第2章 机器学习项目开发环境

卿来云
lyqing@ucas.ac.cn
2023秋

Recall： 机器学习

对于某类任务T和性能度量P，如果计算机程序在T上以P衡量的性能随着经验E而自我完善，就称这个计算机程序从经验E学习。

—— Tom Mitchell

经验E: 训练数据$\mathcal{D}$

模型：预测函数f

机器学习算法：从训练数据$\mathcal{D}$得到模型f

性能度量：模型有多好

Recall：机器学习项目的开发过程

$y = f(\boldsymbol{x})$

数据

- 数据收集：给定任务，分析可能的相关特征，收集数据
- 特征探索分析
- 特征工程：将原始数据转换成算法制定的输入

模型训练

- 确定机器学习算法（函数集合）$\mathcal{F}$
- 模型训练：给定超参数情况下，确定模型参数）
- 模型选择：在验证集上评估模型预测性能

性能评估

- 模型测试和评估
- 模型应用

任务

- 例：对鸢尾花进行分类

数据收集

- 初步确定与任务相关的属性/特征
  - 花瓣的宽度、长度、颜色、形状
  - 花萼的宽度、长度、颜色、形状
  - 叶子的宽度、长度、颜色、形状
  - 花的图片
  - ...
- 确定属性收集的可行性
  - 拍摄照片
  - 图像分割 → 形状
  - 采集鸢尾花，用尺子测量各种属性
  - 人工标注花的颜色、形状
  - ...
- 采取行动，收集数据：多个样本的属性、人工标注训练样本

数据示例

- 鸢尾花数据集：$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
  - 150个样本： $N = 15$
  - 每个样本有$D = 4$维特征$\boldsymbol{x}_i$：花瓣的长度和宽度、花萼的长度和宽度
  - 标签y_i：共3类样本

特征

目标/标签列

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

特征值

样本

标签

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \cdots & \cdots & \cdots & \cdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{pmatrix}$$

数据分析和计算工具包：NumPy、 SciPy 、Pandas

- NumPy：Numeric Python：Python的开源数值计算扩展，可用来存储和处理大型矩阵
  - 多维数组（ ndarray ）
  - 实用的线性代数、傅里叶变换和随机数生成函数
- SciPy：建立在NumPy基础上，提供统计、优化和数值微积分计算等功能
  - 稀疏矩阵运算
- Pandas (Panel data structures)： Python语言的“关系数据库”，数据结构和数据分析工具，非常高效且易于使用
  - 统计、分组、排序、透视表（SQL语句的大部分功能）
  - dataframe：二维表

读取数据

```
import pandas as pd
import numpy as np

#读取数据：csv文件没有列名，增加列名
feat_names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', species ']
dpath = "./data/”
df = pd.read_csv(dpath + "iris.csv", names = feat_names)

#通过观察前5行，了解数据每列特征的概况
df.head()

# 数据总体信息
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

数据探索分析

- 数据探索有助于选择合适的数据预处理和机器学习算法。
- 数据质量
  - 缺失值
  - 离群点/噪声点
- 单个特征分布
  - 统计量
  - 直方图
- 特征之间相关性
- 特征与目标之间的相关性

缺失值分析

- 由于各种原因，实际应用中数据总是存在一些缺失值，通常表示为NaN/NaT（日期型变量）。

- 统计行/列缺失率

```
# 计算每一行有多少个缺失值的值，即按行统计缺失值
row_null = df.isnull().sum(axis=1)

# 按列统计缺失值
col_null = df.isnull().sum(axis=0)

#统计整个df的缺失值
all_null = df.isnull().sum().sum()
```

缺失值处理

■缺失值处理
- 删除含有（1个或多个）缺失特征的样本（行）
- 删除缺失值太多的特征（列）
- 对缺失值进行填补，例如均值、中值
- 不处理（有些算法可处理数据缺失情况，如XGBoost）

■ Pandas库的fillna函数可以对缺失值进行填补，灵活，但重用性较弱
- 训练集中的缺失值用训练集的统计量填补
- 测试集中的缺失值也要用训练集的统计量来填补

```
#用列中值填补
medians = df.median()
df = df.fillna(medians)
```

缺失值填补

- Scikit-learn的SimpleImputer类提供一些常见填补方法
  - 均值mean（默认方法）
  - 中位数median
  - 众数most_frequent，可用于非数值数据
  - 指定的常数，用fill_value替换缺失值，可用于非数值数据

```
class SimpleImputer(
                missing_values = nan,
                Strategy = 'mean',
                fill_value = None,
                verbose=0,
                copy=True,
                add_indicator=False )
```

```
In [40]: from sklearn.impute import SimpleImputer
         imputer = SimpleImputer(strategy="mean")
         data = imputer.fit_transform([[1, 2],
                                       [np.nan, 3],
                                       [7, 6]])

In [41]: data

Out[41]: array([[1., 2.],
                [4., 3.],
                [7., 6.]])
```

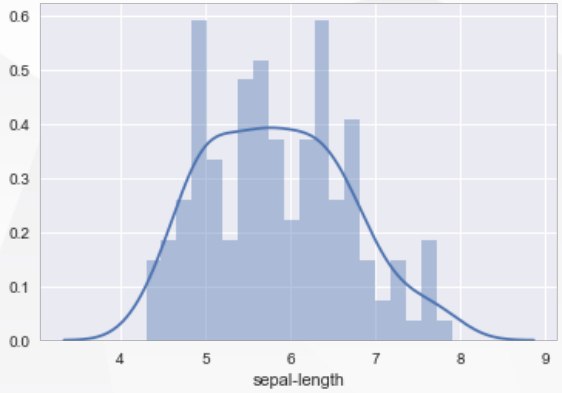
第1列第2行的的缺失值np.nan
被第1列的均值4替代

特征的分布：直方图

- 直方图：每个取值在数据集中出现的次数，可视为概率密度函数的估计
- 核密度估计（Kernel Density Estimation, KDE）：直方图的加窗平滑

- 对连续特征，通常用seaborn工具包的distplot画直方图
- 对离散特征，通常用seaborn工具包的countplot画直方图条形图
  - dataframe的value_counts()得到每个特征取值的样本数目

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.distplot(df['sepal-length'], bins=20, kde=True)
```

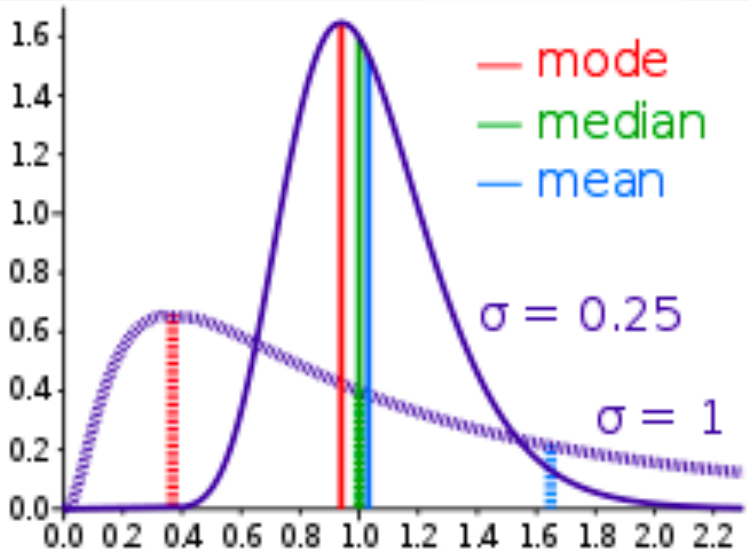


可视化工具包：Matplotlib、 Seaborn

- Matplotlib：2D图形绘制工具
  - 参数较多，复杂灵活，能制作具有更多特色的图
  - http://matplotlib.org/
- Seaborn：基于Matplotlib的可视化工具包
  - 提供更高层次的用户接口，可以给出漂亮的数据统计图
  - https://seaborn.pydata.org/

表示集中趋势的统计量

- 均值
- 中值：在一组排好序数据中
  - 数据数量为奇数，则中值为中间的那个数；
  - 如果数据数量为偶数，则中值为中间的那两个数值的平均值。
- 众数: 出现概率最大的地方

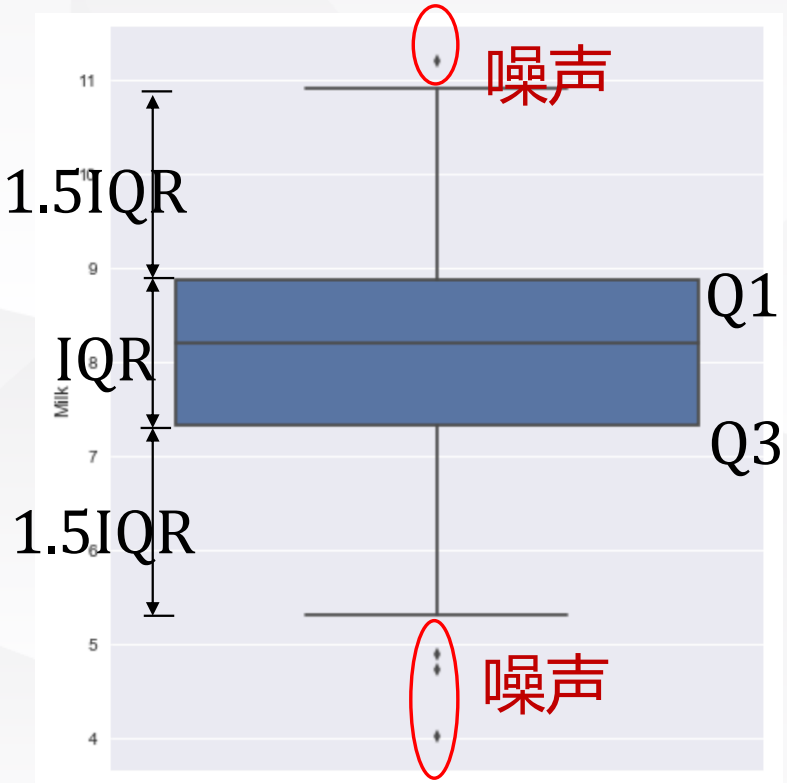


表示散布程度的统计量

- 方差
- 四分位数间距（Interquartile Range, IQR）：25%分位数到75%分位数之间的区间的宽度

- seaborn.boxplot函数
  - 长方形为IQR
  - 中间的线为中值
  - 两头的虚线：1.5 IQR

  - 可识别噪声点



Pandas支持的统计量

- 对数值型特征，describe方法可以特征的基本统计学特性：未缺失值的数值、均值、标准差、范围、四分位数。

- dataframe的describe函数
  - 数值型特征的统计量

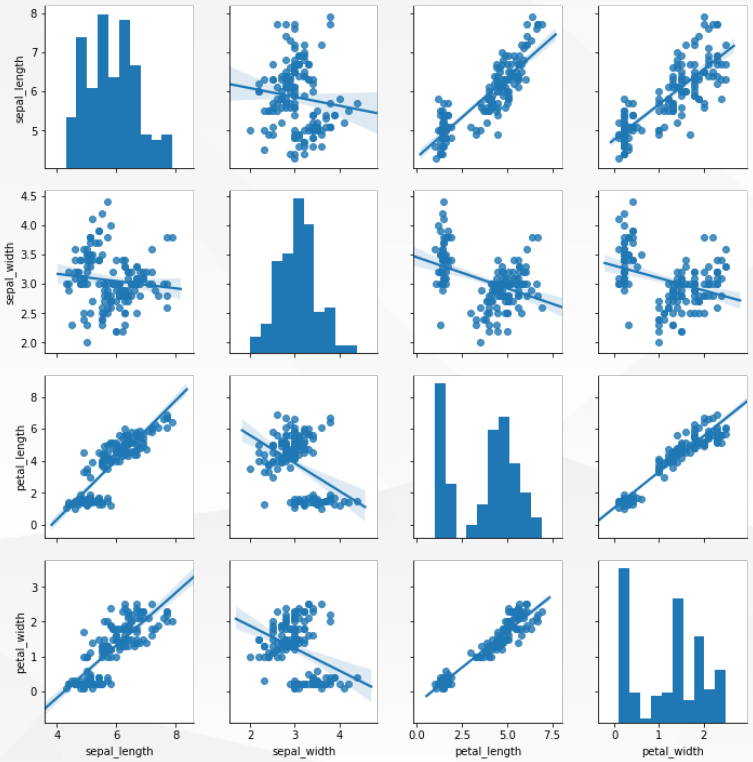
```
df.describe()
```

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

特征之间的相关性

- 可视化：散点图

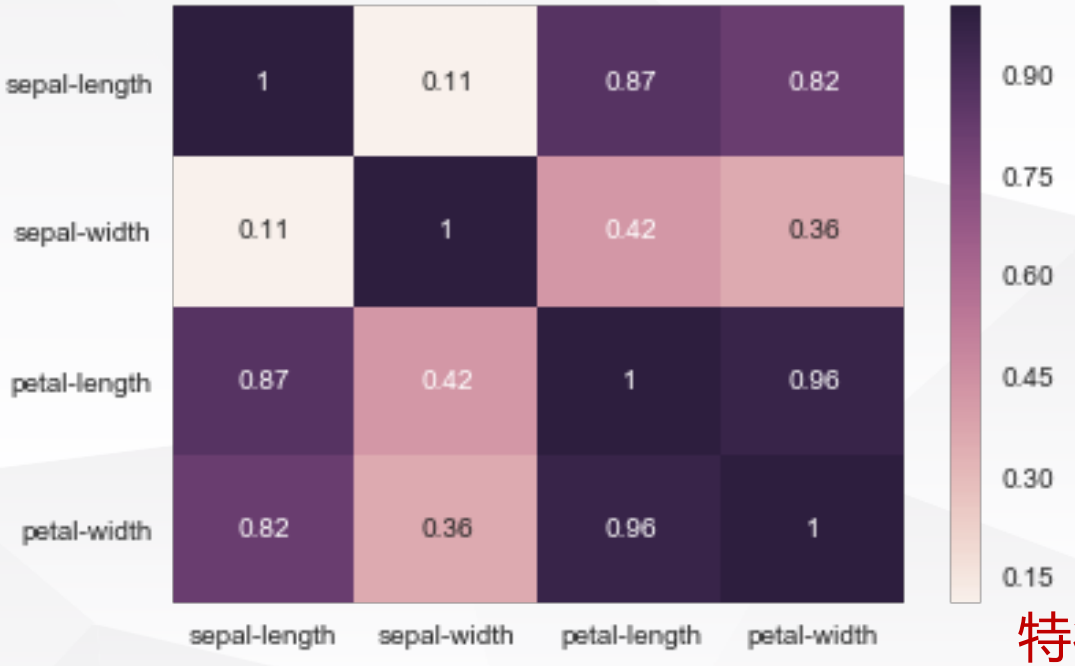
```
# 正对角线上的图表示数据频次的直方图，其他表示散点图
sns.pairplot(df, kind="reg")  #带回归线
```



特征之间的相关性

■数值型特征之间的相关系数：线性相关程度

```
feat_corr = df.corr().abs()
sns.heatmap(feat_corr , annot=True)
```



$$r=\frac{\sum_{i=0}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=0}^{N}(x_i-\bar{x})^2\sum_{i=0}^{n}(y_i-\bar{y})^2}}$$

$-1 \le r \le 1$

通常$|r|>0.5$，认为两者相关性比较强

$$\begin{cases} r=0 & \text{完全不线性相关} \\ r>0 & \text{正相关} \\ r<0 & \text{负相关} \end{cases}$$

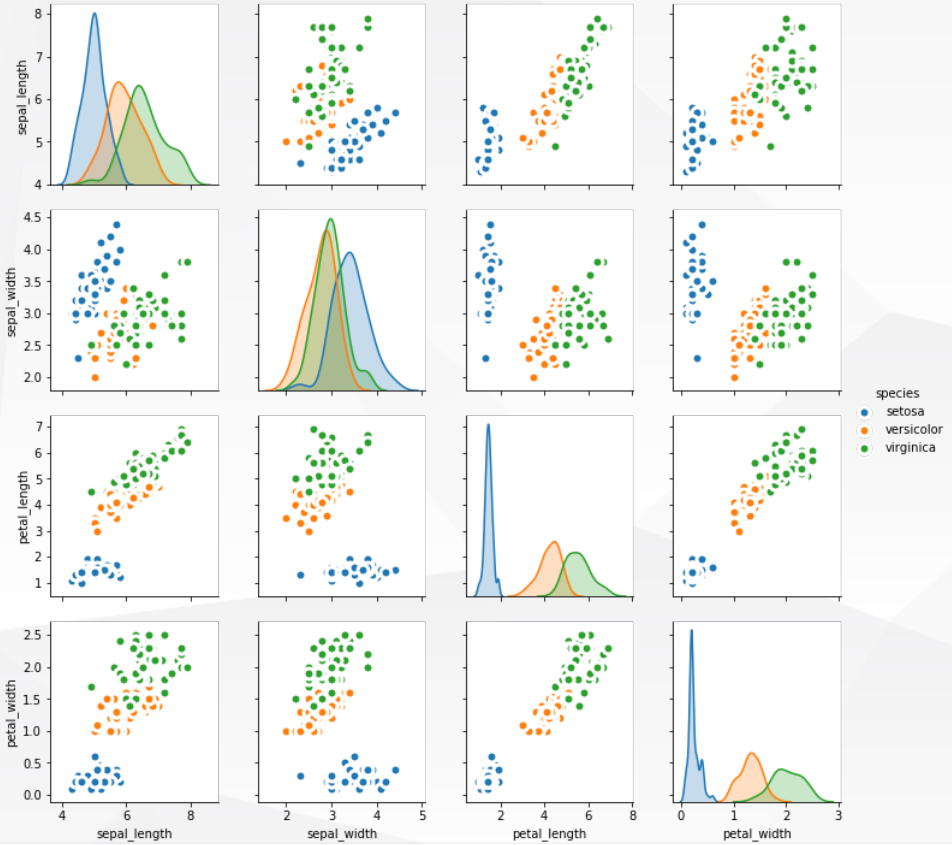
不线性相关并不代表不相关，

可能高阶相关，如$y=x^2$

特征与特征之间强相关的话意味着信息冗余

特征与目标之间的关系

```
# 通过hue设定种类， markers不同种类的点的表示方式
sns.pairplot(df, kind="scatter", hue="species", markers=["o", "s", "D"])
```



我们希望特征与标签强相关分类：不同类别的直方图差异大

特征工程

- 特征工程是原始数据与学习器的连接器。
- 首先，我们根据原始数据的特性初步确定学习器的类型。
- 原始数据可能不能直接输入到学习器。
  - 如字符串类型数据不能直接送入到线性回归等模型
- 在鸢尾花分类例子中，我们初步选择最近邻分类器（KNN）
  - KNN中涉及距离计算，特征的量纲/取值范围会影响距离的计算 → 数据预处理
- 更多特征工程内容后续讨论

数据预处理

- from sklearn.preprocessing import ...
  - 数据取值范围缩放
    数据标准化（Standardization）
    数据缩放（Scaling）
    数据正规化/归一化（Normalization）
  - ...

Scilit-learn中常用的特征缩放器

缩放器	说明
StandardScaler	标准化，对每维特征，将特征值取值范围标准化（0均值、1方差）
MinMaxScaler	区间缩放，对每维特征，将特征值缩放到[0, 1] 对非常小的标准差的特征更鲁棒 在稀疏数据中保留零元素
MaxAbsScaler	对每维特征，将特征值缩放到$[-1,1]$区间

Scilit-learn中特征缩放器的API

函数方法	功能
xxx.fit( )	拟合数据
xxx.fit_transform( )	拟合并转换数据
xxx.transform( )	转换数据
xxx.inverse_transform( )	逆转换

sklearn中的特征变换器（transformer）基本上都是如上接口，
如缺失值填补类SimpleImputer

①transformer.fit(X_train)

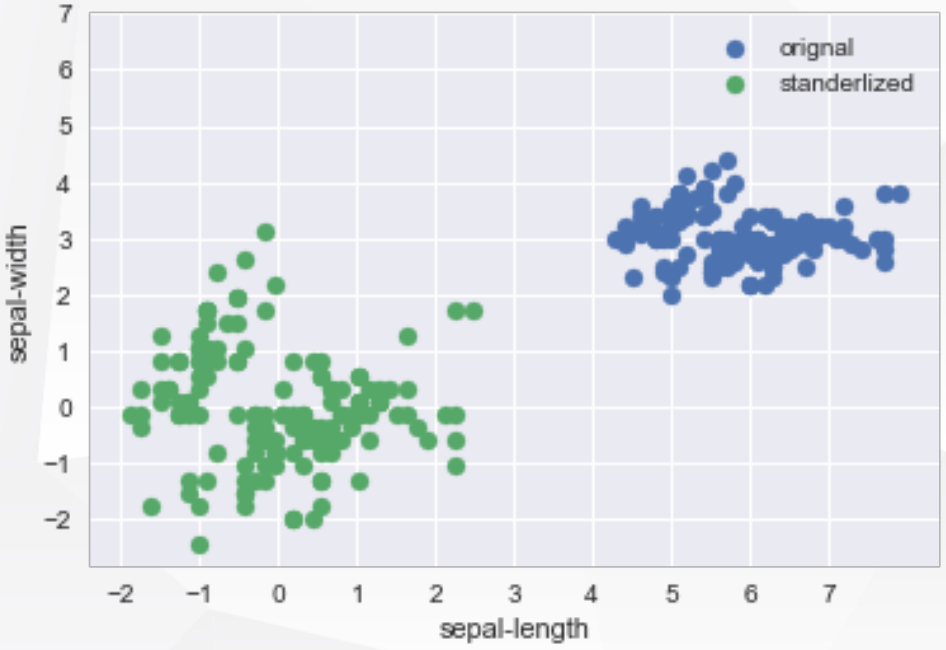
②X_train_transf = transformer.transform(X_train)

③X_test_transf = transformer.transform(X_test)

数据标准化

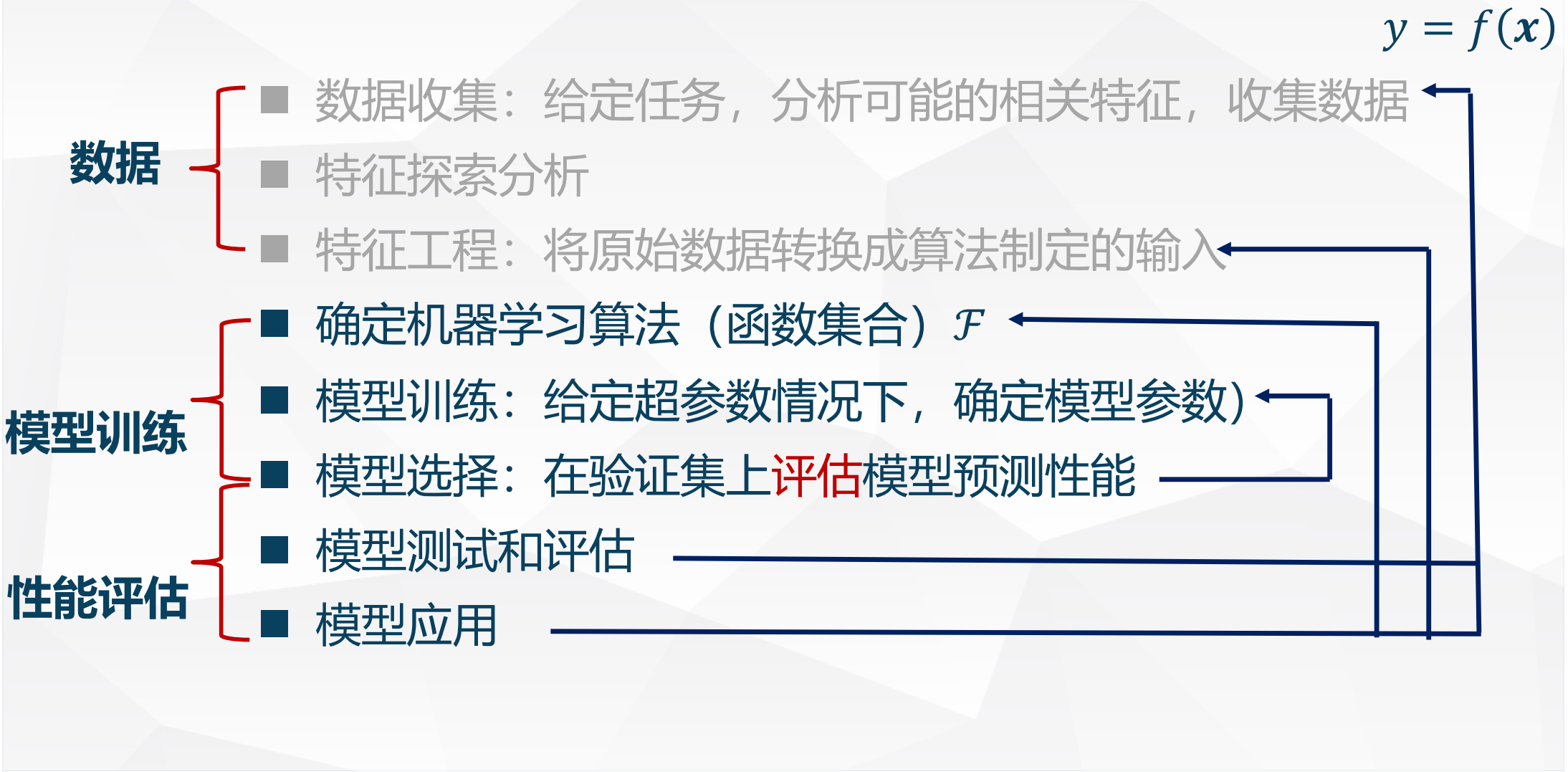
- 标准化：将输入特征的均值变为为0，方差为1

$$x_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sigma_j}$$



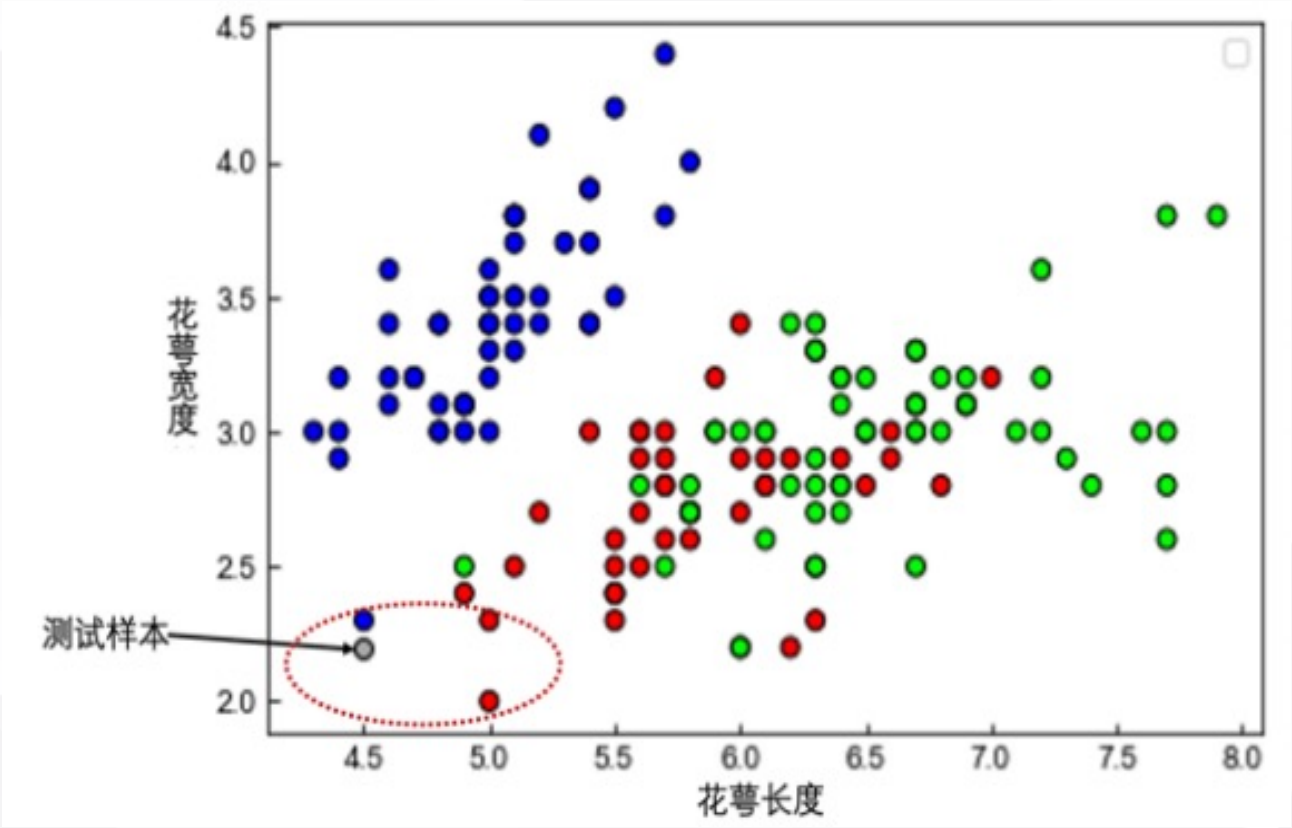

```
# 数据标准化
from sklearn.preprocessing import StandardScaler

# 构造输入特征的标准化器
ss = StandardScaler()

# 分别对训练和测试数据的特征进行标准化处理
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Recall：机器学习项目的开发过程



分类器：KNN

■K近邻（K-Nearest Neighbors， KNN ）：最简单的机器学习算法



“物以类聚，人以群分”

$K = 3$

距离度量

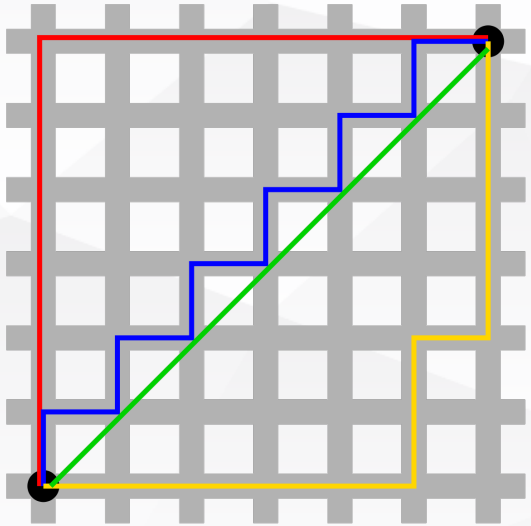
欧氏距离（L2）

$$\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(x_i - x_j)^{\mathrm{T}}(x_i - x_j)}$$

$$= \sqrt{\sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2}$$

曼哈顿距离（L1）

$$\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D} |x_{i,d} - x_{j,d}|$$



绿色线段的长度：欧氏距离

红色、黄色和蓝色折线的长度：走不同路径的曼哈顿距离。
虽然走的路径不同，但路径长度相同

K近邻学习器

- 输入：训练样本：$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$；
- 最近邻的数目K、距离度量函数dist( )
- 测试样本$\boldsymbol{x}_0$
- 输出：测试样本$\boldsymbol{x}_0$的标签$\hat{y}_0$

- 1. 计算测试样本$\boldsymbol{x}_0$与所有训练样本与的距离$d_i = \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_0), i = 1,2, \ldots, N$；
- 2. 对距离$\{d_i\}_{i=1}^N$进行升序排序；
- 3. 取K个最小距离对应的K个样本构成最近邻集合$\mathcal{N}$，综合邻居信息，
  - 对分任务，$\hat{y}_0 = \mathrm{vote}(\mathcal{N})$；
  - 对回归任务，$\hat{y}_0 = \frac{1}{K}\sum_{\boldsymbol{x}_j \in \mathcal{N}} y_j$。

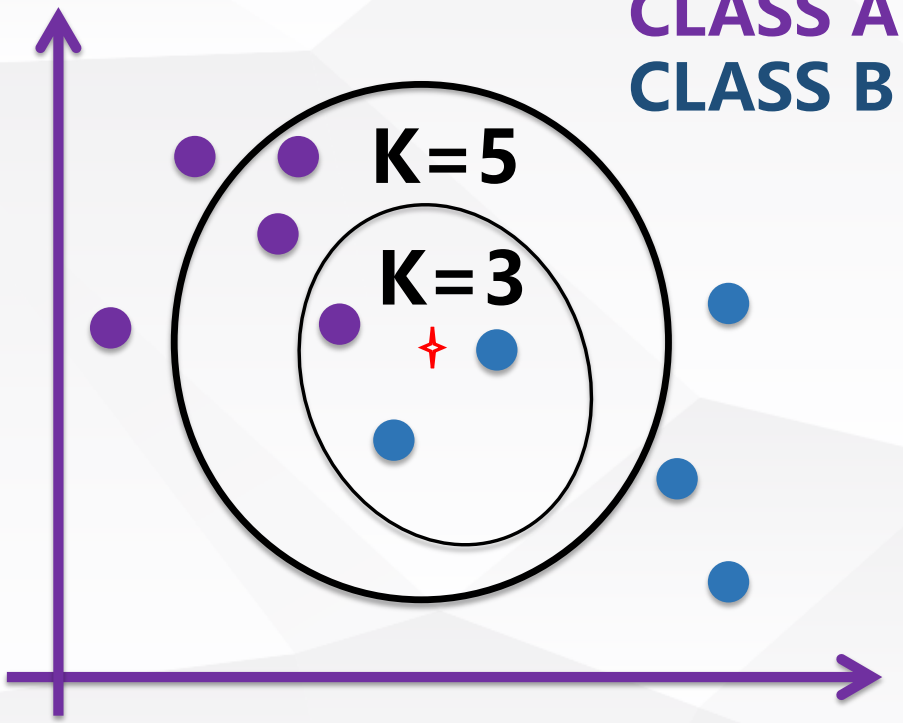
例：用KNN对鸢尾花进行分类

	花萼长度	花萼宽度	花瓣长度	花瓣宽度	类别	到x_0的欧氏距离	到x_0的曼哈顿距离
x_1	5.1	3.5	1.4	0.2	setosa	**0.6**	**1.0**
x_2	4.9	3	1.4	0.2	setosa	**0.3**	**0.5**
x_3	4.7	3.2	1.3	0.2	setosa	**0.2**	**0.4**
x_4	7	3.2	4.7	1.4	versicolor	4.2	6.9
x_5	6.4	3.2	4.5	1.5	versicolor	3.7	6.2
x_6	6.9	3.1	4.9	1.5	versicolor	4.3	7.0
x_7	6.3	3.3	6	2.5	virginica	5.3	8.7
x_8	5.8	2.7	5.1	1.9	virginica	4.2	6.9
x_9	7.1	3	5.9	2.1	virginica	5.4	8.9
x_0	4.6	3.1	1.5	0.2	?		

setosa

最近的3个邻居均为x_3、x_2和x_1

K的取值



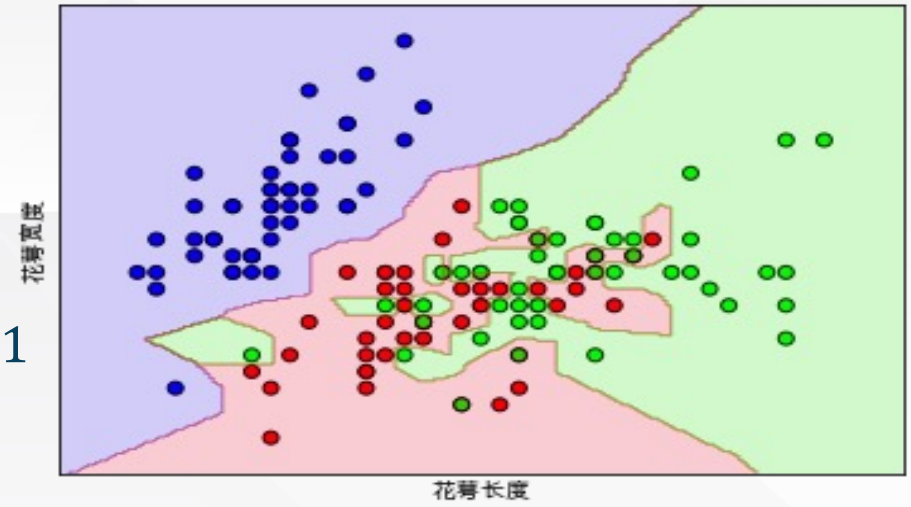
CLASS A
CLASS B

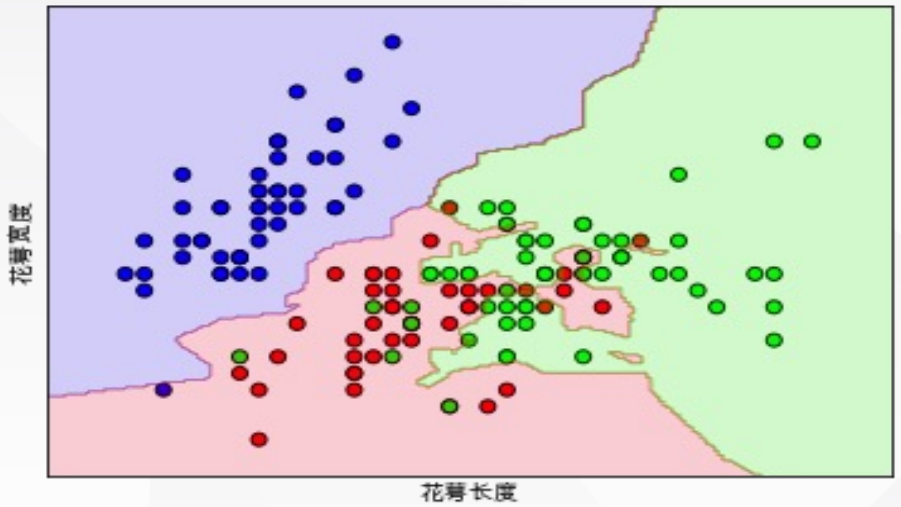
If K = 3
统计： 1 A
2 B
结果： Class B

If K = 5
统计： 3 A
2 B
结果： Class A

K的取值：鸢尾花分类

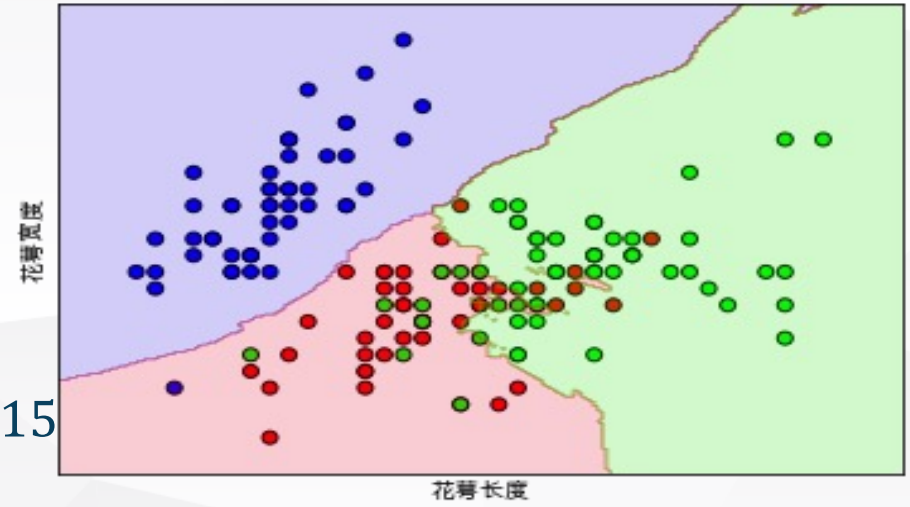
$K = 1$

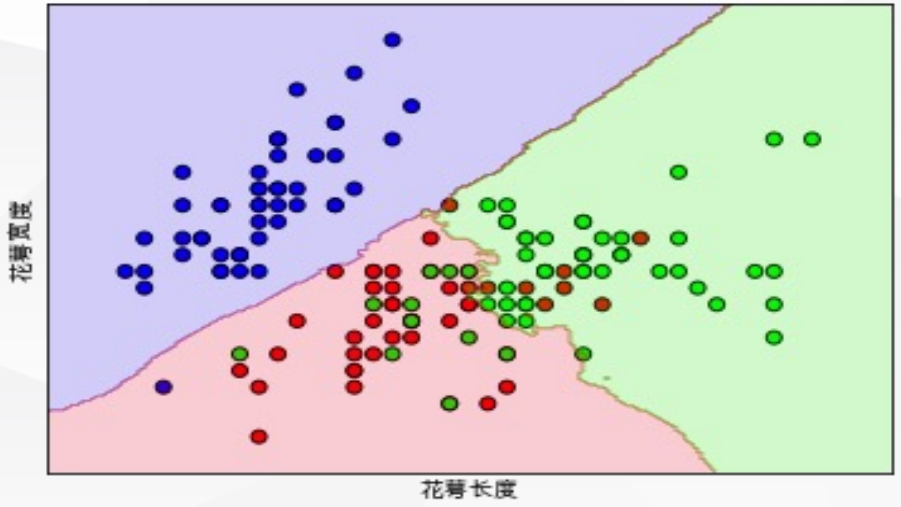




$K = 3$

$K = 15$





$K = 31$

K的取值

- 直觉：模型在训练数据集上性能要好，决策边界也不能太复杂（我们很难记住复杂的模型）。

- 模型越复杂，越能拟合训练数据，在训练数据集上的性能越好，但决策边界不平滑。
  - 例如$K = 1$时、训练样本全部被分类正确，但决策边界很不光滑

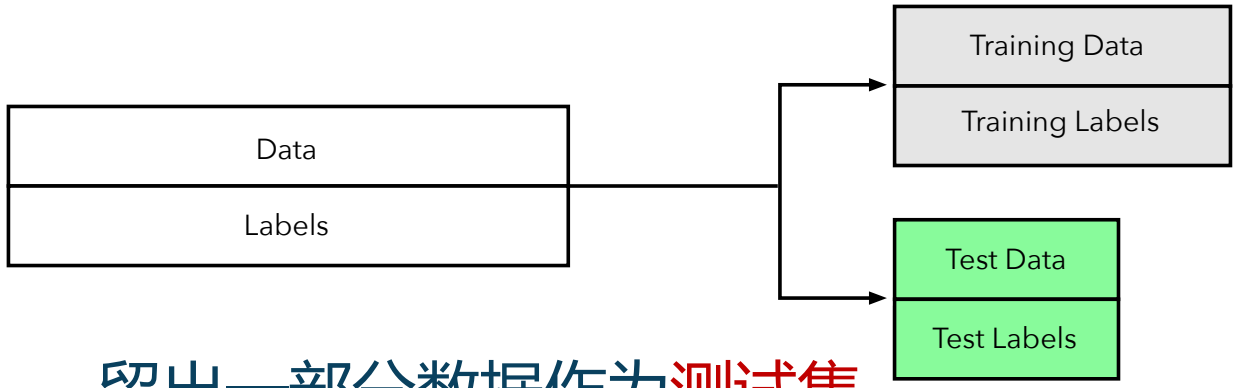
- 模型越简单，越不能拟合训练数据，在训练数据集上的性能越不好，但决策边界越平滑。
  - 例如当$K = 31$时，一些训练样本被分类错误，但决策边界光滑

最佳的K：测试集上性能最好

- 机器学习的目的不是在训练数据集性能好，而是要在假设空间中找到一个最好的模型，使得这个模型在未见过的测试集上的性能最好。
- 模型在测试集上的性能称为模型的泛化性能。

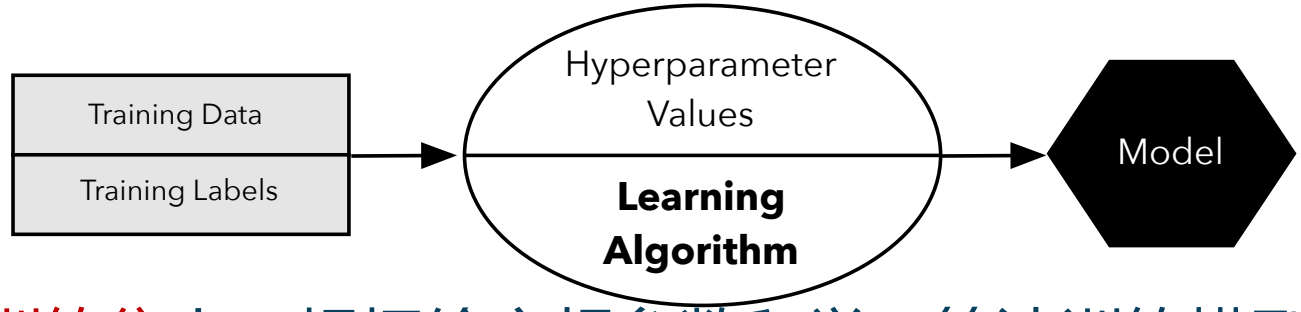
模型性能评估

1



留出一部分数据作为测试集

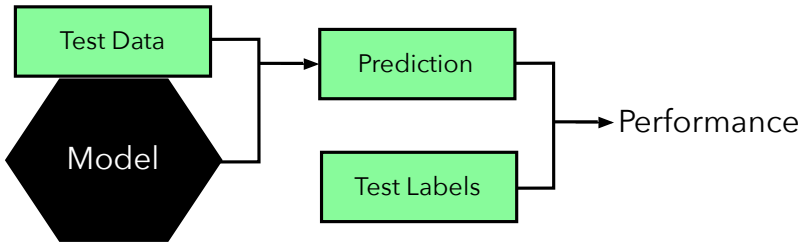
2



在训练集上，根据给定超参数和学习算法训练模型

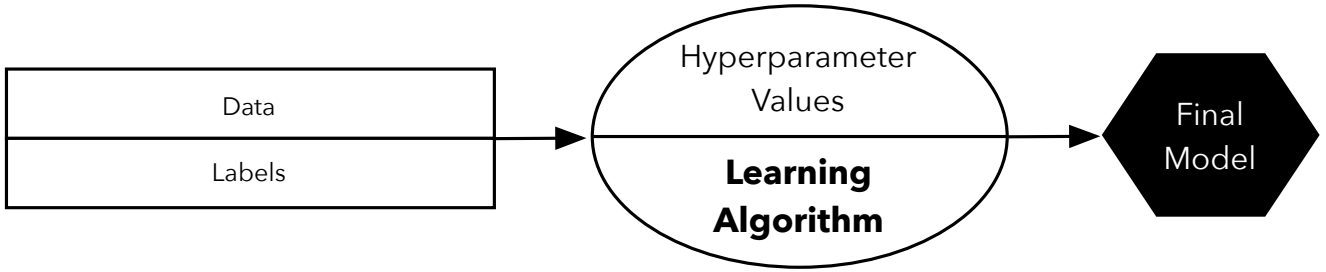
模型性能评估

3



在留出的测试集用训练好的模型预测
比较预测结果和测试集的真实标签评估模型的性能

4



根据所有数据，和确定的最佳超参数，
重新训练模型（最终模型）

Scikit-learn中的数据划分函数

sklearn.model_selection.train_test_split（* arrays， ** options ）

参数	说明
*arrays	**sequence of indexables with same length / shape[0]** 输入可以是列表、numpy数组、稀疏矩阵或padas中的DataFrame
test_size	**float or int, default=None** float：[0.0，1.0]，测试集占数据集的比例；int：测试集样本的数目 None：测试集为训练集的补集。若train_size也是None，取值为0.25
train_size	**float or int, default=None，**类似**test_size**
random_state	**int or RandomState instance, default=None**
shuffle	**bool, default=True，**切分前是否对数据进行打乱。
stratify	**array-like, default=None** 如果不为None，以分层方式切分数据，并将其用作类标签

返回**splitting**： **list, length=2 * len(arrays)，**切分后的训练集和测试集

例：数据集划分

```
from sklearn.model_selection import train_test_split

X = np.arange(1, 25).reshape(12, 2)
y = np.array([0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0])

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.33,
random_state=4, stratify=y )
```

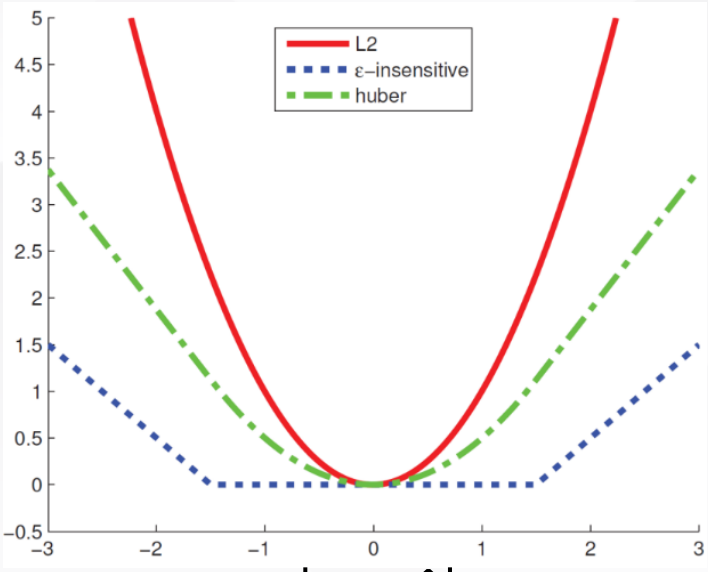
分层法能使测试更加稳定、适合小而不平衡的数据集

```
X_train
array([[21, 22], [ 1, 2], [15, 16], [13, 14], [17, 18], [19, 20], [23, 24], [ 3, 4]])
X_test
array([[11, 12], [ 7, 8], [ 5, 6], [ 9, 10]])
y_train
array([1, 0, 1, 0, 1, 0, 0, 1])
y_test
array([0, 0, 1, 1])
```

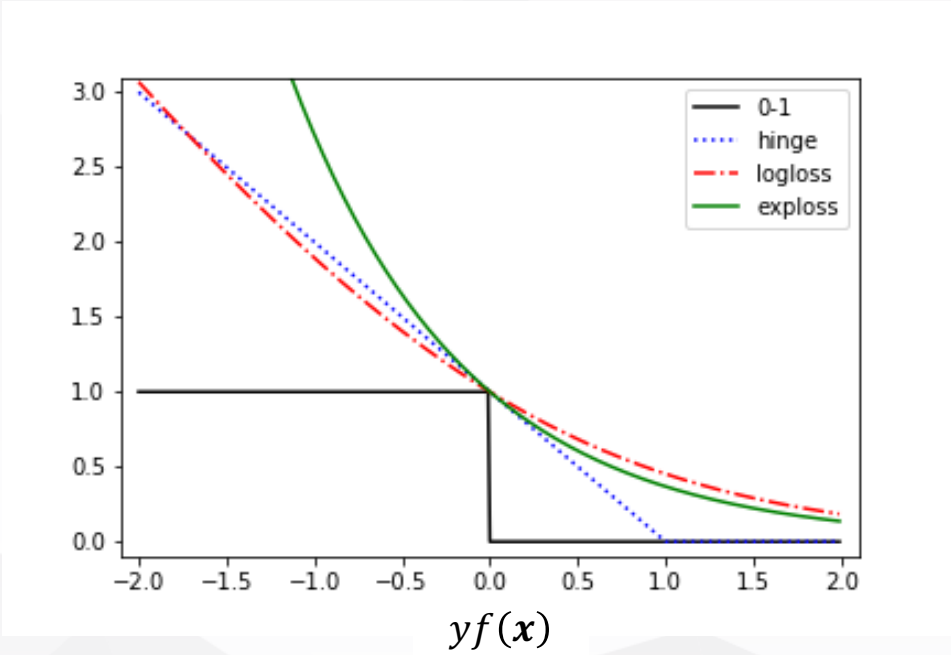
训练集和测试集中0/1的比例相同，
同全体数据相同（0和1各占0.5）

x		y
1	2	0
3	4	1
5	6	1
7	8	0
9	10	1
11	12	0
13	14	0
15	16	1
17	18	1
19	20	0
21	22	1
23	24	0

模型性能评价指标



$|y - \hat{y}|$



$yf(\boldsymbol{x})$

回归：均方根误差、平均绝对误差、平均相对误差、R2分数、…

分类：分类正确率、查全率/P、查准率/R、 F1分数、 P-R曲线、ROC/AUC、…

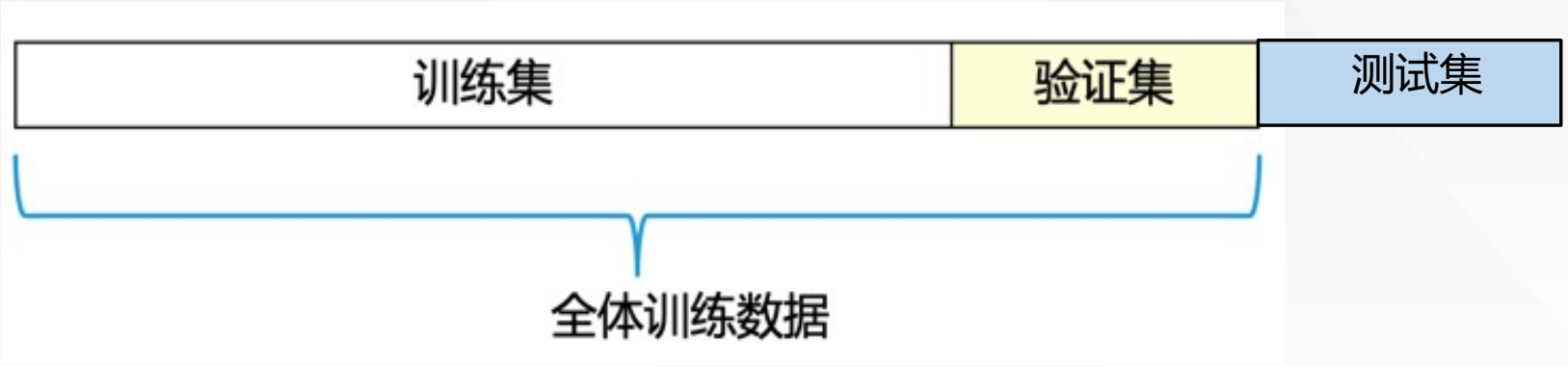
例：性能评估指标

```
from sklearn.metrics import accuracy_score

# y_pred是预测标签
y_pred, y_true=[1,2,3,4], [2,2,3,4]
accuracy_score(y_true=y_true, y_pred=y_pred)
```

输出：0.75

验证集：寻找最佳的K

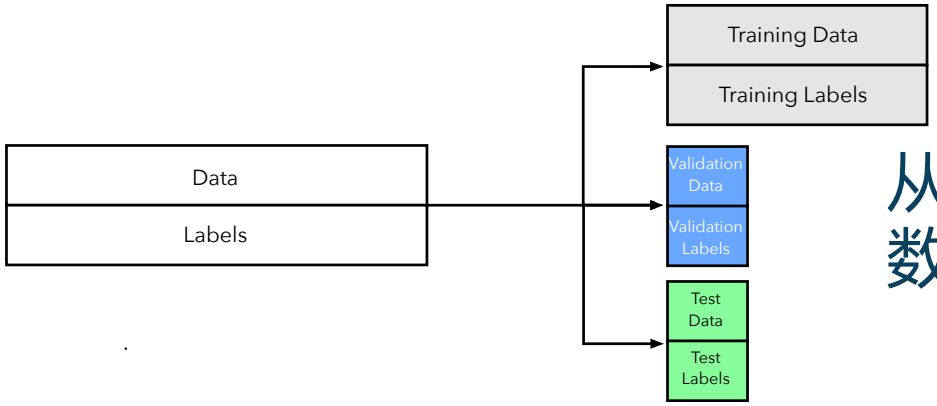
- 在训练集中留出一部分数据做为验证集，以模型在验证集上的性能做为模型泛化性能的估计，并以此作为选择K的依据。



验证集的切分方法同测试集的划分：train_test_split

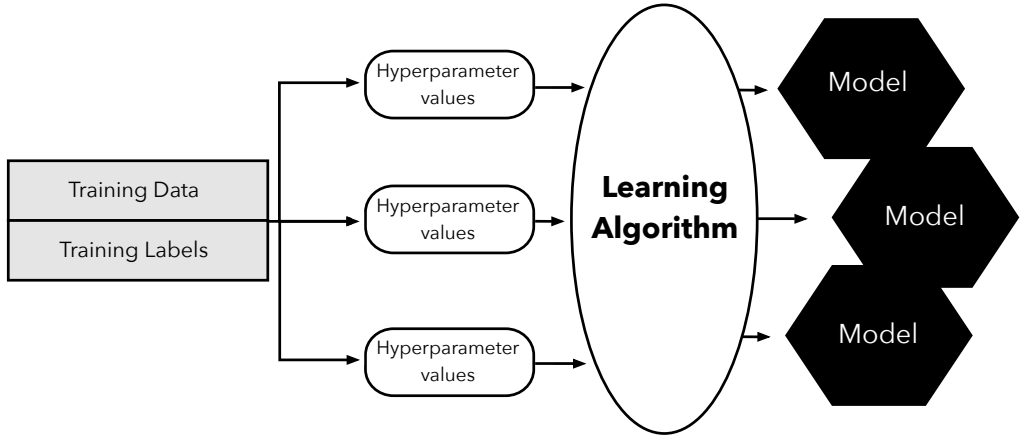
留出法验证

1



从训练集中留出一部分数据作为验证集

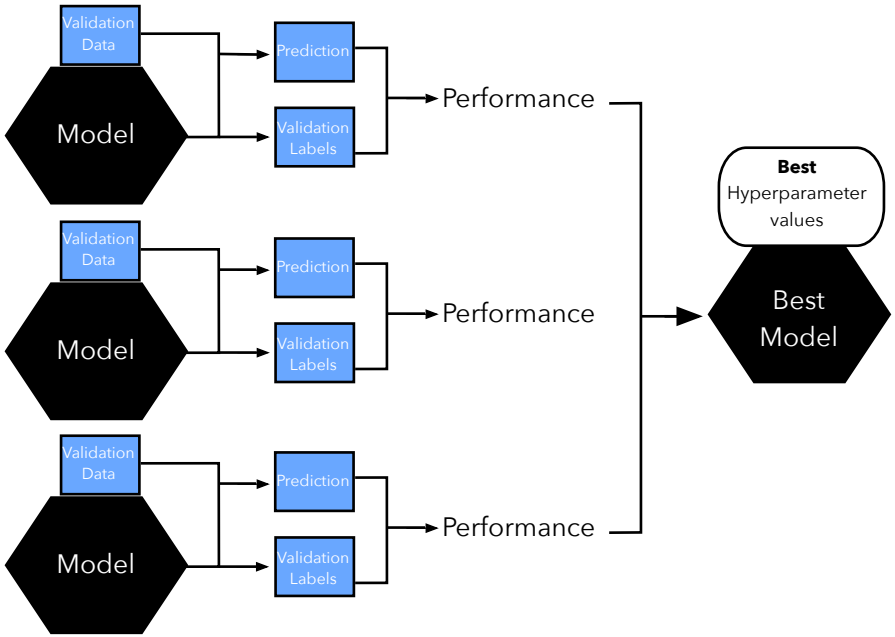
2



在训练集上，训练不同超参数对应的学习算法，得到多个不同超参数对应的模型

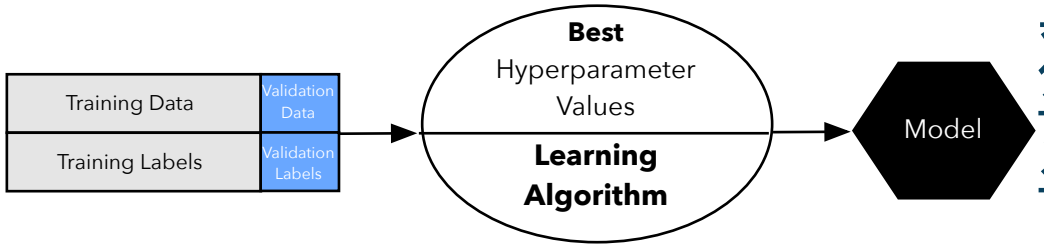
留出法验证

3



在验证集训练集上对多个不同超参数对应的模型进行性能评估，选择性能最好的模型

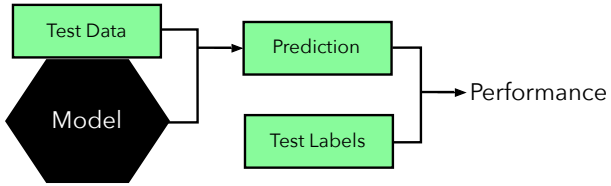
4



对最佳超参数，在所有训练数据（训练集+验证集）上训练模型

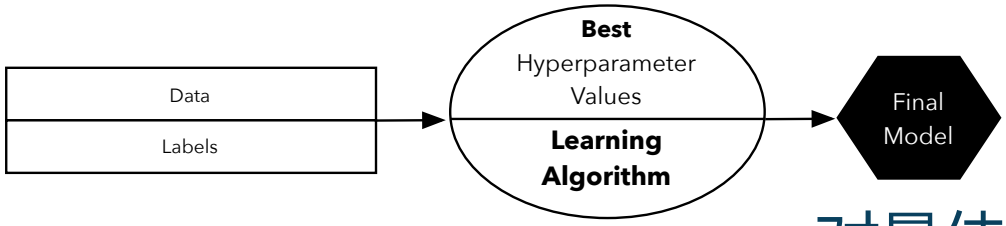
留出法验证

5



对最佳超参数上用所有训练数据训练好的模型
在测试集进行性能评估

6



对最佳超参数，在所有数据（训练集+验证集+测试集）上训练模型

交叉验证：寻找最佳的K

- 但从训练集中分离一部分数据作为验证集，会使得模型训练能使用的数据减少（机器学习中训练样本越多越好）。有更好的办法?
- K折交叉验证

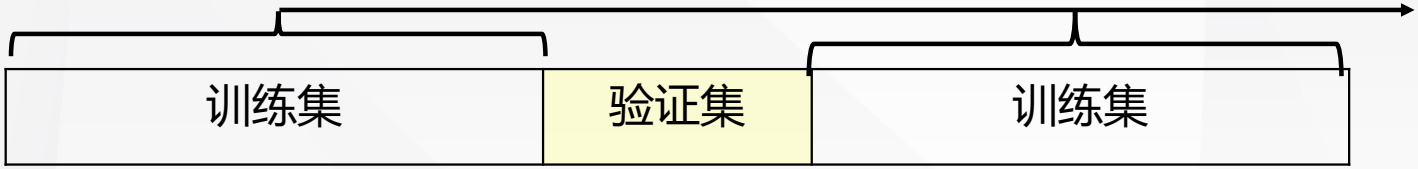
注意：该方法计算代价很高，
但不会浪费太多的数据，
当样本数据集较少时有优势。

交叉验证：寻找最佳的K

$$f(\boldsymbol{x};\widehat{\boldsymbol{\theta}}^1,\lambda)$$

$$e_1 = e(\boldsymbol{y}_{test\_1}, f(\boldsymbol{X}_{test1};\hat{\theta}^1,\lambda))$$

...



$$f(\boldsymbol{x},\widehat{\boldsymbol{\theta}}^k,\lambda)$$

...

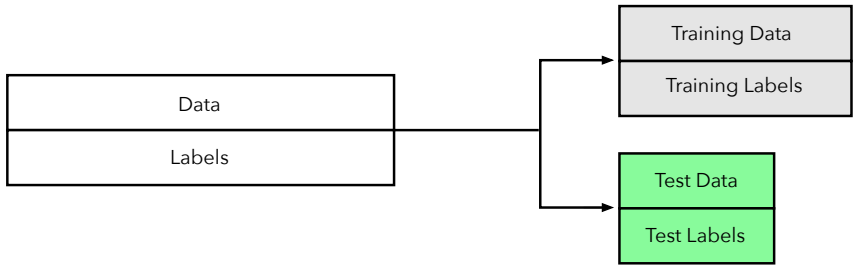
$$e_k = e(\boldsymbol{y}_{test\_k}, f(\boldsymbol{X}_{test\_k};\widehat{\boldsymbol{\theta}}^k,\lambda))$$

超参数为λ的模型的性能：

$$e_\lambda = \frac{1}{K}\sum_{k=1}^{K} e_k$$

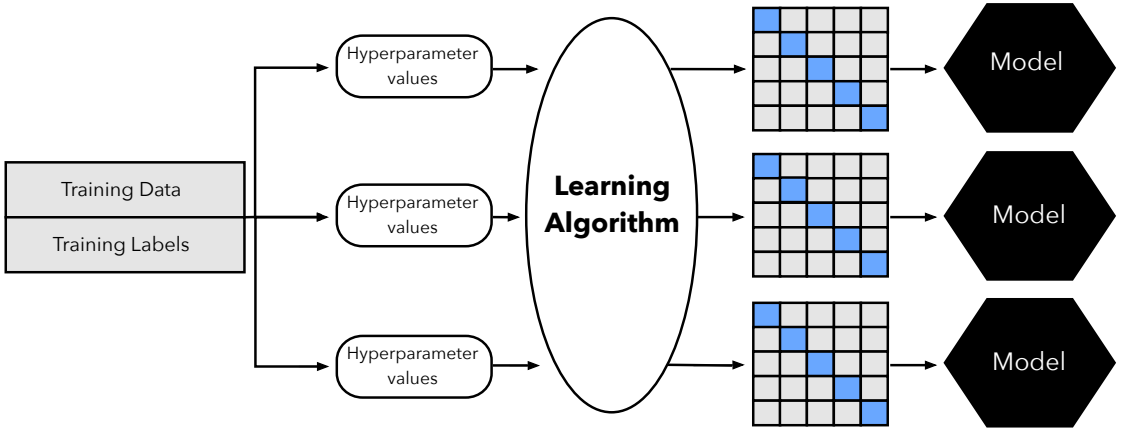
K折交叉验证

1



留出一部分数据作为测试集

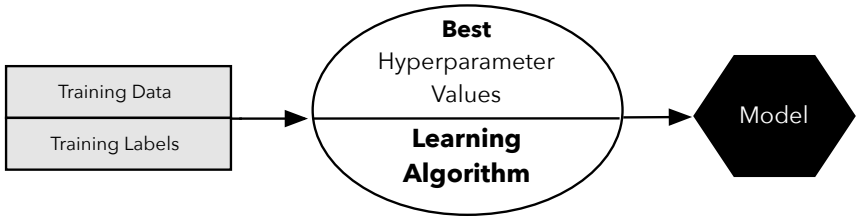
2



对训练集采用K折交叉验证：根据每折的训练数据，训练不同超参数对应的学习算法，得到每个超参数每折交叉验证对应的模型，并在每折的验证集上评估每个超参数对应的模型的性能

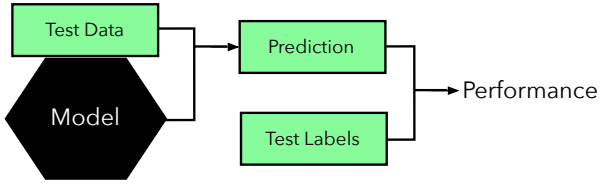
K-fold Cross-Validation Pipeline II

3



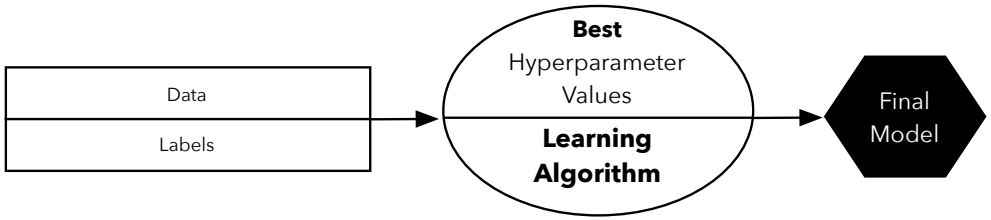
对最佳超参数，在所有训练数据上训练模型

4



对最佳超参数上用所有训练数据训练好的模型在测试集进行性能评估

5



对最佳超参数，在所有数据上训练模型

交叉验证数据划分

- `from sklearn.model_selection import KFold`
  - Kfold：交叉验证，直接随机的将数据划分为k折，不受class和影响

- `from sklearn.model_selection import StratifiedKFold`
  - StratifiedKFold 分层交叉验证：根据数据集的分布来划分，使得 划分后的数据集的目标比例和原始数据集近似

StratifiedKFold

class sklearn.model_selection.StratifiedKFold(n_splits=5, *, shuffle=False, random_state=None)

参数

•n_splits: 折叠次数，默认为3，至少为2。

Methods

StratifiedKFold.split(X, y, groups=None)

•返回值： 训练集数据的 index 与验证集数据的 index。

```
from sklearn.model_selection import StratifiedKFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([0, 0, 0, 1, 1, 1])
skf = StratifiedKFold(n_splits=3)
for train_index, test_index in skf.split(X, y):
        print("TRAIN:", train_index, "TEST:", test_index)
```

输出：

```
TRAIN: [1 2 4 5] TEST: [0 3]
TRAIN: [0 2 3 5] TEST: [1 4]
TRAIN: [0 1 3 4] TEST: [2 5]
```

Scikit-learn实现的交叉验证模型评估

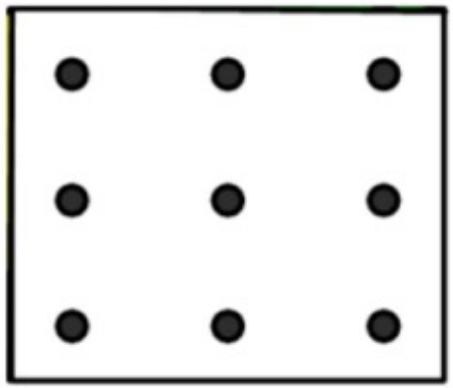
- scikit-learn 利用model_selection模块中的cross_val_score函数来实现交叉验证，其参数为：要评估的模型、训练数据与真实标签

```
from sklearn.model_selection import cross_val_score

#训练分类器
knn = KNeighborsClassifier(n_neighbors = 3)

scores = cross_val_score(knn, X_train, y_train)
print("Cross-validation scores: {}".format(scores))
print("Average cross-validaton score: {:.2f}".format(scores.mean()))
```

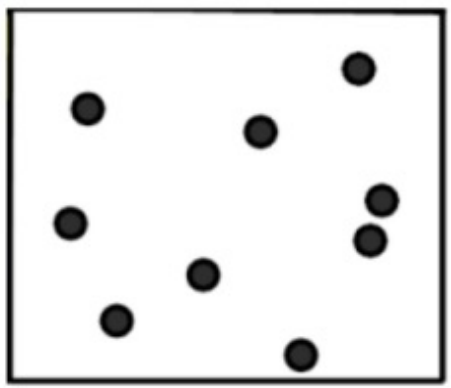
```
Cross-validation scores: [0.95833333 0.95833333 0.95833333 0.95833333 1. ]
Average cross-validaton score: 0.97
```

超参数K的范围设置与搜索策略

- 超参数的搜索范围可以通过分析给定任务的特点和背景知识确定。
  - 例如：$5 \leq K \leq 31$
- 如果超参数的搜索范围较小，我们可以采用网络搜索的方式寻找最佳超参数。
- 如果超参数的搜索范围太大，在给定的时间内无法穷举搜索所有超参数，可对超参数空间采用随机搜索。



网格搜索



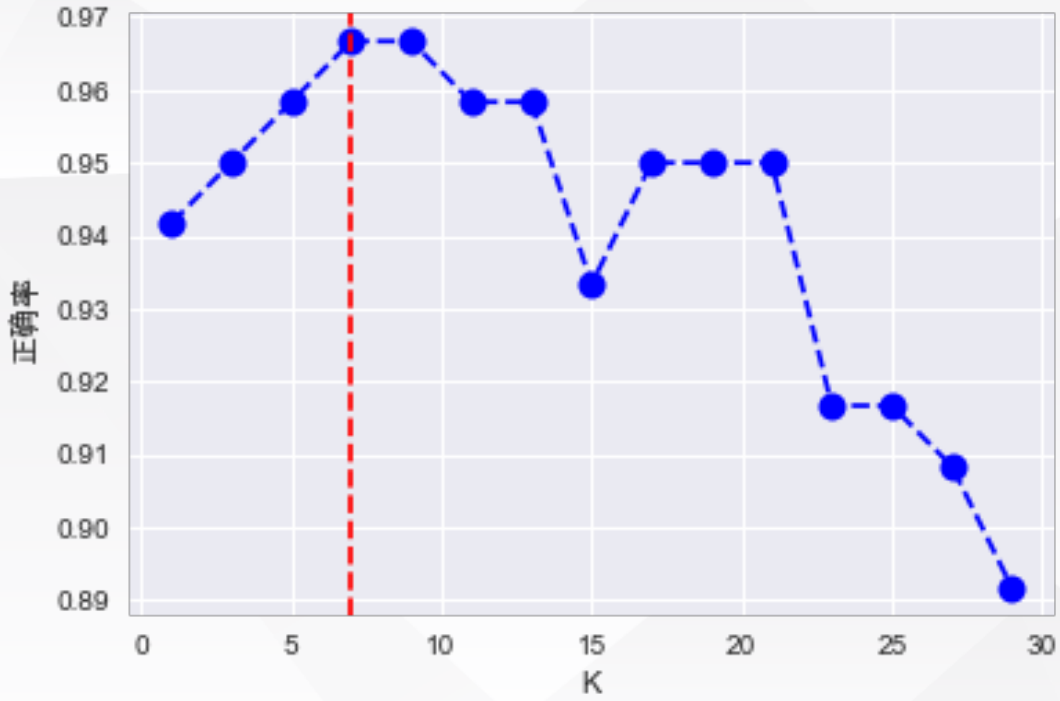
随机搜索

Scikit-learn中的GridSearchCV

- GridSearchCV结合了网格参数搜索和交叉验证

```
from sklearn.model_selection import GridSearchCV

#设置超参数搜索范围
Ks = range(1, 31, 2)
tuned_parameters = dict(n_neighbors = Ks)

#生成学习器实例
knn = KNeighborsClassifier()

#生成GridSearchCV实例
grid= GridSearchCV(knn, tuned_parameters,cv=10, scoring='accuracy',n_jobs = 4)

#训练，交叉验证对超参数调优
grid.fit(X_train,y_train)
```

10折交叉验证选择最佳的K



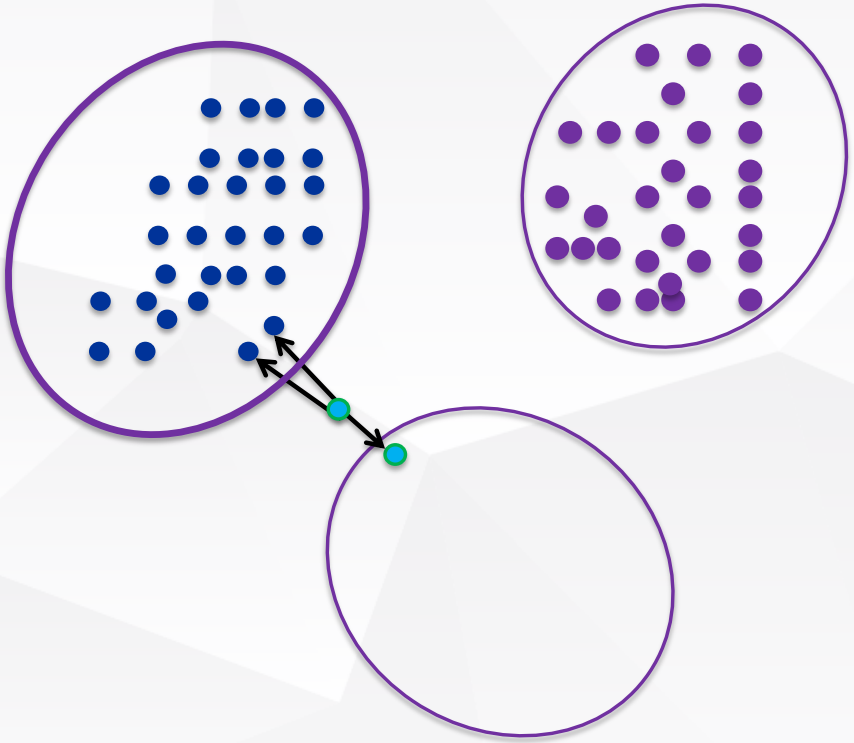
选择150个样本中的120为训练集，其余30个样本为测试集
在120个样本中，采用10折交叉验证寻找最佳的K=7（K=9）
该最佳模型在训练集上交叉验证的到的正确率为0.9667
该模型在测试集上的误差为0.9000

KNN算法优点

- 无需训练
- 不需要庞大的训练样本集
  - K值不会取得特别大的值，不需要特别大的样本数量就能够完成一个简单的分类任务。
- 对异常值不敏感
  - KNN关注的是离测试样本比较近的这些数据点，那么离他比较远的这些数据点自然而然的就被排除在外。
- 是天然的多分类器
  - SVM等只能做两类分类，对多类分类任务，后期需要采用一些其他的策略

KNN算法缺点

- 数据量大时计算量太大
  - 维度D： 距离计算需访问每维特征
  - 样本数目N：需要计算每个测试样本与所有训练样本的距离，并对距离进行排序
- 处理不平衡样本的能力差
- 测试时计算成本高
  - 可接受训练有一定耗时
  - 但测试时要效率高



Scikit-learn实现的KNN

■KNN分类器：

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

■KNN回归器：

class sklearn.neighbors.KNeighborsRegressor(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

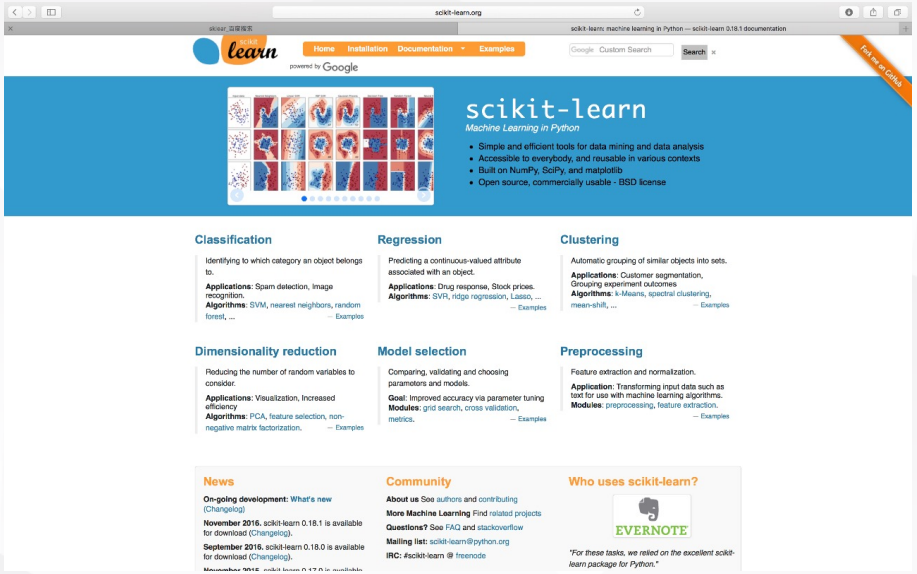
KNN的参数	参数含义	说明
n_neighbors	近邻数目K	通过验证集上的性能选择一个最优的K，默认值是5。
weights	近邻的权重	‘uniform’，：每个近邻的权重相同 ‘distance’ ：权重和距离成反比例 自定义权重：自定义一个函数，输入是距离值，输出是权重
algorithm	搜索K近邻的算法	‘brute’ ：蛮力 ‘kd_tree’ ： K-D树，推荐 ‘ball_tree’ ：球树 ‘auto’ ：在上面3种方法中选择一个拟合最好的最优算法，默认值
metric	样本之间的距离度量	‘euclidean’ ：欧氏距离 ‘manhattan’ ： 曼哈顿距离 ‘chebyshev’ ：切比雪夫距离 ‘minkowski’ （默认）：闵可夫斯基距离 ‘wminkowski’ ：带权重的闵可夫斯基距离 ‘seuclidean’ ：对各特征维度做标准化缩放以后再计算欧氏距离 ‘mahalanobis’ ：马氏距离

Scikit-learn

- 机器学习包：Sklearn

六个部分：
分类（Classification）
回归（Regression）
聚类（Clustering）
数据降维（ Dimensionality Reduction）

模型选择（Model Selection）
数据预处理（Preprocessing）



官网： http://scikit-learn.org/stable/

中文版用户手册：
http://sklearn.apachecn.org/cn/0.19.0/

各种机器学习算法接口统一

Scikit-learn API

■各种机器学习算法接口统一

`estimator.fit(X_train, [y_train])`	
`estimator.predict(X_test)`	`estimator.transform(X_test)`
Classification	Preprocessing
Regression	Dimensionality Reduction
Clustering	Feature Extraction
	Feature selection

$f: \boldsymbol{x} \rightarrow y$

$f: \boldsymbol{x} \rightarrow \boldsymbol{z}$

机器学习项目开发环境推荐

Python工具包 & 环境

- 包 = “工具”；
- 下载包 = “买工具”；
- 写程序 = "用工具做东西"（程序import导入）

- 环境 = “楼里的一间屋”，每间屋存放各种‘包’，每间屋里的‘包’互不影响"
- 激活环境 = “选择一间屋”，使用该屋里的‘包’来做东西
- 移除环境 = “删除屋里的东西”，节省电脑空间

使用Anaconda Python发布和包管理器

- 推荐安装来自Continuum Analytics的开源包管理系统conda。
- conda免费并且采用了许可式的开源证书，帮助在各操作系统中安装数据科学、数学和工程的Python以及其版本管理。
- conda的选择
  - Anaconda：预装了很多科学计算包，新手推荐
  - Miniconda： Anaconda的简化版，和Anaconda基本一致，只是没有预装包
  - Miniforge： Miniconda的一个很好的替代
- 在成功通过 Anaconda、Miniconda或Miniforge安装好了conda后，可执行如下命令安装新的Python包

```
conda install SomePackage
conda update SomePackage
```

Anaconda

- Anaconda是一个方便的Python包管理和环境管理软件，一般用来配置不同的项目环境。
- **管理包**： Anaconda 是在 conda（一个包管理器和环境管理器）上发展出来的。可以使用 conda 来安装、更新 、卸载工具包。
  - 在安装 Anaconda 时就预先集成了像 Numpy、Scipy、 pandas、Scikit-learn 这些在数据分析中常用的包。
- **虚拟环境管理**：在conda中可以建立多个虚拟环境，用于隔离不同项目所需的不同版本的工具包，以防止版本上的冲突。
  - 例如：正在做的项目A和项目B分别基于python2和python3，使用Anaconda创建多两个互不干扰的环境，分别运行不同版本的软件包，以达到兼容的目的

https://www.anaconda.com/download/

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/

开发环境

- 编程语言：Python（建议使用Python 3的最新稳定版本）
  - 集成开发环境：VS Code、PyCharm
  - 交互式开发环境：Jupter Notebook
- 数据处理工具包： NumPy、SciPy、Pandas
- 数据可视化工具包：Matplotlib、Seaborn
- 机器学习工具包：Scikit-learn
- 深度学习工具包：PyTroch、Tensorflow/Keras、CNTK、MXNET

软件安装

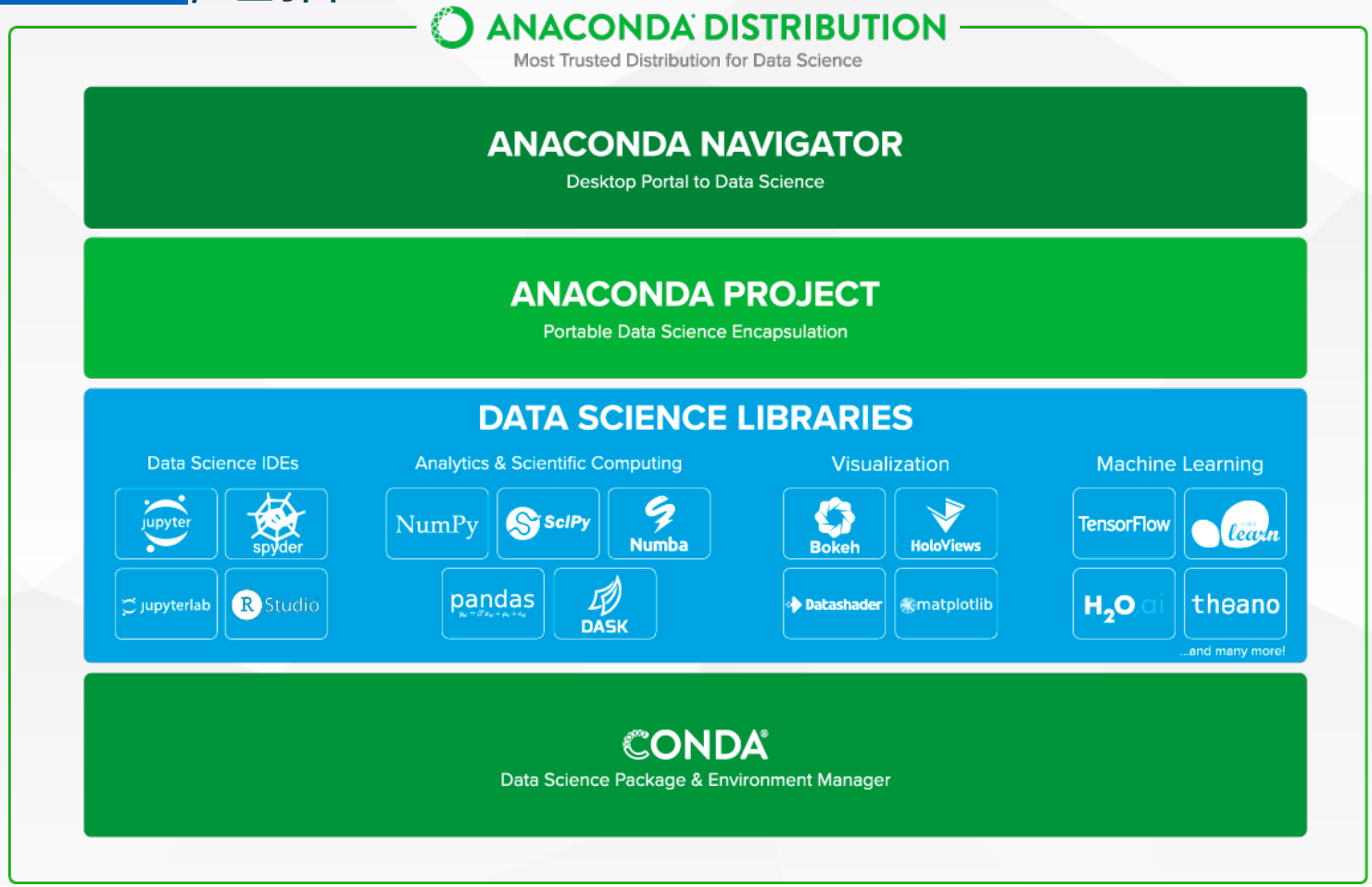
https://www.anaconda.com/download/

■推荐直接安装Anaconda, 包括

很多有用的工具包

集成开发环境:
Jupyter Notebook

conda package manager



课程作业安排

■练习作业（计算、简答、推导）共7次：2024年1月4日前提交

- 手写，拍照放在 word里上传；也可直接在word等文档编辑器里直接完成

■编程实践作业共1次，占作业总成绩的30%：2024年1月19日前提交

- 附件形式打包提交，命名“学号-姓名-作业3.zip”，包括：1. 实验报告 2.可运行代码（建议ipynb格式）

■课后思考题 （不需要提交）

作业迟交酌情扣分，最长不能超过两周，逾期两周未交的作业按0分记录。

练习作业

- 比较重要的算法如Logistic回归、 SVM等会有一些简单的思考题，加深大家去算法的理解。
- 例如：请给出Logistic回归中训练误差和测试误差与正则参数C的关系

编程实践作业

- 从下述四类方法中各选1种方法，解决实际问题
  - 线性方法：线性SVM、 Logistic Regression
  - 非线性方法：Kernel SVM， 决策树
  - 集成学习：Bagging， Boosting
  - 神经元网络：自选结构

- 数据集
  - 课程推荐3种数据集：电信用户流失、手写数字识别、CIFA10
  - 自己科研数据集
  - 竞赛网站上感兴趣的数据集

电信用户流失预测数据集

- 当产品无法留住用户时，产品就像是一个筛子，这也使得放进的砂砾越多流失的也就越多。对于客户流失率而言，每增加5%，利润就可能随之降低25%-85%。随着市场饱和度的上升，电信运营商亟待解决增加用户黏性，延长用户生命周期的问题。因此，电信用户流失分析与预测至关重要。

- 案例数据集选自Kaggle——电信客户流失分析
  (Telco Customer Churn)

- 数据集共有7043条数据，20个字段，

序号	字段名	字段描述
1	customerID	客户ID
2	gender	性别
3	SeniorCitizen	是否是老年人
4	Partner	是否有伴侣
5	Dependents	是否有家属
6	tenure	使用产品时长
7	PhoneService	是否开通电话服务业务
8	MultipleLines	是否开通多线业务
9	InternetService	是否开通互联网服务
10	OnlineSecurity	是否开通网络安全服务
11	OnlineBackup	是否开通在线备份
12	DeviceProtection	是否开通设备保护
13	TechSupport	是否订购技术支持服务
14	StreamingTV	是否订购网络电视
15	StreamingMovies	是否订购网络电影
16	Contract	签订合同方式
17	PaperlessBilling	是否开通电子账单
18	PaymentMethod	客户端支付方式
19	MonthlyCharges	月费用
20	TotalCharges	总费用
21	Churn	是否流失

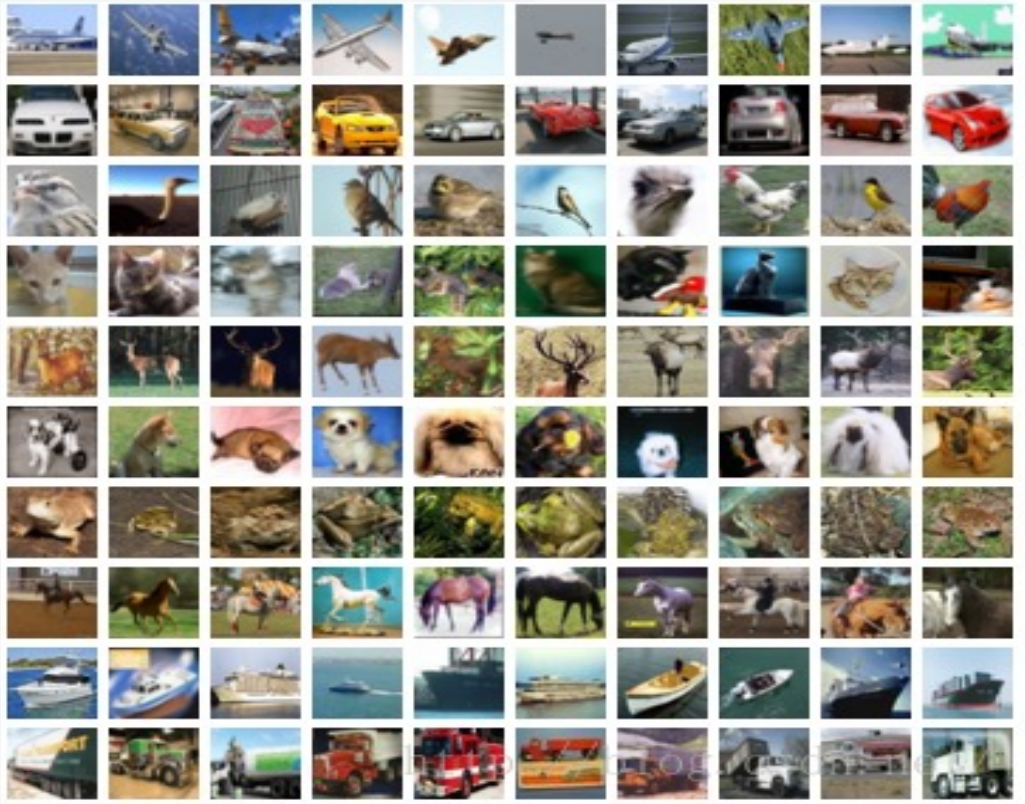
手写数字识别数据集

- Mnist：数字0～9
  - 训练集：60,000 个样本
  - 测试集：10000个样本

- 每个样本： 28*28的灰度图像，即784维的向量

CIFAR-10图像分类数据集

- CIFAR-10数据集是由Alex Krizhevsky, Vinod Nair和Geoffrey Hinton收集的一个用于识别普适物体的小型数据集。共包含10个类别的RGB彩色图片，如飞机、汽车、鸟类、猫、鹿、狗、蛙类、马、船和卡车。
- CIFAR-10数据集包含60000张32x32的彩色图像，每类包含6000张图片，其中50000张作为训练集，10000张作为测试集。
- CIFAR-10数据集的每张图片是以被展开的形式存储，每一类的数据表示为uint8，前1024个数据表示红色通道，接下来的1024个数据表示绿色通道，最后的1024个通道表示蓝色通道。



课后思考题（不需要提交）

- 学完一些章节后，会留一下课后思考题，供大家练习，检验学习效果。

5. 随机森林更适合采用哪种决策树？
   - A. 性能好，深度较深
   - B. 性能弱，深度较浅

6. 基于树的Boosting更适合采用哪种决策树？
   - A. 性能好，深度较深
   - B. 性能弱，深度较浅

7. 如果对决策树采用Bagging方式进行集成学习，更适合采用哪种方法对决策树的超参（如，树的深度）进行调优？
   - A. 交叉验证
   - B. 包外估计

竞赛网站

- Kaggle：https://www.kaggle.com/competitions
- 天池： https://tianchi.aliyun.com/competition/gameList/activeList
- 飞桨：https://aistudio.baidu.com/aistudio
- 讯飞：http://challenge.xfyun.cn/
- 数据科学竞赛创新平台： https://www.datafountain.cn/
- 和鲸：https://www.heywhale.com/home/competition
- DataCastle：https://challenge.datacastle.cn/v3/cmptlist.html
- 会议、协会、企业、...
  - CCF、中国人工智能学会、阿里、腾讯、京东、...

总结

- 机器学习项目开发的一般流程
- 第一个机器学习算法：KNN
- 不同机器学习算法的共同部分：
  - 数据预处理 & 特征工程
  - 模型性能评价指标
  - 超参数调优
- 开发环境配置
- 工具包简介