一样，便于找问题。

3.3 用模型来学习

```
# 创建决策树分类器
decision_tree_classifier = DecisionTreeClassifier()

# 训练数据
decision_tree_classifier.fit(training_inputs, training_classes)

# 分类精度
decision_tree_classifier.score(test_inputs, test_classes)
```

```
0.9736842105263158
```

再一次感叹 scikit-learn 的强大和好用。

四、思考题

这时候有人会问了：

1. 这个训练集和测试集划分是随机的，一次不足以说明你查准率高；
2. 这数据太少没代表性，换套数据模型可能过拟合；
3. 特征选择有很大学问，凭什么就选萼片长度，萼片宽度，花瓣长度和花瓣宽度这四个变量？

有待学习，思考哦。