



大数据，成就未来



# 使用scikit-learn构建模型

2019/12/28

# 使用scikit-learn构建模型

---

## 本章学习目标

- ( 1 ) 掌握sklearn转换器、估计器的使用
  - ( 2 ) 掌握sklearn数据标准化与数据划分
  - ( 3 ) 掌握sklearn中聚类、分类、回归模型的构建
  - ( 4 ) 掌握sklearn中聚类、分类、回归模型的评价。
- 
- scikit-learn ( 简称sklearn ) 库整合了多种机器学习的算法，可以帮助使用者在数据分析过程中快速建立模型，且模型接口统一，使用起来非常方便。
  - 同时，sklearn拥有优秀的官方文档 ( <http://scikit-learn.org/stable/> )，知识点详尽，内容丰富。

# sklearn简介 (补充)



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Custom Search 



## scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ... — Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.  
**Algorithms:** SVR, ridge regression, Lasso, ... — Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes  
**Algorithms:** k-Means, spectral clustering, mean-shift, ... — Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency  
**Algorithms:** PCA, feature selection, non-negative matrix factorization. — Examples

## Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning  
**Modules:** grid search, cross validation, metrics. — Examples

## Preprocessing

Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.  
**Modules:** preprocessing, feature extraction. — Examples

# sklearn简介 (补充)

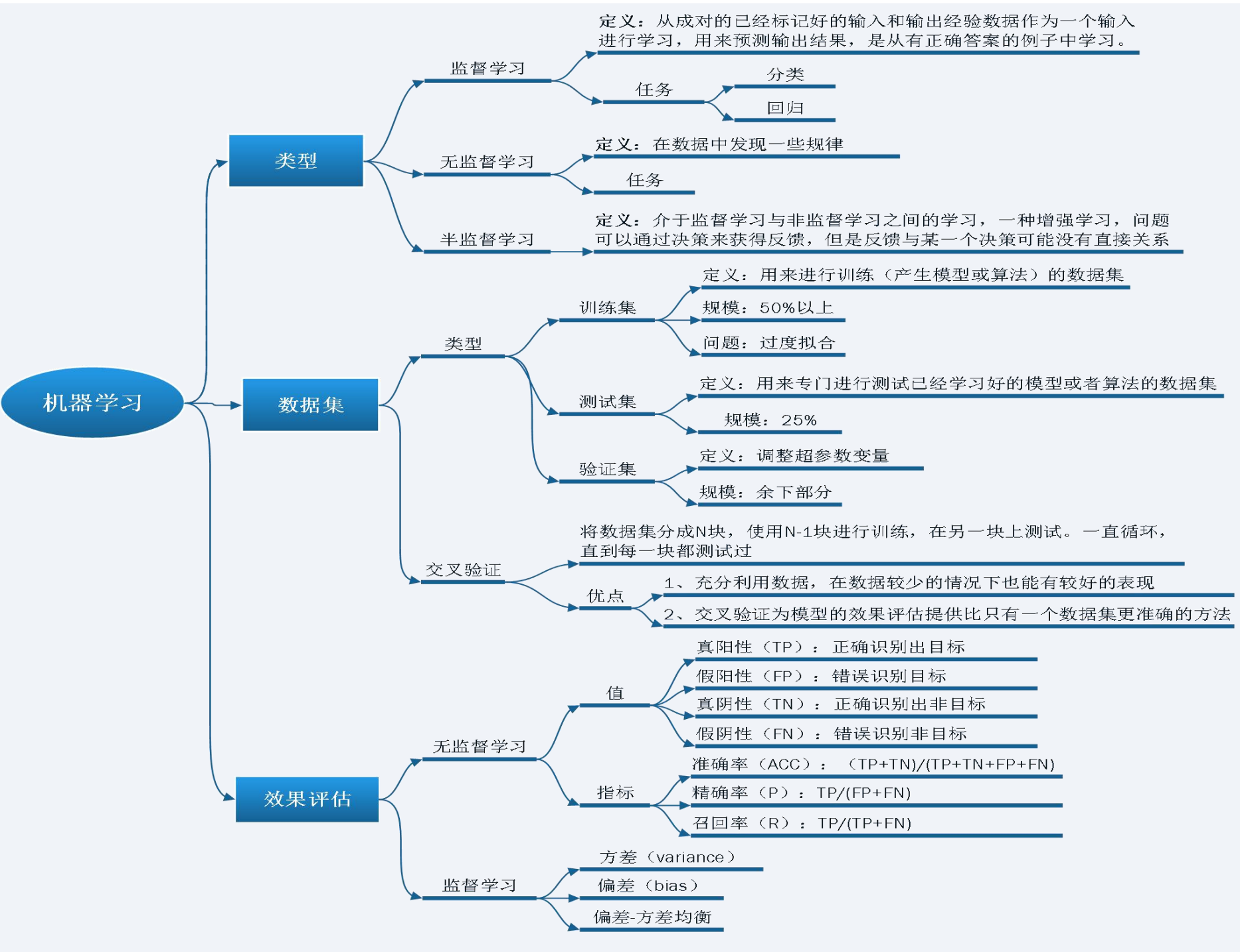
---

## 机器学习的认识

- 从实践的角度出发，机器学习要做的工作就是在已有的一个数据集上建立一个或者多个模型，然后对模型进行优化和评估。

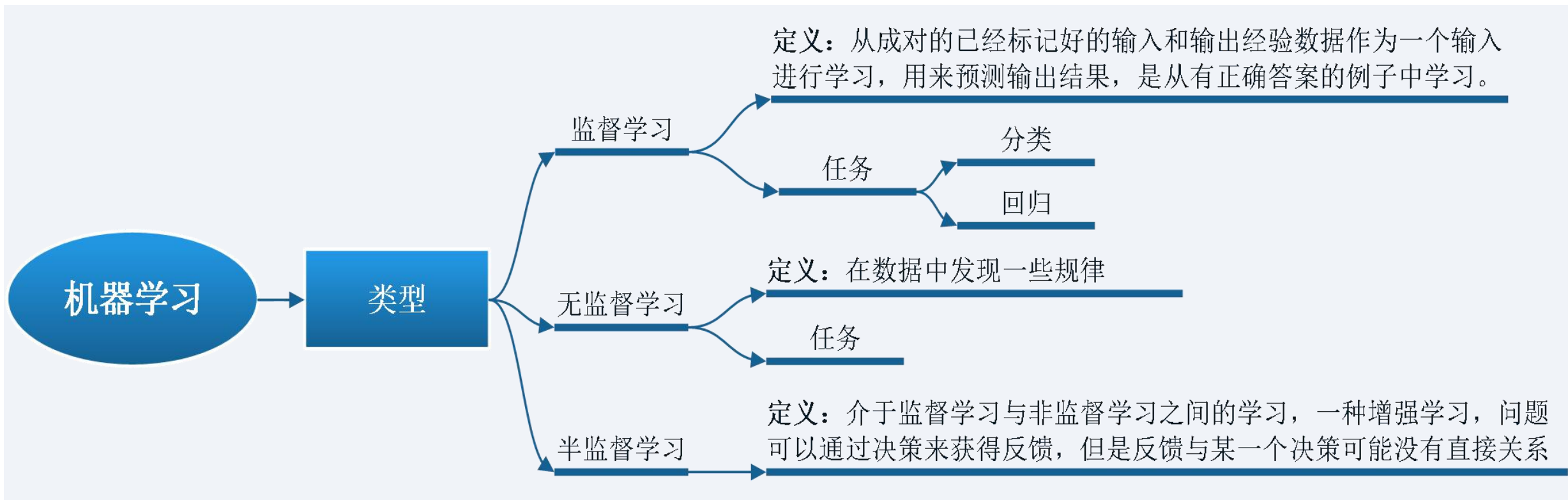
# sklearn简介

## 机器学习的认识



# sklearn简介 (补充)

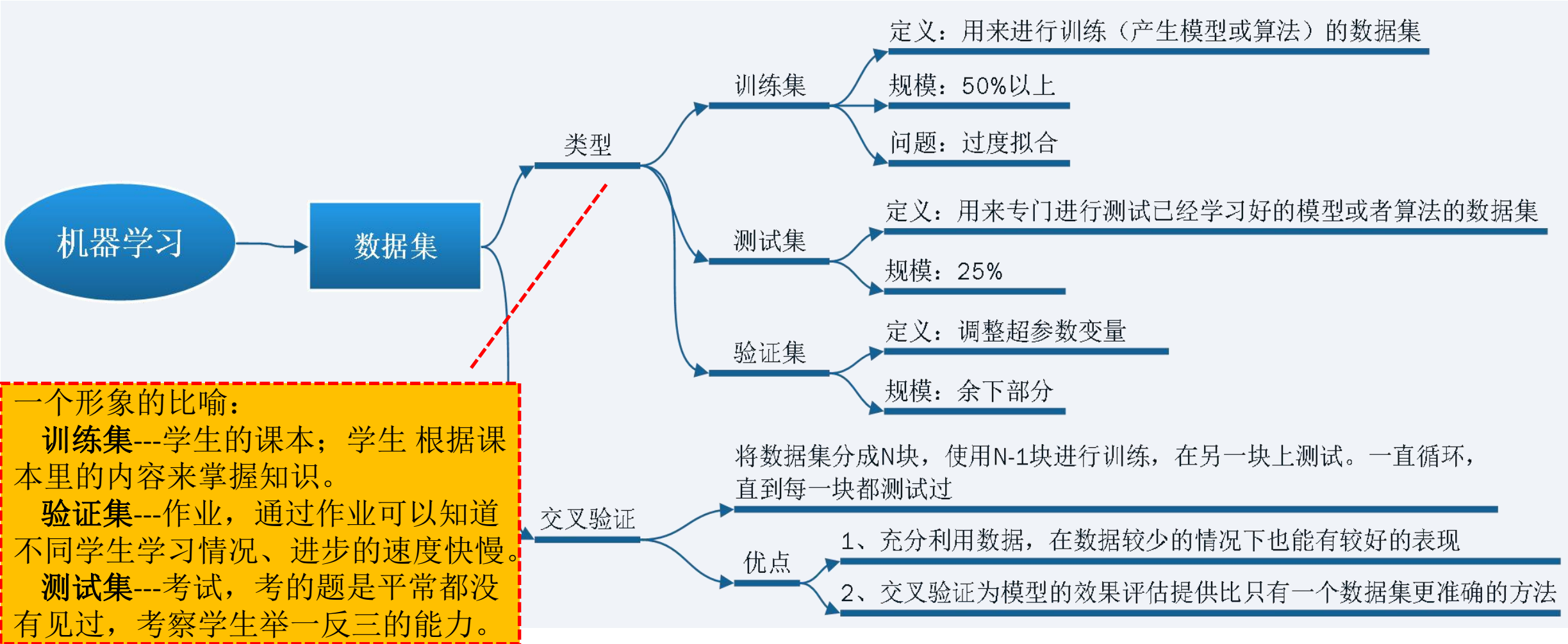
## 机器学习的认识





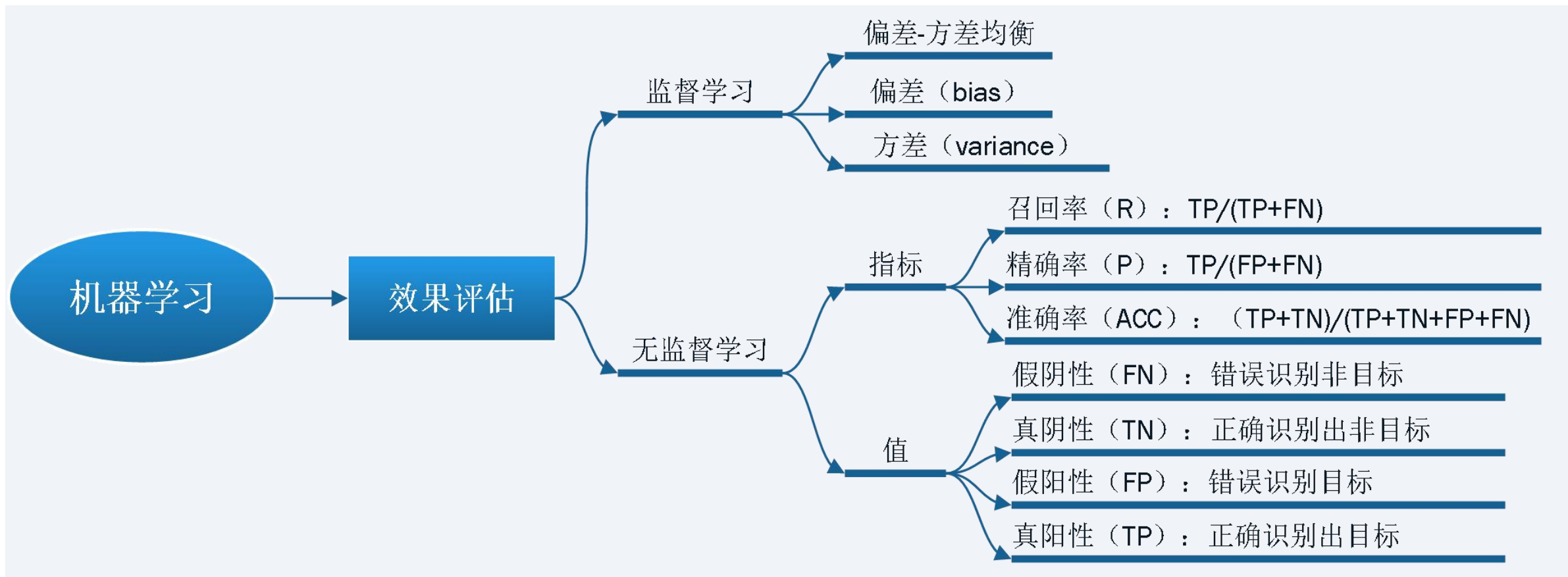
# sklearn简介 (补充)

## 机器学习的认识



# sklearn简介 (补充)

## 机器学习的认识





# sklearn简介（补充）

## sklearn库官方文档结构

- Tutorials：是一个官方教程，可以理解快速上手教程
- User guide（用户指南）：对每一个算法的详细介绍
- API：库调用的方法
- FAQ：常见问题
- contributing：贡献，还介绍最新的一些代码、功能

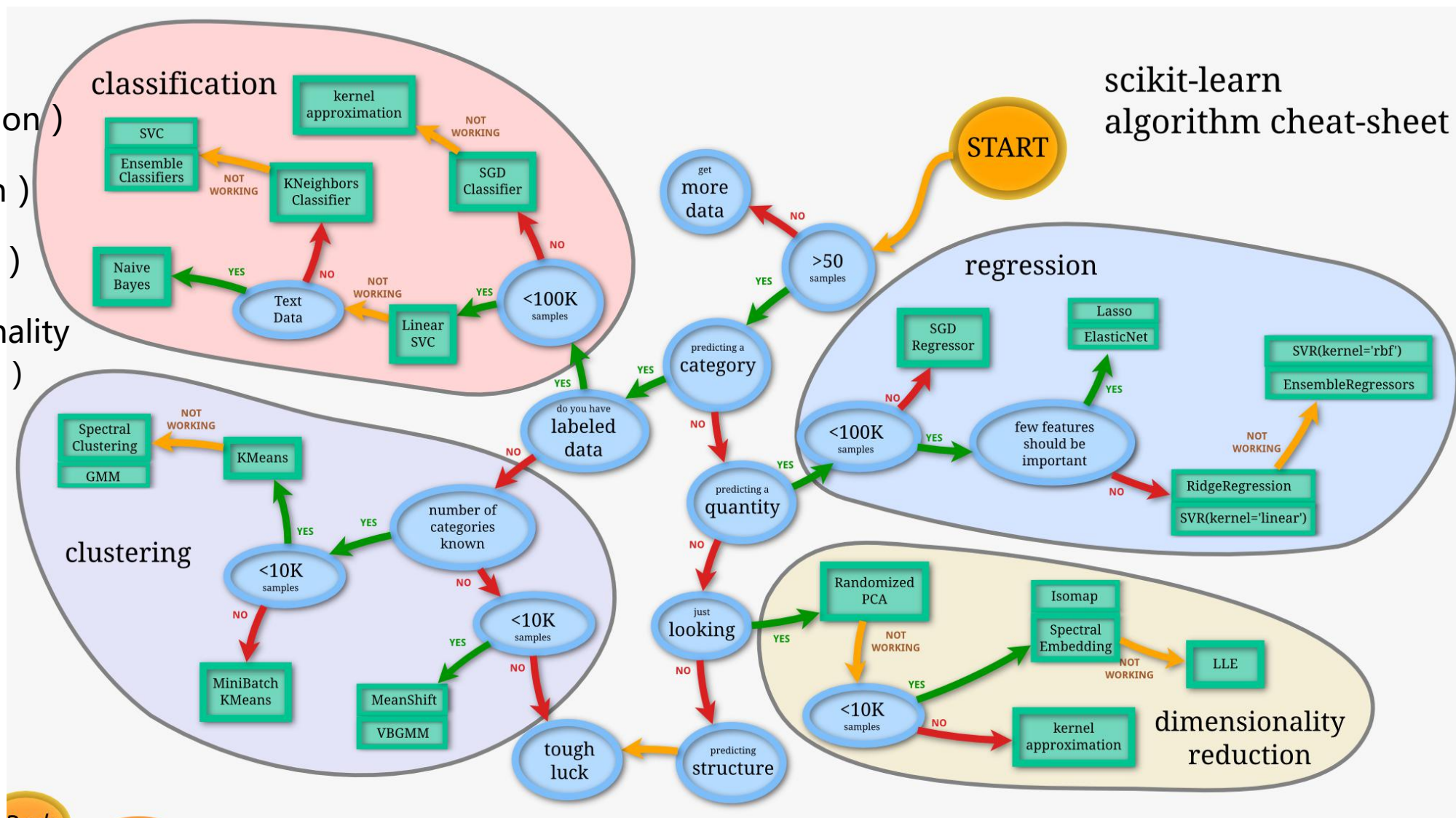
- 总结：一般的做法是API里面找到你要调用的方法，然后可以查看方法参数的情况和使用情况。也可以在指南里面找到具体的解释。



# sklearn简介 (补充)

# sklearn库结构

- 分类 ( classification )
- 回归 ( regression )
- 聚类 ( clustering )
- 降维 ( dimensionality reduction )



# sklearn简介（补充）

---

## 机器学习主要步骤中sklearn库应用

1. 数据集：面对自己的任务肯定有自己的数据集，但是对于初学者来说，sklearn提供了一些数据
  - ▣ 小规模数据集：数据包含在 datasets 里 ( datasets.load\_\*() )
  - ▣ 大规模数据集：需要从网络上下载 ( datasets.fetch\_\*() )
  - ▣ 本地生成数据集 ( datasets.make\_\*() )
2. 数据预处理：数据预处理包括：降维、数据归一化、特征提取和特征转换 ( one-hot ) 等，在sklearn里有很多方法，具体查看API。
3. 选择模型并训练：sklearn里面有很多的机器学习方法，可以查看API找到你需要的方法，sklearn统一了所有模型调用的API，使用起来比较简单。
4. 模型评分
5. 模型的保存与恢复

# sklearn简介（补充）

---

## 机器学习主要步骤中sklearn库应用

1. 数据集：面对自己的任务肯定有自己的数据集，但是对于初学者来说，sklearn提供了一些数据
  - ▣ 小规模数据集：数据包含在 datasets 里 ( datasets.load\_\*() )
  - ▣ 大规模数据集：需要从网络上下载 ( datasets.fetch\_\*() )
  - ▣ 本地生成数据集 ( datasets.make\_\*() )
2. 数据预处理：数据预处理包括：降维、数据归一化、特征提取和特征转换 ( one-hot ) 等，在sklearn里有很多方法，具体查看API。
3. 选择模型并训练：sklearn里面有很多的机器学习方法，可以查看API找到你需要的方法，sklearn统一了所有模型调用的API，使用起来比较简单。
4. 模型评分
5. 模型的保存与恢复

# 目录

---

1	使用sklearn转换器处理数据
2	构建并评价聚类模型
3	构建并评价分类模型
4	构建并评价回归模型
5	小结

# 使用scikit-learn转换器处理数据

---

## 任务描述

- sklearn提供了**model\_selection**模型选择模块、**preprocessing**数据预处理模块与**decomposition**特征分解模块。
- 通过这三个模块能够实现数据的预处理与模型构建前的数据标准化、二值化、数据集的分割、交叉验证和PCA降维等工作。

## 任务分析

- ( 1 ) 掌握sklearn自带数据集的加载函数与任务类型
- ( 2 ) 将breast cancer数据集划分为训练集和测试集
- ( 3 ) 使用转换器进行数据预处理与降维

# 加载datasets模块中数据集

---

## datasets模块常用数据集加载函数及其解释

- sklearn库的datasets模块集成了部分数据分析的经典数据集，可以使用这些数据集进行数据预处理，建模等操作，熟悉sklearn的数据处理流程和建模流程。
- sklearn.datasets模块主要提供了一些导入（load）、在线下载（fetch）及本地生成（make）数据集的方法，可以通过dir或help命令查看，我们会发现主要有三种形式：
  - ❑ load\_<dataset\_name>
  - ❑ fetch\_<dataset\_name>
  - ❑ make\_<dataset\_name>



# 加载datasets模块中数据集










## datasets模块常用数据集加载函数及其解释

➤ **load\_<dataset\_name>**

```
In [1]: from sklearn import datasets
```

```
In [2]: datasets.load_*?  
datasets.load_boston  
datasets.load_breast_cancer  
datasets.load_diabetes  
datasets.load_digits  
datasets.load_files  
datasets.load_iris  
datasets.load_linnerud  
datasets.load_mlcomp  
datasets.load_sample_image  
datasets.load_sample_images  
datasets.load_svmlight_file  
datasets.load_svmlight_files  
datasets.load_wine
```

Anaconda3 ▸ Lib ▸ site-packages ▸ sklearn ▸ datasets ▸ data

名称	修改日期	类型	大小
 boston_house_prices.csv	2017/10/23 22:02	XLS 工作表	35 KB
 breast_cancer.csv	2017/10/23 22:02	XLS 工作表	118 KB
 diabetes_data.csv.gz	2017/10/23 22:02	WinRAR 压缩文件	24 KB
 diabetes_target.csv.gz	2017/10/23 22:02	WinRAR 压缩文件	2 KB
 digits.csv.gz	2017/10/23 22:02	WinRAR 压缩文件	57 KB
 iris.csv	2017/10/23 22:02	XLS 工作表	3 KB
 linnerud_exercise.csv	2017/10/23 22:02	XLS 工作表	1 KB
 linnerud_physiological.csv	2017/10/23 22:02	XLS 工作表	1 KB
 wine_data.csv	2017/10/23 22:02	XLS 工作表	12 KB

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

➤ **load\_**<dataset\_name>

- ❑ datasets.load\_boston : 波士顿房价数据集（回归）
- ❑ datasets.load\_breast\_cancer : 乳腺癌数据集（分类）
- ❑ datasets.load\_diabetes : 糖尿病数据集（回归）
- ❑ datasets.load\_digits : 手写数字数据集（分类）
- ❑ datasets.load\_files : 加载带有类别的文本文件作为子文件夹名称。
- ❑ datasets.load\_iris : 鸢尾花数据集（分类）
- ❑ datasets.load\_linnerud : 体能训练数据集

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

➤ **load\_**<dataset\_name>

- ❑ datasets.load\_mlcomp : 被弃用了
- ❑ datasets.load\_sample\_image : 加载单个样本图像的numpy数组
- ❑ datasets.load\_sample\_images : 加载样本图像以进行图像处理。
- ❑ datasets.load\_svmlight\_file : 将svmlight / libsvm格式的数据集加载到稀疏CSR矩阵中
- ❑ datasets.load\_svmlight\_files : 以SVMLight格式从多个文件加载数据集
- ❑ datasets.load\_wine : 葡萄酒数据集（分类）

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

➤ **fetch\_<dataset\_name>**

```
In [3]: datasets.fetch_*?  
datasets.fetch_20newsgroups  
datasets.fetch_20newsgroups_vectorized  
datasets.fetch_california_housing  
datasets.fetch_covtype  
datasets.fetch_kddcup99  
datasets.fetch_lfw_pairs  
datasets.fetch_lfw_people  
datasets.fetch_mldata  
datasets.fetch_olivetti_faces  
datasets.fetch_rcv1  
datasets.fetch_species_distributions
```

<code>datasets.fetch_20newsgroups_vectorized ([...])</code>	Load the 20 newsgroups data (classification).
<code>datasets.fetch_california_housing ([...])</code>	Load the California housing data.
<code>datasets.fetch_covtype ([data_home, ...])</code>	Load the covtype dataset.
<code>datasets.fetch_kddcup99 ([subset, data_home, ...])</code>	Load the kddcup99 dataset.
<code>datasets.fetch_lfw_pairs ([subset, ...])</code>	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
<code>datasets.fetch_lfw_people ([data_home, ...])</code>	Load the Labeled Faces in the Wild (LFW) dataset (classification).
<code>datasets.fetch_olivetti_faces ([data_home, ...])</code>	Load the Olivetti faces dataset.
<code>datasets.fetch_openml ([name, version, ...])</code>	Fetch dataset from openml.
<code>datasets.fetch_rcv1 ([data_home, subset, ...])</code>	Load the RCV1 multilabel dataset.
<code>datasets.fetch_species_distributions ([...])</code>	Loader for species distribution data.

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

### ➤ **fetch\_**<dataset\_name>

- ❑ datasets.fetch\_20newsgroups : 20个新闻组数据集（分类）
- ❑ datasets.fetch\_20newsgroups\_vectorized : 20个新闻组数据集并将其矢量化为令牌计数（分类）
- ❑ datasets.fetch\_california\_housing : 加州住房数据集（回归）
- ❑ datasets.fetch\_covtype : 隐色数据集（分类）
- ❑ datasets.fetch\_kddcup99 : kddcup99数据集（分类）
- ❑ datasets.fetch\_lfw\_pairs : 加载标签面在野外（LFW）对数据集（分类）
- ❑ datasets.fetch\_lfw\_people : 带标签的人脸数据集（分类）
- ❑ datasets.fetch\_mldata : machine learning data set repository
- ❑ datasets.fetch\_olivetti\_faces : Olivetti脸部图像数据集（分类）
- ❑ datasets.fetch\_rcv1 : 路透社新闻语料数据集
- ❑ datasets.fetch\_species\_distributions : 物种分布数据集



# 加载datasets模块中数据集

scikit-learn

Install User Guide API Examples More

Prev Up Next

scikit-learn 0.22  
Other versions

Please cite us if you use the software.

sklearn.datasets.fetch\_covtype

sklearn.datasets.fetch\_covtype

```
sklearn.datasets.fetch_covtype(data_home=None, download_if_missing=True, random_state=None, shuffle=False, return_X_y=False)
```

[source]

Load the covtype dataset (classification).

Download it if necessary.

Classes	7
Samples total	581012
Dimensionality	54
Features	int

Read more in the [User Guide](#).

Parameters:

**data\_home : string, optional**  
Specify another download and cache folder for the datasets. By default all scikit-learn data is stored in '~/.scikit\_learn\_data' subfolders.

**download\_if\_missing : boolean, default=True**  
If False, raise a IOError if the data is not locally available instead of trying to download the data from the

# 加载datasets模块中数据集

## 5.3.4. Forest covertypes

The samples in this dataset correspond to 30×30m patches of forest in the US, collected for the task of predicting each patch's cover type, i.e. the dominant species of tree. There are ~~seven covertypes~~ [dataset's homepage](#), making this a multiclass classification problem. Each sample has 54 features, described on the [dataset's homepage](#). Some of the features are boolean indicators, while others are discrete or continuous measurements.

### Data Set Characteristics:

Classes	7
Samples total	581012
Dimensionality	54
Features	int

`sklearn.datasets.fetch_covtype` will load the covtype dataset; it returns a dictionary-like object with the feature matrix in the `data` member and the target values in `target`. The dataset will be downloaded from the web if necessary.



# 加载datasets模块中数据集



## Covertypes Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Forest CoverType dataset



Data Set Characteristics:	Multivariate	Number of Instances:	581012	Area:	Life
Attribute Characteristics:	Categorical, Integer	Number of Attributes:	54	Date Donated	1998-08-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	199164

### Source:

Original Owners of Database:

Remote Sensing and GIS Program  
Department of Forest Sciences  
College of Natural Resources  
Colorado State University  
Fort Collins, CO 80523

(contact Jock A. Blackard, [jblackard@fs.fed.us](mailto:jblackard@fs.fed.us) or Dr. Denis J. Dean, [denis.dean@utdallas.edu](mailto:denis.dean@utdallas.edu))

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

➤ **make\_<dataset\_name>**

[python]

```
1. In [4]: datasets.make_*?  
2. datasets.make_biclusters  
3. datasets.make_blobs  
4. datasets.make_checkerboard  
5. datasets.make_circles  
6. datasets.make_classification  
7. datasets.make_friedman1  
8. datasets.make_friedman2  
9. datasets.make_friedman3  
10. datasets.make_gaussian_quantiles  
11. datasets.make_hastie_10_2  
12. datasets.make_low_rank_matrix  
13. datasets.make_moons  
14. datasets.make_multilabel_classification  
15. datasets.make_regression  
16. datasets.make_s_curve  
17. datasets.make_sparse_coded_signal  
18. datasets.make_sparse_spd_matrix  
19. datasets.make_sparse_uncorrelated  
20. datasets.make_spd_matrix  
21. datasets.make_swiss_roll
```

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

➤ datasets模块常用数据集的加载函数与解释如下表所示。

数据集加载函数	数据集任务类型	数据集加载函数	数据集任务类型
load_boston	回归	load_breast_cancer	分类，聚类
fetch_california_housing	回归	load_iris	分类，聚类
load_digits	分类	load_wine	分类

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()    ##将数据集赋值给cancer变量
print('breast_cancer数据集的长度为：',len(cancer))
print('breast_cancer数据集的类型为：',type(cancer))
```

```
breast_cancer数据集的长度为： 5
breast_cancer数据集的类型为： <class 'sklearn.utils.Bunch'>
```

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature\_names，DESCR分别获取数据集的数据，标签，特征名称和描述信息。

```
cancer_data = cancer['data']
```

```
print('breast_cancer数据集的数据为：','\n',cancer_data)
```

```
breast_cancer数据集的数据为：
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

## 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature\_names，DESCR分别获取数据集的数据，标签，特征名称和描述信息。

**cancer\_target = cancer['target']**      **# 取出数据集的标签**

```
print('breast_cancer数据集的标签为：\n',cancer_target)
```

breast cancer数据集的标签为:

[illegible]



# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature\_names，DESCR分别获取数据集的数据，标签，特征名称和描述信息。

```
cancer_names = cancer['feature_names']      ## 取出数据集的特征名
```

```
print('breast_cancer数据集的特征名为：\n',cancer_names)
```

```
breast_cancer数据集的特征名为：
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'  
'mean smoothness' 'mean compactness' 'mean concavity'  
'mean concave points' 'mean symmetry' 'mean fractal dimension'  
'radius error' 'texture error' 'perimeter error' 'area error'  
'smoothness error' 'compactness error' 'concavity error'  
'concave points error' 'symmetry error' 'fractal dimension error'  
'worst radius' 'worst texture' 'worst perimeter' 'worst area'  
'worst smoothness' 'worst compactness' 'worst concavity'  
'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

# 加载datasets模块中数据集

## datasets模块常用数据集加载函数及其解释

- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature\_names，DESCR分别获取数据集的数据，标签，特征名称和描述信息。

```
cancer_desc = cancer['DESCR']          ## 取出数据集的描述信息
```

```
print('breast_cancer数据集的描述信息为：\n',cancer_desc)
```

```
breast_cancer数据集的描述信息为：
Breast Cancer Wisconsin (Diagnostic) Database
=====

Notes
-----
Data Set Characteristics:
 :Number of Instances: 569

 :Number of Attributes: 30 numeric, predictive attributes and the class

 :Attribute Information:
   - radius (mean of distances from center to points on the perimeter)
   - texture (standard deviation of gray-scale values)
```



# 将数据集划分为训练集和测试集

---

## 常用划分方式

- 在数据分析过程中，为了保证模型在实际系统中能够起到预期作用，一般需要将样本分成独立的三部分：
  - ▣ 训练集 ( train set )：用于估计模型。
  - ▣ 验证集 ( validation set)：用于确定网络结构或者控制模型复杂程度的参数。
  - ▣ 测试集 ( test set )：用于检验最优的模型的性能。
- 典型的划分方式是训练集占总样本的50%，而验证集和测试集各占25%。
- 当数据总量较少的时候，使用上面的方法将数据划分为三部分就不合适了。

# 将数据集划分为训练集和测试集

## K折交叉验证法

- 当数据总量较少的时候，使用上面的方法将数据划分为三部分就不合适了。
- 常用的方法是留少部分做测试集，然后对其余N个样本采用**K折交叉验证法**，基本步骤如下：
  - 将样本打乱，均匀分成K份。
  - 轮流选择其中K - 1份做训练，剩余的一份做验证。
  - 计算预测误差平方和，把K次的预测误差平方和的均值作为选择最优模型结构的依据。
- sklearn的model\_selection模块提供了train\_test\_split函数，能够对数据集进行拆分

# 将数据集划分为训练集和测试集

## train\_test\_split函数

➤ sklearn的model\_selection模块提供了train\_test\_split函数，能够对数据集进行拆分，其格式如下

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

参数名称	说明
*arrays	接收一个或多个数据集。代表需要划分的数据集，若为分类回归则分别传入数据和标签，若为聚类则传入数据。无默认。
test_size	接收float，int，None类型的数据。代表测试集的大小。如果传入的为float类型的数据则需要限定在0-1之间，代表测试集在总数中的占比；如果传入为int类型的数据，则表示测试集记录的绝对数目。该参数与train_size可以只传入一个。在0.21版本前，若test_size和train_size均为默认则testsize为25%。
train_size	接收float，int，None类型的数据。代表训练集的大小。该参数与test_size可以只传入一个。
random_state	接收int。代表随机种子编号，相同随机种子编号产生相同的随机结果，不同的随机种子编号产生不同的随机结果。默认为None。
shuffle	接收boolean。代表是否进行有放回抽样。若该参数取值为True则stratify参数必须不能为空。
stratify	接收array或者None。如果不为None，则使用传入的标签进行分层抽样。

# 将数据集划分为训练集和测试集

---

## train\_test\_split函数

- train\_test\_split函数根据传入的数据，分别将传入的数据划分为训练集和测试集。
  - 如果传入的是1组数据，那么生成的就是这一组数据随机划分后的训练集和测试集，总共2组。
  - 如果传入的是2组数据，则生成的训练集和测试集分别2组，总共4组。
- train\_test\_split是最常用的数据划分方法，在model\_selection模块中还提供了其他数据集划分的函数，如PredefinedSplit，ShuffleSplit等。

# 将数据集划分为训练集和测试集

## train\_test\_split函数

```
print('原始数据集数据的形状为：',cancer_data.shape)
```

```
print('原始数据集标签的形状为：',cancer_target.shape)
```

```
from sklearn.model_selection import train_test_split
```

```
cancer_data_train, cancer_data_test,cancer_target_train, cancer_target_test = \
    train_test_split(cancer_data, cancer_target, test_size=0.2, random_state=42)
```

```
print('训练集数据的形状为：',cancer_data_train.shape)
```

```
print('测试集数据的形状为：',cancer_data_test.shape)
```

```
print('训练集标签的形状为：',cancer_target_train.shape)
```

```
print('测试集标签的形状为：',cancer_target_test.shape)
```

训练集数据的形状为：	(455, 30)
测试集数据的形状为：	(114, 30)
训练集标签的形状为：	(455,)
测试集标签的形状为：	(114,)

# 使用sklearn转换器进行数据预处理与降维

---

## transformer

- 使用sklearn进行数据预处理会用到sklearn提供的统一接口——**转换器(transformer)**
- 为了帮助用户实现大量的特征处理相关操作，sklearn把相关的功能封装为转换器。
- 使用sklearn转换器能够实现对传入的NumPy数组进行：
  - ▣ 标准化处理，归一化处理，二值化处理，PCA降维等操作
- 转换器主要包括三个方法：fit、transform、fit\_transform

# 使用sklearn转换器进行数据预处理与降维

## transformer

- 转换器主要包括三个方法：fit、transform、fit\_transform

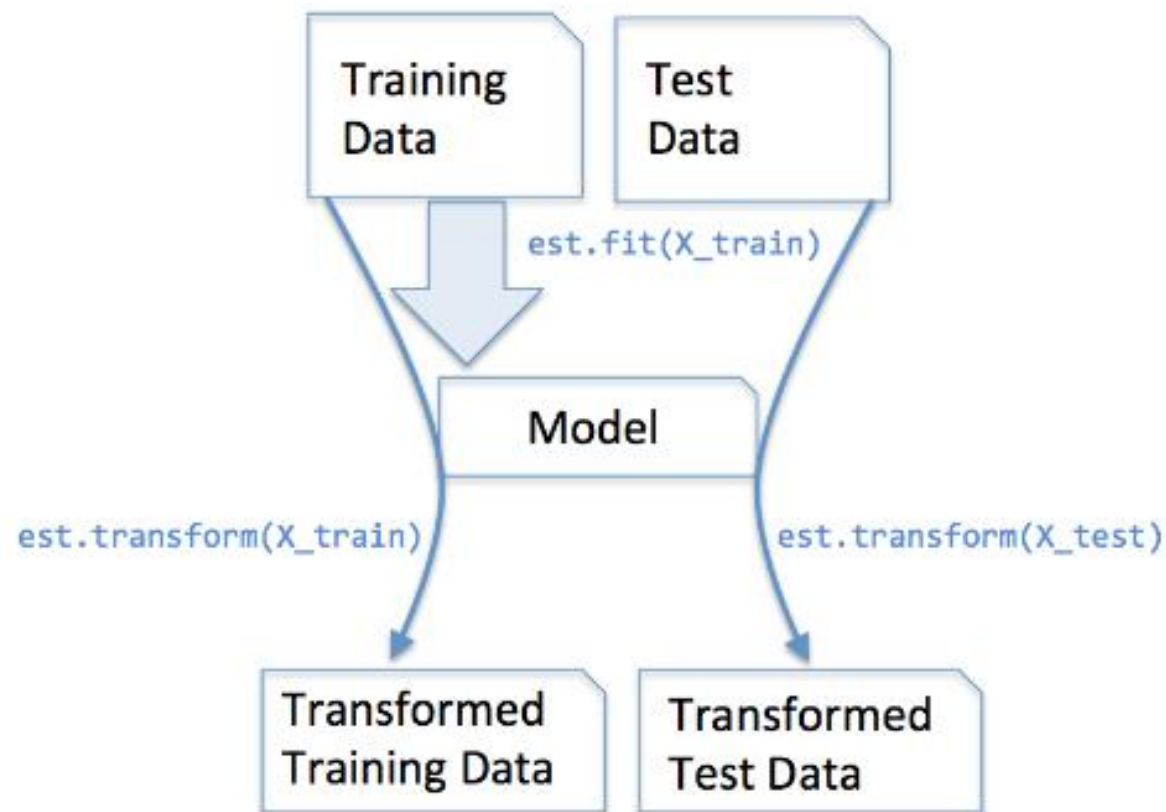
方法名称	说明
fit	fit方法主要通过分析特征和目标值，提取有价值的信息，这些信息可以是统计量，也可以是权值系数等。
transform	transform方法主要用来对特征进行转换。从可利用信息的角度可分为无信息转换和有信息转换。无信息转换是指不利用任何其他信息进行转换，比如指数和对数函数转换等。有信息转换根据是否利用目标值向量又可分为无监督转换和有监督转换。无监督转换指只利用特征的统计信息的转换，比如标准化和PCA降维等。有监督转换指既利用了特征信息又利用了目标值信息的转换，比如通过模型选择特征和LDA降维等。
fit_transform	fit_transform方法就是先调用fit方法，然后调用transform方法。



# 使用sklearn转换器进行数据预处理与降维

## transformer

- `fit()` : 用于从训练数据生成学习模型参数
- `transform()` : 从`fit()`方法生成的参数, 应用于模型以生成转换数据集。
- `fit_transform()` : 在同一数据集上组合`fit()`和`transform()`api



# 使用sklearn转换器进行数据预处理与降维

## sklearn转换器

- 在数据分析过程中，各类特征处理相关的操作都需要对训练集和测试集分开操作，需要将训练集的操作规则，权重系数等应用到测试集中。
- 在第五章介绍pandas的时候，知道使用pandas库进行标准化处理等，应用至测试集的过程相对烦琐，使用sklearn转换器可以解决这一困扰。

$$X^* = \frac{X - \min}{\max - \min}$$

```
import numpy as np
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
Scaler = MinMaxScaler().fit(cancer_data_train)
```

```
cancer_trainScaler = Scaler.transform(cancer_data_train)
```

```
cancer_testScaler = Scaler.transform(cancer_data_test)
```

#离差标准化

#生成规则

#将规则应用于训练集

#将规则应用于测试集

# 使用sklearn转换器进行数据预处理与降维

## sklearn转换器

```
print('离差标准化前训练集数据的最小值为:', np.min(cancer_data_train))
print('离差标准化后训练集数据的最小值为:', np.min(cancer_trainScaler))
print('离差标准化前训练集数据的最大值为:', np.max(cancer_data_train))
print('离差标准化后训练集数据的最大值为:', np.max(cancer_trainScaler))
print('离差标准化前测试集数据的最小值为:', np.min(cancer_data_test))
print('离差标准化后测试集数据的最小值为:', np.min(cancer_testScaler))
print('离差标准化前测试集数据的最大值为:', np.max(cancer_data_test))
print('离差标准化后测试集数据的最大值为:', np.max(cancer_testScaler))
```

0.0
0.0
4254.0
1.000000000000000002
0.0
-0.057127602776294695
3432.0
1.3264399566986453

- 通过代码运行发现，离差标准化之后的训练数据集数据的最小值、最大值的确限定在[0,1]区间，同时由于测试集应用了训练集的离差标准化规则，数据超出了[0,1]的范围，这也侧面证明了此处应用了训练集的规则，如果两个数据集单独做离差标准化，或者两个数据集合并做离差标准化，根据公式则取值范围还是会限定在[0,1]区间。

# 使用sklearn转换器进行数据预处理与降维

## sklearn部分预处理函数与其作用

- sklearn除了提供离差标准化函数MinMaxScaler外，还提供了一系列数据预处理函数，具体如下表：

函数名称	说明
MinMaxScaler	对特征进行离差标准化。
StandardScaler	对特征进行标准差标准化。
Normalizer	对特征进行归一化。
Binarizer	对定量特征进行二值化处理。
OneHotEncoder	对定性特征进行独热编码处理。
FunctionTransformer	对特征进行自定义函数变换。

- 更多参考sklearn官网 <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing>

# 使用sklearn转换器进行数据预处理与降维

## PCA降维算法函数（主成分分析）

- sklearn除了提供基本的特征变换函数外，还提供了降维算法，特征选择算法，这些算法的使用也是通过转换器的方式进行的。

```
from sklearn.decomposition import PCA
```

```
pca_model = PCA(n_components=10).fit(cancer_trainScaler) #生成规则
```

```
cancer_trainPca = pca_model.transform(cancer_trainScaler) #将规则应用于训练集
```

```
cancer_testPca = pca_model.transform(cancer_testScaler) #将规则应用于测试集
```

```
print('PCA降维前训练集数据的形状为：',cancer_trainScaler.shape)
```

```
print('PCA降维后训练集数据的形状为：',cancer_trainPca.shape)
```

```
print('PCA降维前测试集数据的形状为：',cancer_testScaler.shape)
```

```
print('PCA降维后测试集数据的形状为：',cancer_testPca.shape)
```

# 使用sklearn转换器进行数据预处理与降维

## PCA降维算法函数（主成分分析）

- sklearn除了提供基本的特征变换函数外，还提供了降维算法，特征选择算法，这些算法的使用也是通过转换器的方式进行的。

```
from sklearn.decomposition import PCA
```

```
pca_model = PCA(n_components=10).fit(cancer_trainScaler) #生成规则
```

```
cancer_trainPca = pca_model.transform(cancer_trainScaler) #将规则应用于训练集
```

```
cancer_testPca = pca_model.transform(cancer_testScaler) #将规则应用于测试集
```

```
print('PCA降维前训练集数据的形状为：', PCA降维前训练集数据的形状为： (455, 30))
print('PCA降维后训练集数据的形状为：', PCA降维后训练集数据的形状为： (455, 10))
print('PCA降维前测试集数据的形状为：', PCA降维前测试集数据的形状为： (114, 30))
print('PCA降维后测试集数据的形状为：', PCA降维后测试集数据的形状为： (114, 10))
```



# 使用sklearn转换器进行数据预处理与降维

## PCA降维算法函数常用参数及其作用

函数名称	说明
n_components	接收None，int，float或string。未指定时，代表所有特征均会被保留下来；如果为int，则表示将原始数据降低到n个维度；如果为float，同时svd_solver参数等于full；赋值为string，比如n_components='mle'，将自动选取特征个数n，使得满足所要求的方差百分比。默认为None。
copy	接收bool。代表是否在运行算法时将原始数据复制一份，如果为True，则运行后，原始数据的值不会有任何改变；如果为False，则运行PCA算法后，原始训练数据的值会发生改变。默认为True
whiten	接收boolean。表示白化，所谓白化，就是对降维后的数据的每个特征进行归一化，让方差都为1。默认为False。
svd_solver	接收string { 'auto'， 'full'， 'arpack'， 'randomized' }。代表使用的SVD算法。randomized一般适用于数据量大，数据维度多，同时主成分数目比例又较低的PCA降维，它使用了一些加快SVD的随机算法。full是使用SciPy库实现的传统SVD算法。arpack和randomized的适用场景类似，区别是randomized使用的是sklearn自己的SVD实现，而arpack直接使用了SciPy库的sparse SVD实现。auto则代表PCA类会自动在上述三种算法中去权衡，选择一个合适的SVD算法来降维。默认为auto。

# sklearn转换器任务实现

---

## 1.读取数据

- 获取sklearn自带的boston数据集

## 2.将数据集划分为训练集和测试集

- 使用train\_test\_split划分boston数据集

## 3.使用转换器进行数据预处理

- 使用stdScale.transform分别对训练集和测试集进行预处理

## 4.使用转换器进行PCA降维

- 使用PCA.transform分别对训练集和测试集进行PCA降维

# sklearn转换器任务实现

## 1.读取数据

- 获取sklearn自带的boston数据集

```
from sklearn.datasets import load_boston
```

```
boston = load_boston()
```

```
boston_data = boston['data']
```

```
boston_target = boston['target']
```

```
boston_names = boston['feature_names']
```

```
print('boston数据集数据的形状为:', boston_data.shape)
```

```
print('boston数据集标签的形状为:', boston_target.shape)
```

```
print('boston数据集特征名的形状为:', boston_names.shape)
```

```
boston数据集数据的形状为: (506, 13)
```

```
boston数据集标签的形状为: (506,)
```

```
boston数据集特征名的形状为: (13,)
```

# sklearn转换器任务实现

## 1.读取数据

- 获取sklearn自带的boston数据集
  - CRIM - 城镇人均犯罪率
  - ZN - 住宅用地超过 25000 sq.ft. 的比例.
  - INDUS - 城镇非零售商用土地的比例.
  - CHAS - 查理斯河哑变量 ( 如果边界是河流 , 则为1 ; 否则为0 )
  - NOX - 一氧化氮浓度 (parts per 10 million)
  - RM - 住宅平均房间数
  - AGE - 1940 年之前建成的自用房屋比例
  - DIS - 到波士顿五个中心区域的加权距离
  - RAD - 辐射性公路的接近指数
  - TAX - 每 10000 美元的全值财产税率
  - PTRATIO - 城镇师生比例
  - B -  $1000(B_k - 0.63)^2$  , 其中  $B_k$  指代城镇中黑人的比例
  - LSTAT - 人口中地位低下者的比例
  - MEDV - 自住房的平均房价 , 以千美元计

# sklearn转换器任务实现

## 2.将数据集划分为训练集和测试集

- 使用train\_test\_split划分boston数据集

```
from sklearn.model_selection import train_test_split
boston_data_train, boston_data_test, boston_target_train, boston_target_test = \
train_test_split(boston_data, boston_target, test_size=0.2, random_state=42)
print('训练集数据的形状为：',boston_data_train.shape)
print('训练集标签的形状为：',boston_target_train.shape)
print('测试集数据的形状为：',boston_data_test.shape)
print('测试集标签的形状为：',boston_target_test.shape)
```

```
训练集数据的形状为： (404, 13)
训练集标签的形状为： (404,)
测试集数据的形状为： (102, 13)
测试集标签的形状为： (102,)
```

# sklearn转换器任务实现

## 3.使用转换器进行数据预处理

- 使用stdScale.transform分别对训练集和测试集进行预处理

```
import numpy as np
```

```
from sklearn.preprocessing import StandardScaler
```

```
stdScale = StandardScaler().fit(boston_data_train) ## 生成规则
```

```
boston_trainScaler = stdScale.transform(boston_data_train)    ## 将规则应用于训练集
```

```
boston_testScaler = stdScale.transform(boston_data_test)      ## 将规则应用于测试集
```

```
print('标准差标准化后训练集数据的方差为：',np.var(boston_trainScaler))
```

```
print('标准差标准化后训练集数据的均值为：',np.mean(boston_trainScaler))
```

```
print('标准差标准化后测试集数据的方差为：',np.var(boston_testScaler))
```

```
print('标准差标准化后测试集数据的均值为：',np.mean(boston_testScaler))
```

# sklearn转换器任务实现

## 4.使用转换器进行PCA降维

- 使用PCA.transform分别对训练集和测试集进行PCA降维

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=5).fit(boston_trainScaler) ## 生成规则
```

```
boston_trainPca = pca.transform(boston_trainScaler) ## 将规则应用于训练集
```

```
boston_testPca = pca.transform(boston_testScaler) ## 将规则应用于测试集
```

```
print('降维后boston数据集数据测试集的形状为:',boston_trainPca.shape)
```

```
print('降维后boston数据集数据训练集的形状为:',boston_testPca.shape)
```

```
降维后boston数据集数据测试集的形状为: (404, 5)
```

```
降维后boston数据集数据训练集的形状为: (102, 5)
```



# 目录

---



# 构建并评价聚类模型

---

## 任务描述

- **聚类分析**是在没有给定划分类别的情况下，根据数据相似度进行样本分组的一种方法。
- 聚类模型可以将无类标记的数据聚集为多个簇，视为一类，是一种无监督的学习算法。
- 在商业上，聚类可以帮助市场分析人员从消费者数据库中区分出不同的消费群体，并且概况出每一类消费者的消费模式或消费习惯。
- 同时，聚类分析也可以作为数据分析算法中其他分析算法的一个预处理步骤，比如异常值识别、连续性特征离散化等。

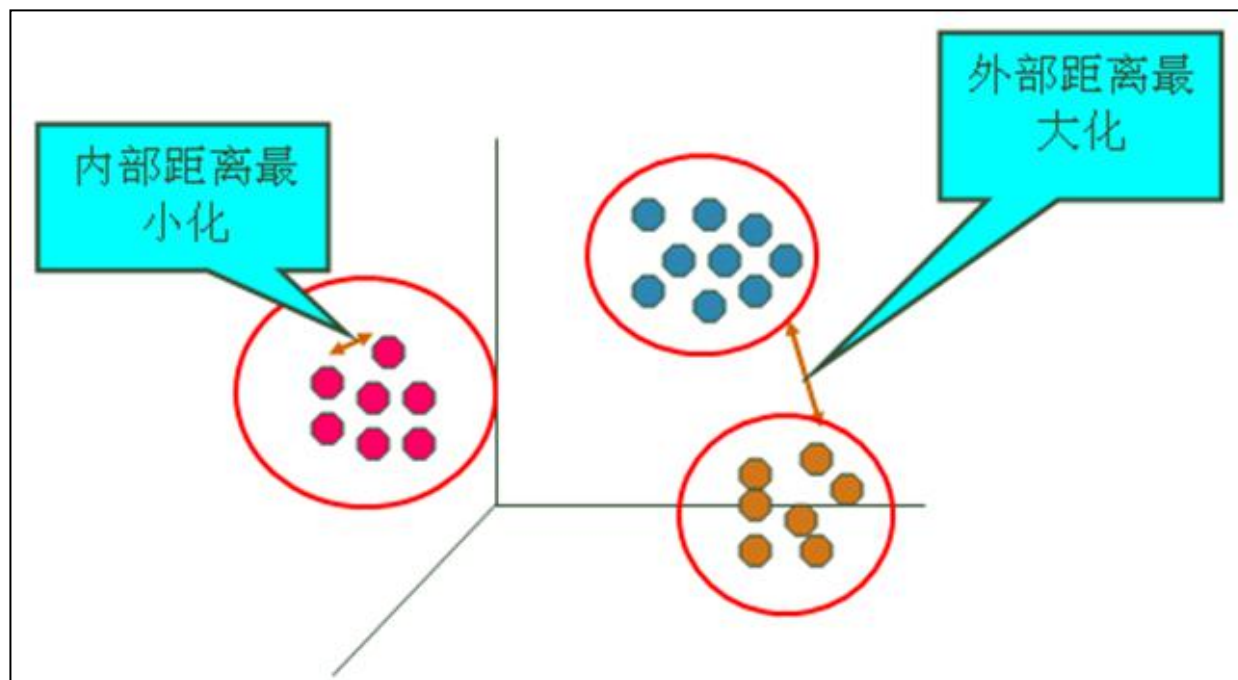
## 任务分析

- ( 1 ) 使用sklearn估计器构建K-Means聚类模型
- ( 2 ) 根据聚类模型评价指标评价K-Means聚类模型

# 使用sklearn估计器构建聚类模型

## 聚类

- 聚类的输入是一组未被标记的样本，聚类根据数据自身的距离或相似度将他们划分为若干组；
- 划分的原则是组内（内部）样本距离最小化，而组间（外部）距离最大化，如图所示。



# 使用sklearn估计器构建聚类模型

## 聚类方法类别

➤ 常用的聚类算法如表：

算法类别	包括的主要算法
划分（分裂）方法	K-Means算法（K-平均），K-MEDOIDS算法（K-中心点）和CLARANS算法（基于选择的算法）。
层次分析方法	BIRCH算法（平衡迭代规约和聚类），CURE算法（代表点聚类）和CHAMELEON算法（动态模型）。
基于密度的方法	DBSCAN算法（基于高密度连接区域），DENCLUE算法（密度分布函数）和OPTICS算法（对象排序识别）。
基于网格的方法	STING算法（统计信息网络），CLIOUE算法（聚类高维空间）和WAVE-CLUSTER算法（小波变换）。

# 使用sklearn估计器构建聚类模型

## cluster提供的聚类算法及其适用范围

➤ sklearn常用的聚类算法模块cluster提供的聚类算法及其适用范围如下所示：

函数名称	参数	适用范围(样本量、聚类数目)	距离度量
KMeans	簇数	样本量很大，聚类数目中等	点之间的距离
Spectral clustering	簇数	样本量中等，聚类数目较小。	图距离
Ward hierarchical clustering	簇数	样本量较大，聚类数目较大	点之间的距离
Agglomerative clustering	簇数，链接类型，距离	样本量较大，聚类数目较大	任意成对点线图间的距离
DBSCAN	半径大小，最低成员数目	样本量很大，聚类数目中等	最近的点之间的距离
Birch	分支因子，阈值， 可选全局集群	样本量很大，聚类数目较大	点之间的欧式距离

➤ 详情参考：<https://sklearn.org/modules/clustering.html#clustering>

# 使用sklearn估计器构建聚类模型

## sklearn估计器

- 聚类算法实现需要sklearn估计器（estimator）。sklearn估计器和转换器类似，拥有fit和predict两个方法。
- 两个方法的作用如下：

方法名称	说明
fit	fit方法主要用于训练算法。该方法可接收用于有监督学习的训练集及其标签两个参数，也可以接收用于无监督学习的数据。
predict	predict用于预测有监督学习的测试集标签，亦可以用于划分传入数据的类别。

# 使用sklearn估计器构建聚类模型

---

## sklearn估计器

- 以iris数据为例，使用sklearn估计器构建K-Means聚类模型，代码如下：

```
from sklearn.datasets import load_iris
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.cluster import KMeans
```

```
iris = load_iris()
```

```
iris_data = iris['data']           # 提取数据集中的特征
```

```
iris_target = iris['target']       # 提取数据集中的标签
```

```
iris_names = iris['feature_names'] # 提取特征名
```



# 使用sklearn估计器构建聚类模型

## sklearn估计器

```
scale = MinMaxScaler().fit(iris_data)           # 构建并训练标准化规则
iris_dataScale = scale.transform(iris_data)      # 应用规则
#构建并训练聚类模型(分3类)
kmeans = KMeans(n_clusters=3,random_state=123).fit(iris_dataScale)
print('构建的K-Means模型为 : \n',kmeans)
result = kmeans.predict([[1.5,1.5,1.5,1.5]]) #使用构建的模型进行预测
print('花瓣花萼长度宽度全为1.5的鸢尾花预测类别为 : ', result[0])
```

构建的K-Means模型为：

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
      n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
      random_state=123, tol=0.0001, verbose=0)
花瓣花萼长度宽度全为1.5的鸢尾花预测类别为： 0
```

# 使用sklearn估计器构建聚类模型

## TSNE函数

- 聚类完成后需要通过可视化的方式查看聚类效果，通过sklearn的manifold模块中的TSNE函数可以实现多维数据的可视化展现。其原理是使用TSNE进行数据降维，降成二维。

```
import pandas as pd
```

```
from sklearn.manifold import TSNE
```

```
import matplotlib.pyplot as plt
```

```
# 使用TSNE进行数据降维,降成二维
```

```
tsne = TSNE(n_components=2,init='random', random_state=177).fit(iris_data)
```

```
df=pd.DataFrame(tsne.embedding_)    #将原始数据转换为DataFrame
```

```
df['labels'] = kmeans.labels_      #将聚类结果存储进df数据表
```

# 使用sklearn估计器构建聚类模型

## TSNE函数

##提取不同标签的数据

```
df1 = df[df['labels']==0]
```

```
df2 = df[df['labels']==1]
```

```
df3 = df[df['labels']==2]
```

## 绘制图形

```
fig = plt.figure(figsize=(9,6))
```

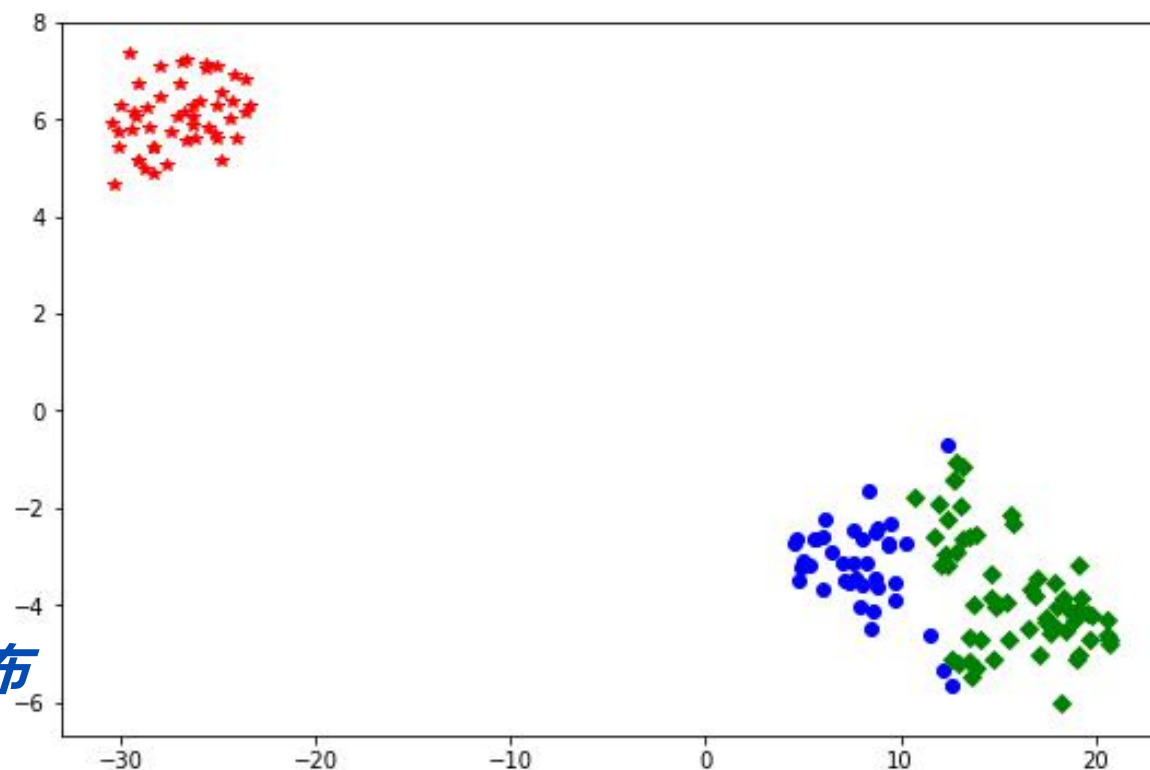
##用不同的颜色表示不同数据

```
plt.plot(df1[0],df1[1],'bo',df2[0],df2[1],'r*',df3[0],df3[1],'gD')
```

```
plt.savefig('tmp/聚类结果.png')
```

```
plt.show() ##显示图片
```

#设定画布



# 评价聚类模型

## 聚类模型评价指标

- 聚类评价的标准是组内的对象相互之间是相似的（相关的），而不同组中的对象是不同的（不相关的）。即组内的相似性越大，组间差别越大，聚类效果就越好。sklearn的metrics模块提供的聚类模型评价指标。

方法名称	真实值	最佳值	sklearn函数
ARI评价法（兰德系数）	需要	1.0	adjusted_rand_score
AMI评价法（互信息）	需要	1.0	adjusted_mutual_info_score
V-measure评分	需要	1.0	completeness_score
FMI评价法	需要	1.0	fowlkes_mallows_score
轮廓系数评价法	不需要	畸变程度最大	silhouette_score
Calinski-Harabasz指数评价法	不需要	相较最大	calinski_harabaz_score

# 评价聚类模型

## 聚类模型评价指标

- 上表总共列出了6种评价的方法，其中前4种方法均需要真实值的配合才能够评价聚类算法的优劣，后2种则不需要真实值的配合。但是前4种方法评价的效果更具有说服力，并且在实际运行的过程中在有真实值做参考的情况下，聚类方法的评价可以等同于分类算法的评价。
- 除了轮廓系数以外的评价方法，在不考虑业务场景的情况下都是得分越高，其效果越好，最高分值均为1。而轮廓系数则需要判断不同类别数目的情况下其轮廓系数的走势，寻找最优的聚类数目。
- 在具备真实值作为参考的情况下，几种方法均可以很好地评估聚类模型。在没有真实值作为参考的时候，轮廓系数评价方法和Calinski-Harabasz指数评价方法可以结合使用。

# 评价聚类模型

## 聚类模型评价指标

- 使用FMI评价法评价K-Means聚类模型

```
from sklearn.metrics import fowlkes_mallows_score    #FMI评价法
```

```
for i in range(2,7):                                #分别评价K-Means聚类模型聚成2-7类的情况
```

```
    kmeans = KMeans(n_clusters = i,random_state=123).fit(iris_data) #构建并训练模型
```

```
    score = fowlkes_mallows_score(iris_target,kmeans.labels_)
```

```
    print('iris数据聚%d类FMI评价分值为： %f' %(i,score))
```

```
iris数据聚2类FMI评价分值为： 0.750473
iris数据聚3类FMI评价分值为： 0.820808
iris数据聚4类FMI评价分值为： 0.753970
iris数据聚5类FMI评价分值为： 0.725483
iris数据聚6类FMI评价分值为： 0.600691
```

# 评价聚类模型

## 聚类模型评价指标

- 使用轮廓系数评价法评价K-Means聚类模型

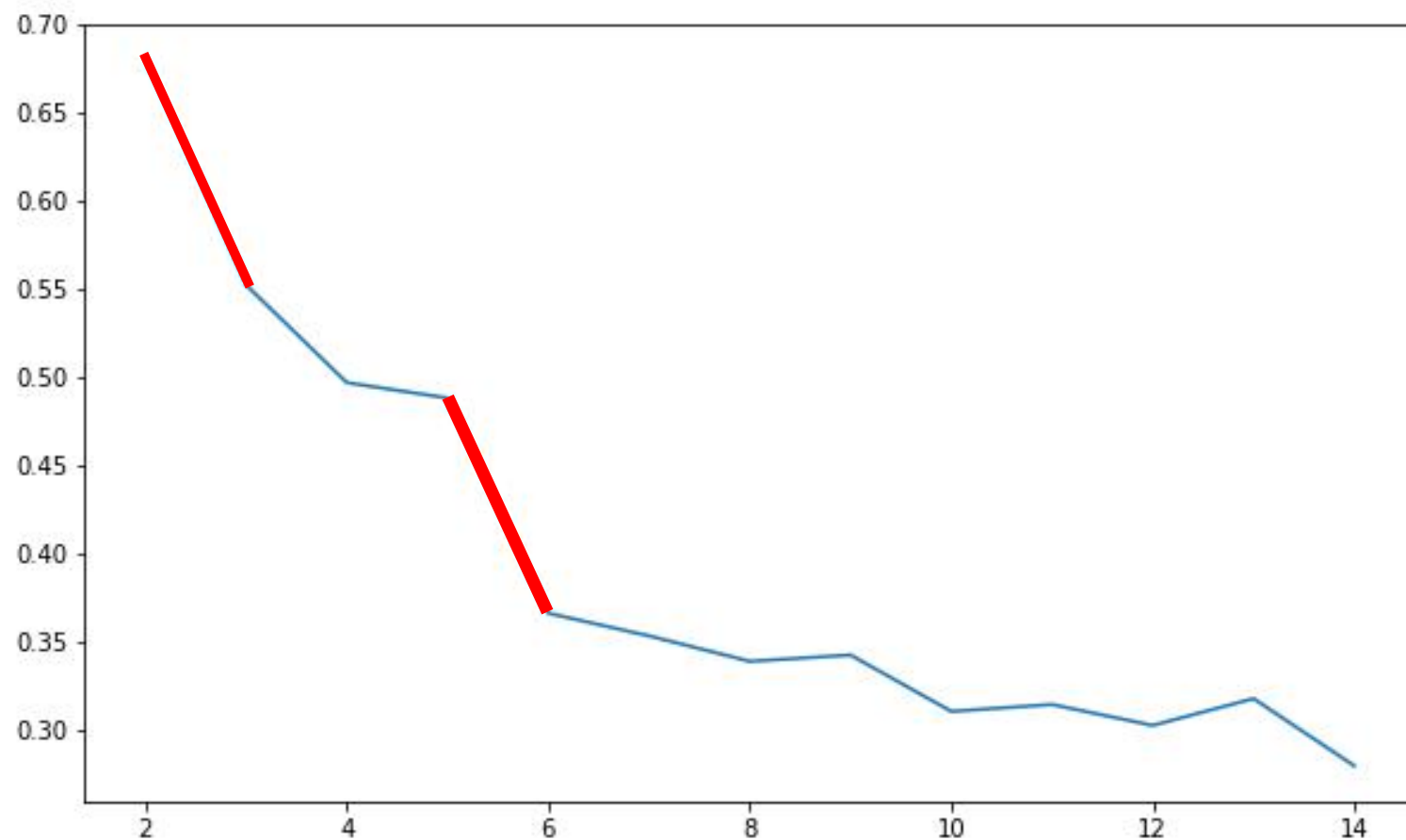
```
from sklearn.metrics import silhouette_score          #轮廓系统评价法
import matplotlib.pyplot as plt
silhouetteScore = []
for i in range(2,15):                                #分别评价K-Means聚类模型聚成2-15类的情况
    kmeans = KMeans(n_clusters = i,random_state=123).fit(iris_data) #构建并训练模型
    score = silhouette_score(iris_data,kmeans.labels_)
    silhouetteScore.append(score)
plt.figure(figsize=(10,6))
plt.plot(range(2,15),silhouetteScore,linewidth=1.5, linestyle="-")
plt.show()
```



# 评价聚类模型

## 聚类模型评价指标

- 使用轮廓系数评价法评价K-Means聚类模型



# 评价聚类模型

## 聚类模型评价指标

- 使用Calinski-Harabasz指数评价法评价K-Means聚类模型

```
from sklearn.metrics import calinski_harabaz_score    #Calinski-Harabasz指数评价法
for i in range(2,7):                                #分别评价K-Means聚类模型聚成2-7类的情况
    kmeans = KMeans(n_clusters = i,random_state=123).fit(iris_data) #构建并训练模型
    score = calinski_harabaz_score(iris_data,kmeans.labels_)
    print('iris数据聚%d类calinski_harabaz指数为 : %f'%(i,score))
```

```
iris数据聚2类calinski_harabaz指数为: 513.303843
iris数据聚3类calinski_harabaz指数为: 560.399924
iris数据聚4类calinski_harabaz指数为: 529.120719
iris数据聚5类calinski_harabaz指数为: 494.094382
iris数据聚6类calinski_harabaz指数为: 474.753604
```

# 构建并评价聚类模型任务

## 对Seeds数据构建K-Means聚类模型并评价该模型

### 1. 对Seeds数据构建K-Means聚类模型

```
import pandas as pd
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.cluster import KMeans
```

```
seeds = pd.read_csv('data/seeds_dataset.txt', sep = '\t')    ##读取数据集
```

```
print('数据集形状为 : ', seeds.shape)
```

```
seeds_data = seeds.iloc[:, :7].values    ##提取数据集中的特征
```

```
seeds_target = seeds.iloc[:, 7].values    ##提取数据集中的标签
```

```
seeds_names = seeds.columns[:7] ##提取特征名
```

# 构建并评价聚类模型任务

---

## 对Seeds数据构建K-Means聚类模型并评价该模型

### 1. 对Seeds数据构建K-Means聚类模型

*# 处理数据*

```
stdScale = StandardScaler().fit(seeds_data)
```

*#标准差标准化数据训练模型*

```
seeds_dataScale = stdScale.transform(seeds_data)
```

*#标准差标准化*

*# 构建并训练模型*

```
kmeans = KMeans(n_clusters=3, random_state=42).fit(seeds_data)
```

```
print('构建的KM-eans模型为 : \n',kmeans)
```

# 构建并评价聚类模型任务

## 对Seeds数据构建K-Means聚类模型并评价该模型

### 2. 对构建的K-Means聚类模型进行评价

```
from sklearn.metrics import calinski_harabaz_score #选取评价方法

for i in range(2,7):
    #构建并训练模型
    kmeans = KMeans(n_clusters = i,random_state=12).fit(seeds_data)
    score = calinski_harabaz_score(seeds_data,kmeans.labels_) #评价模型
    print('seeds数据聚%d类calinski_harabaz指数为 : %f'%(i,score))
```

```
seeds数据聚2类calinski_harabaz指数为： 351.179992
seeds数据聚3类calinski_harabaz指数为： 375.804961
seeds数据聚4类calinski_harabaz指数为： 327.835320
seeds数据聚5类calinski_harabaz指数为： 310.331840
seeds数据聚6类calinski_harabaz指数为： 302.473069
```

# 目录

---



# 构建并评价分类模型

---

## 任务描述

- 分类是指构造一个分类模型，输入样本的特征值，输出对应的类别，将每个样本映射到预先定义好的类别。分类模型建立在已有类别标记的数据集上，属于有监督学习。
- 在实际应用场景中，分类算法被应用于行为分析、物品识别、图像检测等。

## 任务分析

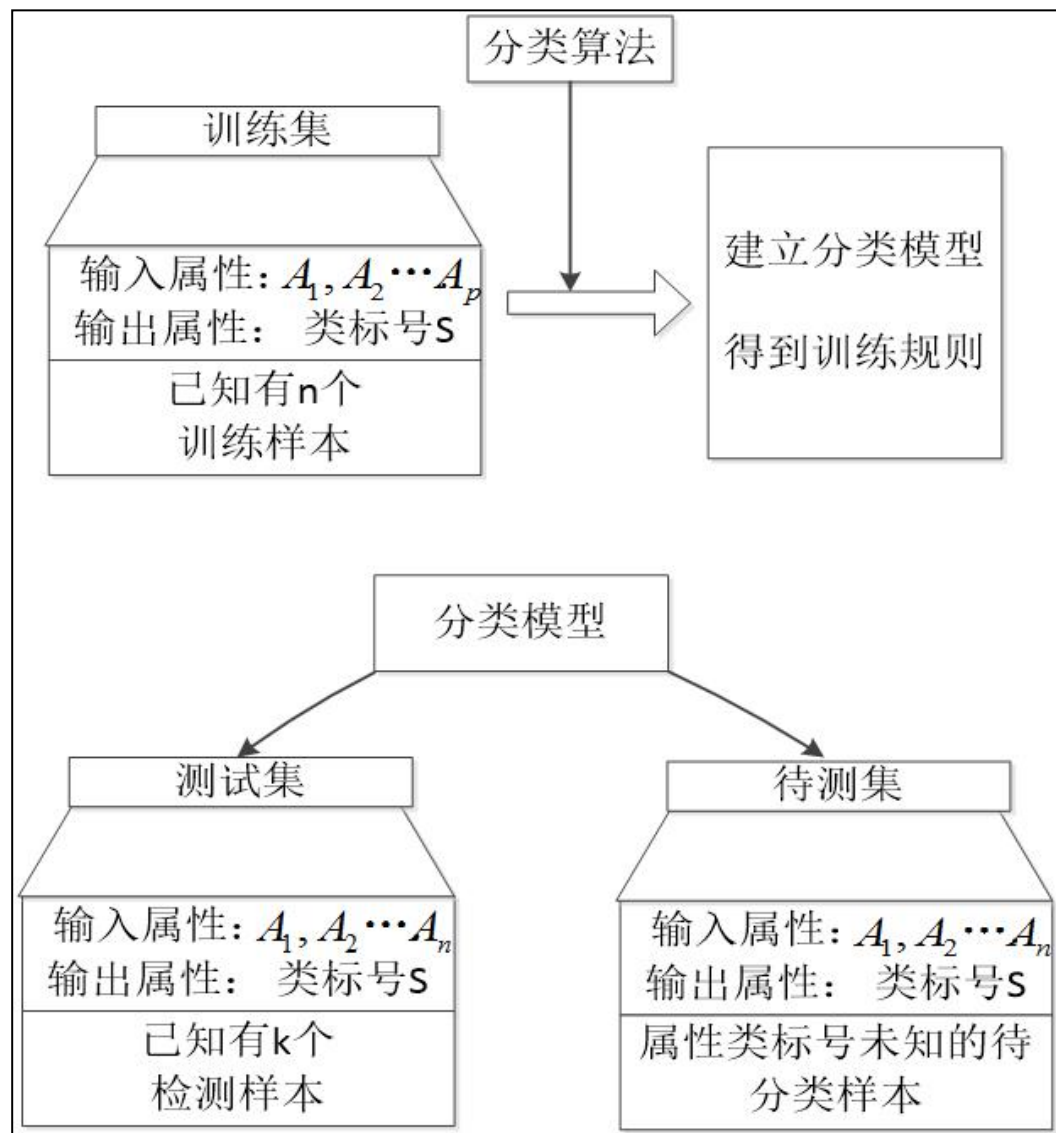
- ( 1 ) 使用sklearn估计器构建支持向量机 ( SVM ) 模型
- ( 2 ) 根据分类模型的评价指标评价支持向量机 ( SVM ) 模型。



# 使用sklearn估计器构建分类模型

## 分类算法的实现过程

- 在数据分析领域，分类算法有很多，其原理千差万别
  - ▣ 有基于样本距离的最近邻算法；
  - ▣ 有基于特征信息熵的决策树；
  - ▣ 有基于bagging的随机森林；
  - ▣ 有基于boosting的梯度提升分类树；
- 但其实现的过程相差不大。过程如图所示。



# 使用sklearn估计器构建分类模型

---

## 构建SVM模型

*# 加载所需的函数*

```
import numpy as np
```

```
from sklearn.datasets import load_breast_cancer
```

```
from sklearn.svm import SVC
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
```

```
cancer = load_breast_cancer()
```

```
cancer_data = cancer['data']
```

```
cancer_target = cancer['target']
```

```
cancer_names = cancer['feature_names']
```

# 使用sklearn估计器构建分类模型

---

## 构建SVM模型

*# 将数据划分为训练集测试集*

```
cancer_data_train, cancer_data_test, cancer_target_train, cancer_target_test = \
    train_test_split(cancer_data, cancer_target, test_size = 0.2, random_state = 22)
```

*# 数据标准化*

```
stdScaler = StandardScaler().fit(cancer_data_train)
cancer_trainStd = stdScaler.transform(cancer_data_train)
cancer_testStd = stdScaler.transform(cancer_data_test)
```

# 使用sklearn估计器构建分类模型

## 构建SVM模型

*## 建立SVM模型*

```
svm = SVC().fit(cancer_trainStd,cancer_target_train)      #有监督学习
```

```
print('建立的SVM模型为 : \n',svm)
```

*## 预测测试集结果*

```
cancer_target_pred = svm.predict(cancer_testStd)
```

```
print('预测前20个结果为 : \n',cancer_target_pred[:20])
```

建立的SVM模型为：

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

预测前20个结果为：

```
[1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1]
```

# 使用sklearn估计器构建分类模型

## 构建SVM模型

- 将预测结果和真实结果做比对，求出预测对的结果和预测错的结果，并求出准确率。

*# 求出预测和真实一样的数目*

```
true = np.sum(cancer_target_pred == cancer_target_test )
```

```
print('预测对的结果数目为：', true)
```

```
print('预测错的的结果数目为：', cancer_target_test.shape[0]-true)
```

```
print('预测结果准确率为：', true/cancer_target_test.shape[0])
```

```
预测对的结果数目为： 111  
预测错的的结果数目为： 3  
预测结果准确率为： 0.9736842105263158
```

# 使用sklearn估计器构建分类模型

## sklearn库常用分类算法函数

➤ sklearn中提供的分类算法非常多，分别存在于不同的模块中。常用的分类算法如下表所示。

模块名称	函数名称	算法名称
linear_model	LogisticRegression	逻辑斯蒂回归
svm	SVC	支持向量机
neighbors	KNeighborsClassifier	K最近邻分类
naive_bayes	GaussianNB	高斯朴素贝叶斯
tree	DecisionTreeClassifier	分类决策树
ensemble	RandomForestClassifier	随机森林分类
ensemble	GradientBoostingClassifier	梯度提升分类树

# 评价分类模型

## 分类模型的评价指标

- 分类模型对测试集进行预测而得出的准确率并不能很好地反映模型的性能，为了有效判断一个预测模型的性能表现，需要结合真实值，计算出精确率、召回率、F1值和Cohen's Kappa系数等指标来衡量。
- 常规分类模型的评价指标如表所示。分类模型评价方法前4种都是分值越高越好，其使用方法基本相同。

方法名称	最佳值	sklearn函数
Precision ( 精确率 )	1.0	metrics.precision_score
Recall ( 召回率 )	1.0	metrics.recall_score
F1值	1.0	metrics.f1_score
Cohen' s Kappa系数	1.0	metrics.cohen_kappa_score
ROC曲线	最靠近y轴	metrics. roc_curve

# 评价分类模型

## 分类模型的评价指标

- Precision ( 精确率 )、Recall ( 召回率 )、F1值、AUC和ROC曲线其实都是评价模型好坏的指标，而且相互之间是有关系的，只是侧重点不同，下面就用一个例子解释这些指标。
- 以一个有监督的二分类预测模型为例，模型对每个样本的预测结果为一个概率值，我们需要从中选取一个阈值来区分为预测结果为正值，还是负值。这样就可以计算出混淆矩阵 ( Confusion matrix )。

混淆矩阵		预测表现	
		0	1
实际表现	0	True Negative ( TN )	False Positive ( FP )
	1	False Negative ( FN )	True Positive ( TP )

精确率 $P = TP / (TP + FP)$

召回率 $R = TP / (TP + FN)$

$F1 = 2PR / (P + R)$

F1相当于精确率和召回率的  
综合评价指标



# 评价分类模型

## 分类模型的评价实例

```
from sklearn.metrics import accuracy_score, precision_score,  
                             recall_score, f1_score, cohen_kappa_score  
  
print('使用SVM预测breast_cancer数据的准确率为 : ',  
      accuracy_score(cancer_target_test, cancer_target_pred))  
print('使用SVM预测breast_cancer数据的精确率为 : ',  
      precision_score(cancer_target_test, cancer_target_pred))  
print('使用SVM预测breast_cancer数据的召回率为 : ',  
      recall_score(cancer_target_test, cancer_target_pred))  
print('使用SVM预测breast_cancer数据的F1值为 : ',  
      f1_score(cancer_target_test, cancer_target_pred))  
print('使用SVM预测breast_cancer数据的Cohen' s Kappa系数为 : ',  
      cohen_kappa_score(cancer_target_test, cancer_target_pred))
```

# 评价分类模型

## 分类模型的评价实例

```
from sklearn.metrics import accuracy_score, precision_score,  
                                recall_score, f1_score, cohen_kappa_score
```

```
print('使用SVM预测breast_cancer数据的准确率为：',
```

```
accuracy_score(cancer_target_test, cancer_target_pred))
```

```
print('使用SVM预测breast_cancer数据的精确率为：',
```

```
precision_score(cancer_target_test, cancer_target_pred))
```

```
print('使用SVM预测breast_cancer数据的召回率为：',
```

```
使用SVM预测breast_cancer数据的准确率为： 0.9736842105263158
```

```
使用SVM预测breast_cancer数据的精确率为： 0.9594594594594594
```

```
使用SVM预测breast_cancer数据的召回率为： 1.0
```

```
使用SVM预测breast_cancer数据的F1值为： 0.9793103448275862
```

```
使用SVM预测breast_cancer数据的Cohen's Kappa系数为： 0.9432082364662903
```

```
cohen_kappa_score(cancer_target_test, cancer_target_pred))
```

# 评价分类模型

## 分类模型的评价指标

- sklearn的metrics模块除了提供准确率、精确率、召回率等单一评价指标的函数外，还提供了一个能够输出分类模型评价报告的函数classification\_report，代码如下：

```
from sklearn.metrics import classification_report  
  
print('使用SVM预测iris数据的分类报告为：', '\n',  
      classification_report(cancer_target_test, cancer_target_pred))
```

使用SVM预测breast\_cancer数据的分类报告为：

	precision	recall	f1-score	support
0	1.00	0.93	0.96	43
1	0.96	1.00	0.98	71
avg / total	0.97	0.97	0.97	114

# 评价分类模型

## ROC曲线

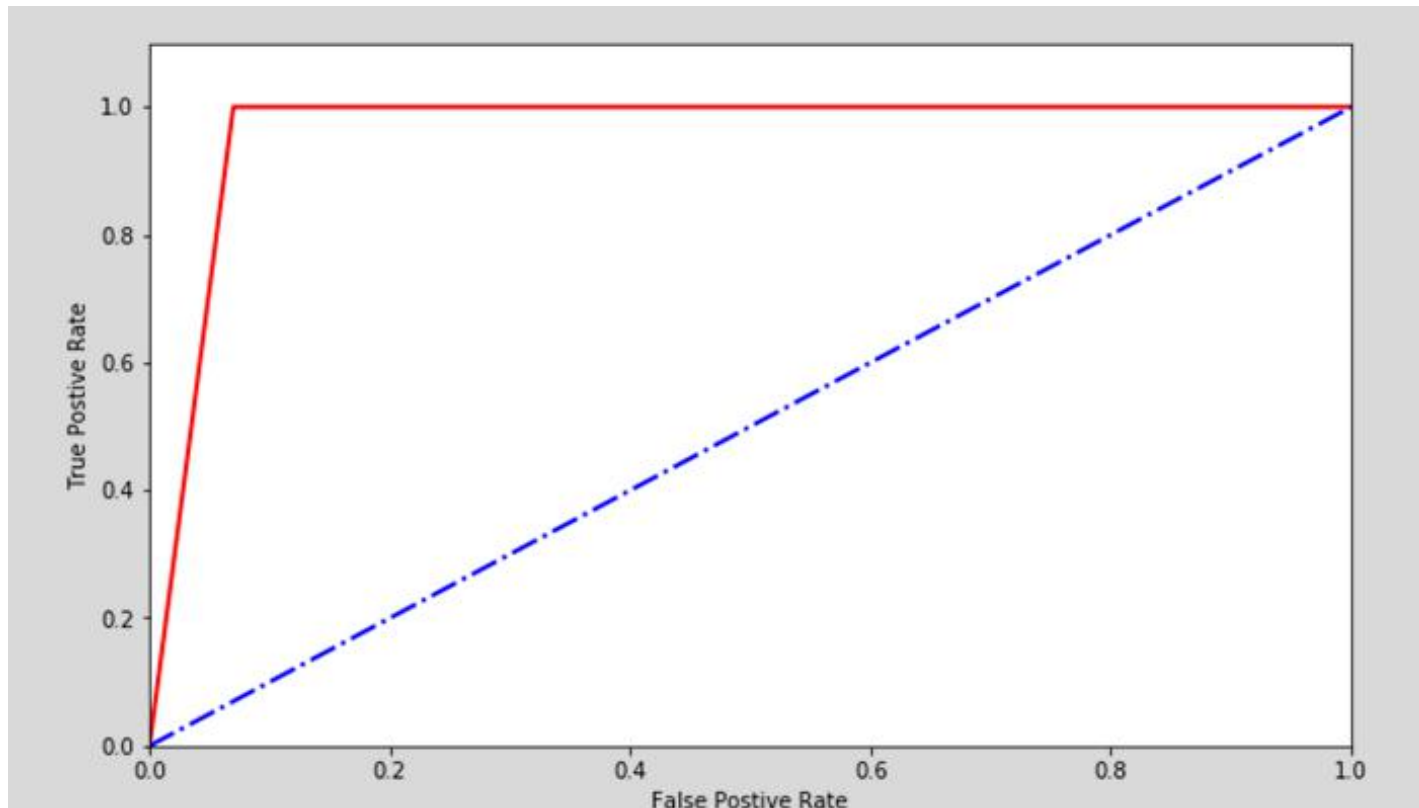
- 除了使用数值，表格形式评估分类模型的性能，还可通过绘制ROC曲线的方式来评估分类模型。

```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
# 求出ROC曲线的x轴和y轴
fpr, tpr, thresholds = roc_curve(cancer_target_test, cancer_target_pred)
plt.figure(figsize=(10,6))
plt.xlim(0,1) # 设定x轴的范围
plt.ylim(0.0,1.1) # 设定y轴的范围
plt.xlabel('False Postive Rate') # 假阳性
plt.ylabel('True Postive Rate') # 真阳性
plt.plot(fpr,tpr,linewidth=2, linestyle="--",color='red')
plt.show()
```

# 评价分类模型

## ROC曲线

- ROC曲线横纵坐标范围为 $[0,1]$ ，通常情况下ROC曲线与X轴形成的面积越大，表示模型性能越好。但是当ROC曲线处于下图中蓝色虚线的位置，就表明了模型的计算结果基本都是随机得来的，在此种情况下模型起到的作用几乎为零。故在实际中ROC曲线离图中蓝色虚线越远表示模型效果越好。



# 构建并评价分类模型

---

## 任务实现

- 常规的鲍鱼的年龄是通过显微镜查看切削、染色后的外壳上的环的数量得到的，十分耗时。
- 一些常规的物理量（如性别、长度、高度、壳体重量等）的测量十分容易获得，若能够使用这些物理量预测年龄将节省大量时间，使用sklearn构建鲍鱼预测的步骤如下。

1. 构建SVM分类模型
2. 评价构建的SVM分类模型

# 构建并评价分类模型

## 任务实现

### 1. 构建SVM分类模型

*# 加载所需的函数*

```
import pandas as pd
```

```
from sklearn.svm import SVC
```

```
from sklearn.metrics import classification_report
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
```

```
abalone = pd.read_csv('data/abalone.data',sep=',')
```

*# 将数据和标签拆开*

```
abalone_data = abalone.iloc[:,8] #提取鲍鱼数据中的特征数据
```

```
abalone_target = abalone.iloc[:,8] #提取鲍鱼数据中的特征标签
```

# 构建并评价分类模型

---

## 任务实现

### 1. 构建SVM分类模型

**## 连续型特征离散化**

```
sex = pd.get_dummies(abalone_data['sex'])
```

```
abalone_data = pd.concat([abalone_data,sex], axis = 1 )
```

```
abalone_data.drop('sex',axis = 1,inplace = True)
```

**## 划分训练集，测试集**

```
abalone_train,abalone_test, abalone_target_train,abalone_target_test = \  
train_test_split(abalone_data,abalone_target,train_size = 0.8,random_state = 42)
```



# 构建并评价分类模型

## 任务实现

### 1. 构建SVM分类模型

#### ## 标准化

```
stdScaler = StandardScaler().fit(abalone_train)
abalone_std_train = stdScaler.transform(abalone_train)
abalone_std_test = stdScaler.transform(abalone_test)
```

#### ## 建模

```
svm_abalone = SVC().fit(abalone_std_train, abalone_target_train)
print('建立的SVM模型为 : ', '\n', svm_abalone)
```

建立的SVM模型为：

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

# 构建并评价分类模型

## 任务实现

### 2. 评价构建的SVM分类模型

```
abalone_target_pred = svm_abalone.predict(abalone_std_test)
```

```
print('abalone数据集的SVM分类报告为：\n',
```

```
      classification_report(abalone_target_test,abalone_target_pred))
```

```
....:      classification_report(abalone_target_test,abalone_target_pred))
abalone数据集的SVM分类报告为：
```

	precision	recall	f1-score	support
3	0.00	0.00	0.00	3
4	0.45	0.69	0.55	13
5	0.54	0.22	0.31	32
6	0.40	0.33	0.36	48
7	0.41	0.44	0.42	84
8	0.37	0.36	0.37	99
9	0.28	0.57	0.38	142
10	0.24	0.33	0.28	139
11	0.25	0.25	0.25	93
12	0.00	0.00	0.00	51

# 目录

---



# 构建并评价回归模型

---

## 任务描述

- 回归算法的实现过程与分类算法类似，原理相差不大。
- 分类和回归的主要区别在于，分类算法的标签是离散的，但是回归算法的标签是连续的。
- 回归算法在交通、物流、社交网络和金融领域都能发挥巨大作用。

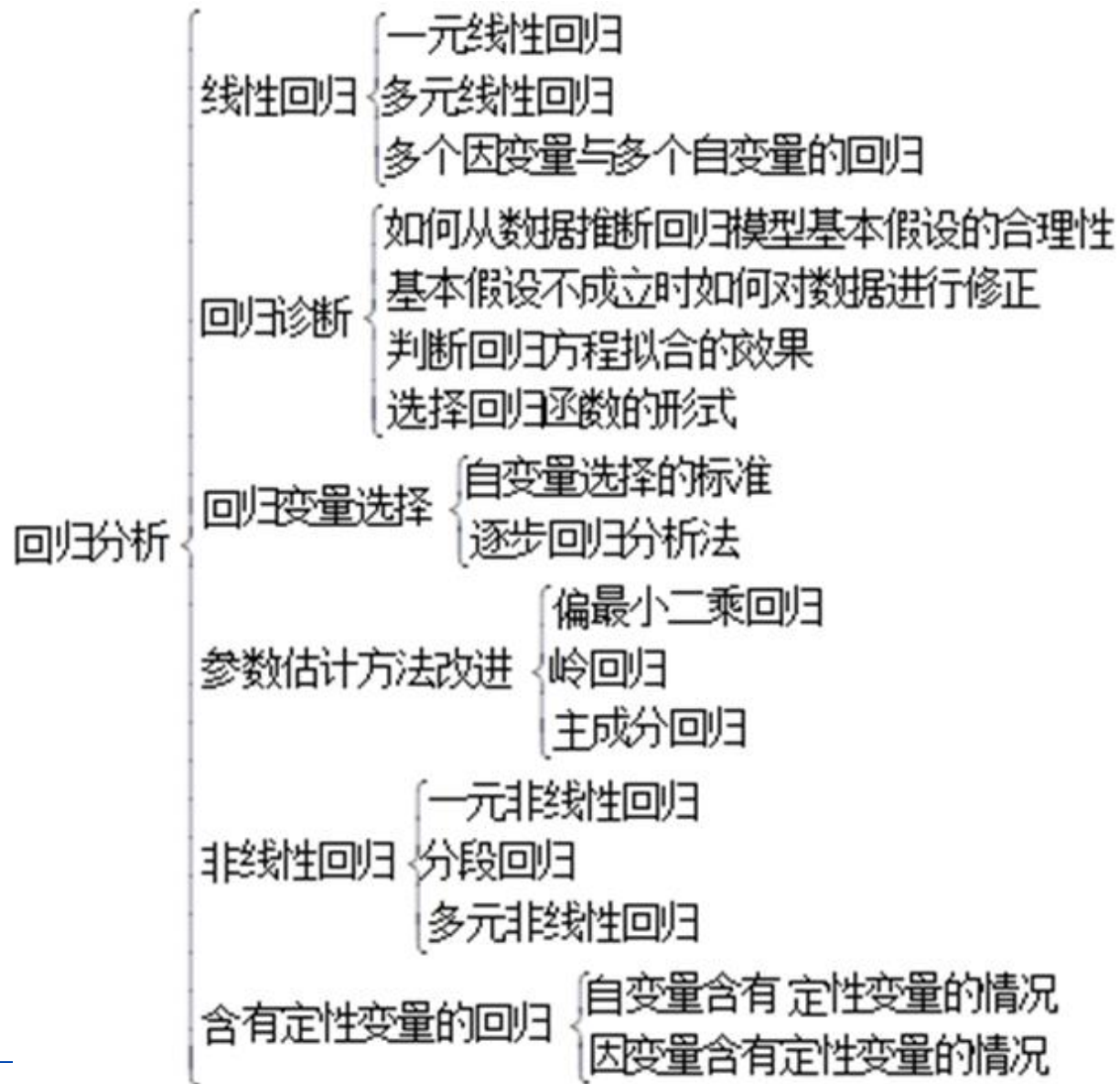
## 任务分析

- ( 1 ) 使用sklearn估计器构建线性回归 ( Linear Regression ) 模型
- ( 2 ) 根据回归模型评价指标评价线性回归模型

# 使用sklearn估计器构建回归模型

## 回归分析方法

- 从19世纪初高斯提出最小二乘估计法算起，回归分析的历史已有200多年。
- 从经典的回归分析方法到近代的回归分析方法，按照研究方法划分，回归分析研究的范围大致如图所示。
- 回归算法的实现步骤和分类算法基本相同，分为**学习和预测**2个步骤：
  - ▣ 学习是通过训练样本数据来拟合回归方程；
  - ▣ 预测则是利用学习过程中拟合出的回归方程，将测试数据放入方程中求出预测值。



# 使用sklearn估计器构建回归模型

## 常用的回归模型

回归模型名称	适用条件	算法描述
线性回归	因变量与自变量是线性关系	对一个或多个自变量和因变量之间的线性关系进行建模，可用最小二乘法求解模型系数。
非线性回归	因变量与自变量之间不都是线性关系	对一个或多个自变量和因变量之间的非线性关系进行建模。如果非线性关系可以通过简单的函数变换转化成线性关系，用线性回归的思想求解；如果不能转化，用非线性最小二乘方法求解。
Logistic回归	因变量一般有1和0（是与否）两种取值	是广义线性回归模型的特例，利用Logistic函数将因变量的取值范围控制在0和1之间，表示取值为1的概率。
岭回归	参与建模的自变量之间具有多重共线性	是一种改进最小二乘估计的方法。
主成分回归	参与建模的自变量之间具有多重共线性	主成分回归是根据主成分分析的思想提出来的，是对最小二乘法的一种改进，它是参数估计的一种有偏估计。可以消除自变量之间的多重共线性。

# 使用sklearn估计器构建回归模型

## sklearn库常用回归算法函数

➤ sklearn内部提供了不少回归算法，常用的函数如下表所示。

模块名称	函数名称	算法名称
linear_model	LinearRegression	线性回归
svm	SVR	支持向量回归
neighbors	KNeighborsRegressor	最近邻回归
tree	DecisionTreeRegressor	回归决策树
ensemble	RandomForestRegressor	随机森林回归
ensemble	GradientBoostingRegressor	梯度提升回归树

# 使用sklearn估计器构建回归模型

---

## 使用sklearn估计器构建boston数据集线性回归模型

**##加载所需函数**

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.datasets import load_boston
```

```
from sklearn.model_selection import train_test_split
```

**## 加载boston数据**

```
boston = load_boston()
```

```
X = boston['data']    #提取boston数据集中的特征数据
```

```
y = boston['target']  #提取boston数据集中的特征标签
```

```
boston_names = boston['feature_names'] #提取boston数据集中的特征名称
```



# 使用sklearn估计器构建回归模型

## 使用sklearn估计器构建boston数据集线性回归模型

## 将数据划分为训练集测试集

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state=125)
```

```
clf = LinearRegression().fit(X_train, y_train)          ## 建立线性回归模型 ( 监督学习 )
```

```
print('建立的LinearRegression模型为 : ','\n',clf)
```

```
y_pred = clf.predict(X_test)  ## 预测训练集结果
```

```
print('预测前20个结果为 : ','\n',y_pred[:20])
```

建立的LinearRegression模型为：

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

预测前20个结果为：

```
[21.12953164 19.67578799 22.01735047 24.62046819 14.45164813 23.32325459
16.6468677  14.9175848  33.58466804 17.48328609 25.50385719 36.60215179
25.95309333 28.48503161 19.34928078 20.16966217 25.9788081  18.25959831
16.52754056 17.08448854]
```

# 使用sklearn估计器构建回归模型

## 模型回归结果可视化

- 可以利用预测结果和真实结果画出折线图作对比，以便更直观看出现性回归模型效果。

```
import matplotlib.pyplot as plt
```

```
from matplotlib import rcParams
```

```
rcParams['font.sans-serif'] = 'SimHei'
```

```
fig = plt.figure(figsize=(10,6))    ##设定空白画布，并制定大小
```

```
##用不同的颜色表示不同数据
```

```
plt.plot(range(y_test.shape[0]),y_test,color="blue", linewidth=1.5, linestyle="-")
```

```
plt.plot(range(y_test.shape[0]),y_pred,color="red", linewidth=1.5, linestyle="-.")
```

```
plt.legend(['真实值','预测值'])
```

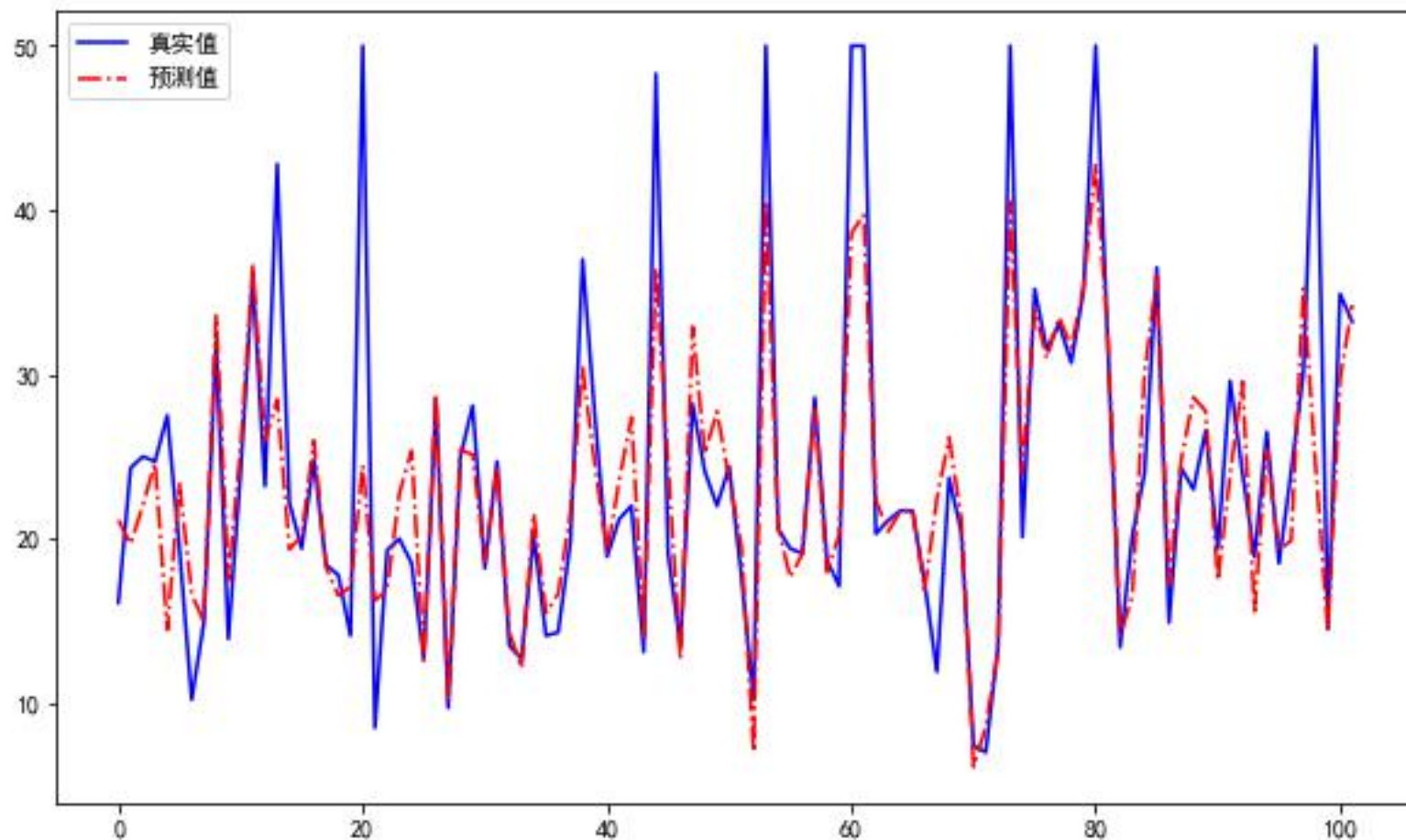
```
plt.savefig('../tmp/聚类结果.png')
```

```
plt.show() ##显示图片
```

# 使用sklearn估计器构建回归模型

## 模型回归结果可视化

- 可以利用预测结果和真实结果画出折线图作对比，以便更直观看出现性回归模型效果。



# 评价回归模型

## 回归模型评价指标

- 回归模型的性能评估不同于分类模型，虽然都是对照真实值进行评估，但由于回归模型的预测结果和真实值都是连续的，所以不能够求取Precision、Recall和F1值等评价指标。**回归模型拥有一套独立的评价指标。**
- 常用的回归模型评价指标如下表所示。平均绝对误差、均方误差和中值绝对误差的值越靠近0，模型性能越好。可解释方差值和R方值则越靠近1，模型性能越好。

方法名称	最优值	sklearn函数
平均绝对误差	0.0	metrics.mean_absolute_error
均方误差	0.0	metrics.mean_squared_error
中值绝对误差	0.0	metrics.median_absolute_error
可解释方差值	1.0	metrics.explained_variance_score
R方值	1.0	metrics.r2_score

# 评价回归模型

---

## 实例

*# 导入平均绝对误差, 均方误差, 中值绝对误差,  $R^2$ 值, 可解释方差值评估模块*

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, \
    median_absolute_error, explained_variance_score, r2_score
print('Boston数据线性回归模型的平均绝对误差为：', mean_absolute_error(y_test,y_pred))
print('Boston数据线性回归模型的均方误差为：', mean_squared_error(y_test,y_pred))
print('Boston数据线性回归模型的中值绝对误差为：', median_absolute_error(y_test,y_pred))
print('Boston数据线性回归模型的可解释方差值为：', explained_variance_score(y_test,y_pred))
print('Boston数据线性回归模型的R方值为：', r2_score(y_test,y_pred))
```

# 评价回归模型

## 实例

*# 导入平均绝对误差, 均方误差, 中值绝对误差,  $R^2$ 值, 可解释方差值评估模块*

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, \
    median_absolute_error, explained_variance_score, r2_score
```

```
print('Boston数据线性回归模型的平均绝对误差为:', mean_absolute_error(y_test,y_pred))
```

```
print('Boston数据线性回归模型的均方误差为:', mean_squared_error(y_test,y_pred))
```

```
Boston数据线性回归模型的平均绝对误差为: 3.377642697362818
Boston数据线性回归模型的均方误差为: 31.15059667690504
Boston数据线性回归模型的中值绝对误差为: 1.7774213157360652
Boston数据线性回归模型的可解释方差值为: 0.7105949626282904
Boston数据线性回归模型的R方值为: 0.7068954225782409
```

# 构建并评价回归模型任务

## 1. 构建加利福尼亚住房数据回归模型

```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor # 梯度提升回归模型
from sklearn.model_selection import train_test_split
house = pd.read_csv('data/cal_housing.data',sep=',')
house_data = house.iloc[:, :-1]
house_target = house.iloc[:, -1]
house_names = ['longitude', 'latitude', 'housingMedianAge', 'totalRooms',
               'totalBedrooms', 'population', 'households', 'medianIncome']
house_train, house_test, house_target_train, house_target_test = \
    train_test_split(house_data, house_target, test_size = 0.2, random_state = 42)
GBR_house = GradientBoostingRegressor().fit(house_train, house_target_train)
print('建立的梯度提升回归模型为：', '\n', GBR_house)
```



# 构建并评价回归模型任务

## 2. 评价构建的加利福尼亚住房数据回归模型

```
house_target_pred = GBR_house.predict(house_test)
```

```
#对回归模型进行评估，导入平均绝对误差，均方误差，中值绝对误差，R2值，可解释方差值评估模块
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,median_absolute_error, \
                                explained_variance_score,r2_score
```

```
print('回归树模型的平均绝对误差为：', mean_absolute_error(house_target_test,house_target_pred))
```

```
print('回归树模型的均方误差为：', mean_squared_error(house_target_test,house_target_pred))
```

```
print('回归树模型的中值绝对误差为：', median_absolute_error(house_target_test,house_target_pred))
```

```
print('回归树模型的可解释方差值为：',explained_variance_score(house_target_test, house_target_pred))
```

```
print('california_housing数据梯度提升回归树模型的R方值为：',
                                r2_score(house_target_test,house_target_pred))
```



# 构建并评价回归模型任务

## 2. 评价构建的加利福尼亚住房数据回归模型

```
house_target_pred = GBR_house.predict(house_test)
```

```
#对回归模型进行评估，导入平均绝对误差，均方误差，中值绝对误差，R2值，可解释方差值评估模块
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,median_absolute_error, \
    explained_variance_score,r2_score
```

```
print('回归树模型的平均绝对误差为：', mean_absolute_error(house_target_test,house_target_pred))
```

```
print('回归树模型的均方误差为：', mean_squared_error(house_target_test,house_target_pred))
```

```
print('回归树模型的中值绝对误差为：', median_absolute_error(house_target_test,house_target_pred))
```

```
california_housing数据梯度提升回归树模型的平均绝对误差为： 38056.75815557929 (house_target_pred))
california_housing数据梯度提升回归树模型的均方误差为： 3103233005.5273113
california_housing数据梯度提升回归树模型的中值绝对误差为： 26179.478445698274
california_housing数据梯度提升回归树模型的可解释方差值为： 0.7618952080246163 (house_target_pred))
california_housing数据梯度提升回归树模型的R方值为： 0.7618473592138286
```

# 构建并评价回归模型任务

## 2. 评价构建的加利福尼亚住房数据回归模型

#尝试使用随机森林回归

```
from sklearn.ensemble import RandomForestRegressor
```

```
RFR_house = RandomForestRegressor().fit(house_train,house_target_train)
```

```
print('建立的随机森林回归模型为 : ','\n',RFR_house)
```

```
house_target_pred = RFR_house.predict(house_test)
```

```
california_housing数据随机森林回归模型的平均绝对误差为: 34279.96727228681
california_housing数据随机森林回归模型的均方误差为: 2815471138.9420857
california_housing数据随机森林回归模型的中值绝对误差为: 20175.0
california_housing数据随机森林回归模型的可解释方差值为: 0.7841673132372502
california_housing数据随机森林回归模型的R方值为: 0.7839311822212423
```

➤ 尝试用随机森林回归构建模型，并评价，发现更好。

# 目录

---



# 小结

---

本章主要根据数据分析的应用分类，重点介绍了对应的数据分析建模方法及实现过程。

- sklearn数据分析技术的基本任务主要体现在聚类、分类和回归三类。
- 每一类又有对应的多种评估方法，能够评价所构建模型的性能优劣。

通过这一章的学习，基本能够掌握常用的模型构建与评估方法，可在以后的数据分析过程中采用适当的算法并按所介绍的步骤实现综合应用。