```
In [2]: # FIRST APPLICATION
      # 在本节中,将通过一个简单的机器学习应用,创建第一个模型
      # 假设一位业余植物学家对区分她发现的一些鸢尾花的物种感兴趣
      # 她收集了每朵鸢尾花的相关测量数据
      #包括花瓣的长度和宽度,以及花萼的长度和宽度,单位为厘米
      # 物种有山鸢尾 (Setosa)、变色鸢尾 (Versicolor) 和维吉尼亚鸢尾 (Virginica)
      # 通过这些数据,她可以确定每朵鸢尾花所属的物种。
      # 目标是建立一个机器学习模型,该模型从已知的鸢尾花测量数据中学习
      # 从而预测新鸢尾花的物种
      # 因为拥有已知物种的鸢尾花测量数据,这属于一个"监督学习"问题
      # 在这个问题中,我们希望预测鸢尾花的物种,所以这是一个分类问题的示例
      # 可能的输出结果(不同种类的鸢尾花)被称为"类别"
      #数据集中每朵鸢尾花都属于三个类别之一,因此这是一个"三分类问题"
      # 对于单个数据点(即一朵鸢尾花),期望的输出是这朵花的物种
      # 对于特定的数据点,它所属的物种被称为"标签"
      # 本例中使用的数据是鸢尾花数据集(Iris dataset)
      # 该数据集包含在 scikit-learn 的datasets模块中,可以通过调用Load_iris函数来加载它
      import mglearn
      import numpy as np
      from sklearn.datasets import load_iris
      iris_dataset = load_iris()
      # 数据集包含了哪些信息
      print("\nKey of iris_dataset:\n{}".format(iris_dataset.keys()))
      # DESCR数据集的描述信息
      print(iris dataset['DESCR'][:193]+"\n...")
      #数据集包含了3个品种,也就是分类
      print("Targe Name:{}".format(iris_dataset['target_names']))
      #数据集的4个特征列表
      print("Feature Names:{}".format(iris_dataset['feature_names']))
      # data字段包含花萼长度、花萼宽度、花瓣长度和花瓣宽度的数值测量
      # 以NumPy数组的形式呈现
      print("\nType of data:\n{}".format(iris_dataset['data']))
      # 该数组形式包含150种不同花朵的4种测量数据。
      print("\nShape of data:\n{}".format(iris dataset['data'].shape))
      # 前五个样本的特征值
      print("\nFirst five column of data:\n{}".format(iris dataset['data'][0:5]))
      # target数组包含了每朵被测量花朵的物种,同样以NumPy数组的形式呈现
      print("Type of target:{}".format(type(iris dataset['target'])))
      # target是一个一维数组,每个条目对应一朵花
      print("Shape of target:{}".format(iris_dataset['target'].shape))
      # 这些数字的含义由iris['target_names']数组给出:
      # 0 表示山鸢尾, 1 表示变色鸢尾, 2 表示维吉尼亚鸢尾
      print("Target:\n{}".format(iris dataset['target']))
      # 为了评估模型的性能,需要向它展示新数据(模型之前未见过的数据)
      # 通过将收集的有标签数据(150个花的测量值)分成两部分来完成
      #一部分数据用于构建机器学习模型,称为训练数据或训练集
      # 其余的数据则用于评估模型的表现, 称为测试数据、测试集或保留集
      # scikit-learn提供了一个函数,可以打乱数据集并将其分割: train test split函数
      # 该函数提取数据中的75%行作为训练集
      # 剩余的25%数据和相应的标签被声明为测试集,使用25%的测试集是一条不错的经验法则
      # 在scikit-learn中,数据通常用大写的X表示,而标签则用小写的y表示
      #大写的X数据是一个二维数组(矩阵),而小写的y则是是一个一维数组(向量)
      from sklearn.model selection import train test split
      X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'],
                                            iris dataset['target'],
                                            random_state=0)
```

```
print("X_train shape:\n{}".format(X_train.shape))
print("y_train shape:\n{}".format(y_train.shape))
print("X_test shape:\n{}".format(X_test.shape))
print("y_test shape:\n{}".format(y_test.shape))
import pandas as pd
# 在 jupyter notebook 中运行可查看绘图
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
grr = pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),
                          marker='o',
                          hist kwds={'bins': 20},
                          s=60, alpha=.8, cmap=mglearn.cm3)
# 使用 k-近邻分类器
# k-近邻中的 k 表示,不仅可以使用最近的邻居进行预测
# 还可以考虑训练集中任意固定数量 k 的邻居(例如,最近的三个或五个邻居)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n neighbors=1)
# 训练模型
knn.fit(X_train, y_train)
# 假设在野外发现了一朵鸢尾花,
# 其花萼长度为5厘米, 花萼宽度为2.9厘米, 花瓣长度为1厘米, 花瓣宽度为0.2厘米。
X_{\text{new}} = \text{np.array}([[5, 2.9, 1, 0.2]])
print("X_new.shaple:{}".format(X_new.shape))
# 通过模型预测类别
prediction = knn.predict(X_new)
print("Prediction:{}".format(prediction))
print("Predicted target name:{}".format(
   iris_dataset['target_names'][prediction]))
# 评估模型可信度
# 对测试集进行预测,并与其标签(已知的物种)进行比较
y pred = knn.predict(X test)
print("Test set predictions:\n {}".format(y_pred))
# 计算预测的精度, 其中y pred==y test得到的是一个bool向量, np.mean返回True的比例
print("Test set score: {:.2f}".format(np.mean(y_pred == y_test)))
# 也可以直接用模型的score方法完成预测+精度计算
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
# 构建模型时使用了一个包含正确的物种信息数据集,使其成为一个监督学习任务。
# 三分类问题: 山鸢尾(0)、变色鸢尾(1)或维吉尼卡鸢尾(2)
# 可能的物种称为类别,而单个鸢尾花的物种称为其标签
# 鸢尾花数据集由两个 NumPy 数组组成:
#一个包含数据,在scikit-Learn中称为X
# 数组X是一个二维特征数组,每个数据点一行,每个特征一列
# 一个包含正确或期望的输出, 称为 y
# 数组y是一个一维数组,每个样本包含一个类别标签,即一个范围为@到2的整数
#数据集分为训练集,用来构建模型
# 以及测试集,用来评估我们的模型在未见过的数据上的泛化能力
# k-近邻分类算法,通过考虑训练集中最近的邻居来对新数据点进行预测
# 这一算法在KNeighborsClassifier类中实现
# 通过调用fit来构建模型,将训练数据(X train)和训练输出(y train)作为参数传入
#使用score评估模型,计算模型的准确率。
```

```
Key of iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
.. _iris_dataset:
Iris plants dataset
_____
**Data Set Characteristics:**
:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive
Targe Name:['setosa' 'versicolor' 'virginica']
Feature Names:['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'pet
al width (cm)']
Type of data:
[[5.1 3.5 1.4 0.2]
[4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
[4.6 3.1 1.5 0.2]
[5. 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5. 3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3. 1.4 0.1]
 [4.3 3. 1.1 0.1]
 [5.8 4. 1.2 0.2]
[5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1. 0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5. 3. 1.6 0.2]
 [5. 3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5. 3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3. 1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5. 3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
```

- [4.4 3.2 1.3 0.2]
- [5. 3.5 1.6 0.6]
- [5.1 3.8 1.9 0.4]
- [4.8 3. 1.4 0.3]
- [5.1 3.8 1.6 0.2]
- [4.6 3.2 1.4 0.2]
- [5.3 3.7 1.5 0.2]
- [5. 3.3 1.4 0.2]
- [7. 3.2 4.7 1.4]
- [6.4 3.2 4.5 1.5]
- [6.9 3.1 4.9 1.5]
- [5.5 2.3 4. 1.3]
- [6.5 2.8 4.6 1.5]
- [5.7 2.8 4.5 1.3]
- [6.3 3.3 4.7 1.6]
- [4.9 2.4 3.3 1.]
- [6.6 2.9 4.6 1.3]
- [5.2 2.7 3.9 1.4]
- [5. 2. 3.5 1.]
- [5.9 3. 4.2 1.5]
- [6. 2.2 4. 1.]
- [6.1 2.9 4.7 1.4]
- [5.6 2.9 3.6 1.3]
- [6.7 3.1 4.4 1.4]
- [5.6 3. 4.5 1.5]
- [5.8 2.7 4.1 1.]
- [6.2 2.2 4.5 1.5]
- [5.6 2.5 3.9 1.1]
- [5.9 3.2 4.8 1.8]
- [6.1 2.8 4. 1.3]
- [6.3 2.5 4.9 1.5]
- [6.1 2.8 4.7 1.2]
- [6.4 2.9 4.3 1.3]
- [6.6 3. 4.4 1.4]
- [6.8 2.8 4.8 1.4]
- [6.7 3. 5. 1.7] [6. 2.9 4.5 1.5]
- [5.7 2.6 3.5 1.]
- [5.5 2.4 3.8 1.1]
- [5.5 2.4 3.7 1.]
- [5.8 2.7 3.9 1.2] [6. 2.7 5.1 1.6]
- [5.4 3. 4.5 1.5]
- [6. 3.4 4.5 1.6]
- [6.7 3.1 4.7 1.5]
- [6.3 2.3 4.4 1.3]
- [5.6 3. 4.1 1.3]
- [5.5 2.5 4. 1.3]
- [5.5 2.6 4.4 1.2]
- [6.1 3. 4.6 1.4]
- [5.8 2.6 4. 1.2]
- [5. 2.3 3.3 1.]
- [5.6 2.7 4.2 1.3]
- [5.7 3. 4.2 1.2]
- [5.7 2.9 4.2 1.3]
- [6.2 2.9 4.3 1.3] [5.1 2.5 3. 1.1]
- [5.7 2.8 4.1 1.3]
- [6.3 3.3 6. 2.5]
- [5.8 2.7 5.1 1.9]

```
[7.1 3. 5.9 2.1]
 [6.3 2.9 5.6 1.8]
 [6.5 3. 5.8 2.2]
 [7.6 3. 6.6 2.1]
[4.9 2.5 4.5 1.7]
 [7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
[6.8 3. 5.5 2.1]
[5.7 2.5 5. 2.]
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
 [6.5 3. 5.5 1.8]
[7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
[6. 2.2 5. 1.5]
 [6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2.]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6. 1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3. 4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3. 5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3. 6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6. 3. 4.8 1.8]
[6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
[6.7 3. 5.2 2.3]
[6.3 2.5 5. 1.9]
[6.5 3. 5.2 2.]
 [6.2 3.4 5.4 2.3]
[5.9 3. 5.1 1.8]]
Shape of data:
(150, 4)
First five column of data:
[[5.1 3.5 1.4 0.2]
[4.9 3. 1.4 0.2]
[4.7 3.2 1.3 0.2]
[4.6 3.1 1.5 0.2]
[5. 3.6 1.4 0.2]]
Type of target:<class 'numpy.ndarray'>
Shape of target:(150,)
```

```
Target:
2 2]
X_train shape:
(112, 4)
y_train shape:
(112,)
X_test shape:
(38, 4)
y_test shape:
(38,)
X_new.shaple:(1, 4)
Prediction:[0]
Predicted target name:['setosa']
Test set predictions:
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
2]
```



