第5讲 Scikit-learn

- 初用sklearn
- 案例--鸢尾花分类
- 模型评估方法
- 模型评价指标
- 偏差与方差均衡

初用sklearn

- scikit-learn(sklearn)：开源的机器学习工具包

  官网：http://scikit-learn.org/stable/testimonials/testimonials.html

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

初用 sklearn

- Scikit-learn安装与引用：

  安装：pip install scikit-learn

  conda install scikit-learn

  导入：import sklearn

  from sklearn. <...> import ...

- 示例: from sklearn import model_selection

  from sklearn.linear_model import LogisticRegression

初用sklearn

sklearn数据格式：Array-like

- The most common data format for input to Scikit-learn estimators and functions, array-like is any type object for which numpy.asarray will produce an array of appropriate shape (usually 1 or 2-dimensional) of appropriate dtype (usually numeric).

a numpy array，a list of numbers，a list of length-k lists of numbers for some fixed length k，a pandas.DataFrame with all columns numeric，a numeric pandas.Series

- Note that output from scikit-learn estimators and functions (e.g. predictions) should generally be arrays or sparse matrices, or lists thereof (as in multi-output tree.DecisionTreeClassifier’s predict_proba)

初用sklearn

sklearn内置数据集

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

- 三种引入数据形式：

打包好的数据：对于小数据集
sklearn.datasets.load_*

分流下载数据：对于大数据集
sklearn.datasets.fetch_*

随机创建数据：为了快速展示
sklearn.datasets.make_*

	数据集名称	调用方式	适用算法	数据规模
小数据集	波士顿房价数据集	load_boston()	回归	506*13
	鸢尾花数据集	load_iris()	分类	150*4
	糖尿病数据集	load_diabetes()	回归	442*10
	手写数字数据集	load_digits()	分类	5620*64
大数据集	Olivetti脸部图像数据集	fetch_olivetti_faces()	降维	400*64*64
	新闻分类数据集	fetch_20newsgroups()	分类	-
	带标签的人脸数据集	fetch_lfw_people()	分类；降维	-
	路透社新闻语料数据集	fetch_rcv1()	分类	804414*47236

注：小数据集可以直接使用，大数据集在第一次使用的时会自动下载

初用sklearn

- 示例：sklearn.datasets 模块中引入iris数据集

```
from sklearn.datasets import load_iris
iris = load_iris()
```

数据是以字典(dict)格式存储，查看 iris键：

```
iris.keys()
```

```
dict_keys(['data', 'target', 'DESCR', 'feature_names', 'target_names'])
```

初用sklearn

◆ 示例：sklearn.datasets 模块中导入iris数据集


```
print(iris.data.shape)
print(iris.feature_names)
iris.data[0:5]
```

```
(150, 4)
['sepal length (cm)', 'sepal width (cr

array([[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]])
```

```
print(iris.target.shape)
print(iris.target_names)
iris.target
```

```
(150,)
['setosa' 'versicolor' 'virginica']

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

初用sklearn

◆ 示例：sklearn.datasets API获取20newsgroups数据集

```
from sklearn.datasets import fetch_20newsgroups
newsgroups_train = fetch_20newsgroups(subset='train')
```

```
Downloading 20news dataset. This may take a few minutes.
Downloading dataset from https://ndownloader.figshare.com/files/5975967 (14 MB)
```

```
newsgroups_train.keys()
```

```
dict_keys(['target', 'description', 'data', 'DESCR', 'target_names', 'filenames'])
```

初用 sklearn

- 示例，生成分类数据集: sklearn.datasets.make_classification

```
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=6, n_features=5, n_informative=2,
    n_redundant=2, n_classes=2, n_clusters_per_class=2, scale=1.0,
    random_state=20)
```

```
print(X.shape,y.shape)
```

```
(6, 5) (6,)
```

初用sklearn

估计器(Estimator)：实现了机器学习算法的APIs

例如，如下不同类型的估计器

分类器(classifier)

- sklearn.neighbors
- sklearn.naive_bayes
- sklearn.linear_model.LogisticRegression

回归器(regressor)

- sklearn.linear_model.LinearRegression
  sklearn.linear_model.Ridge

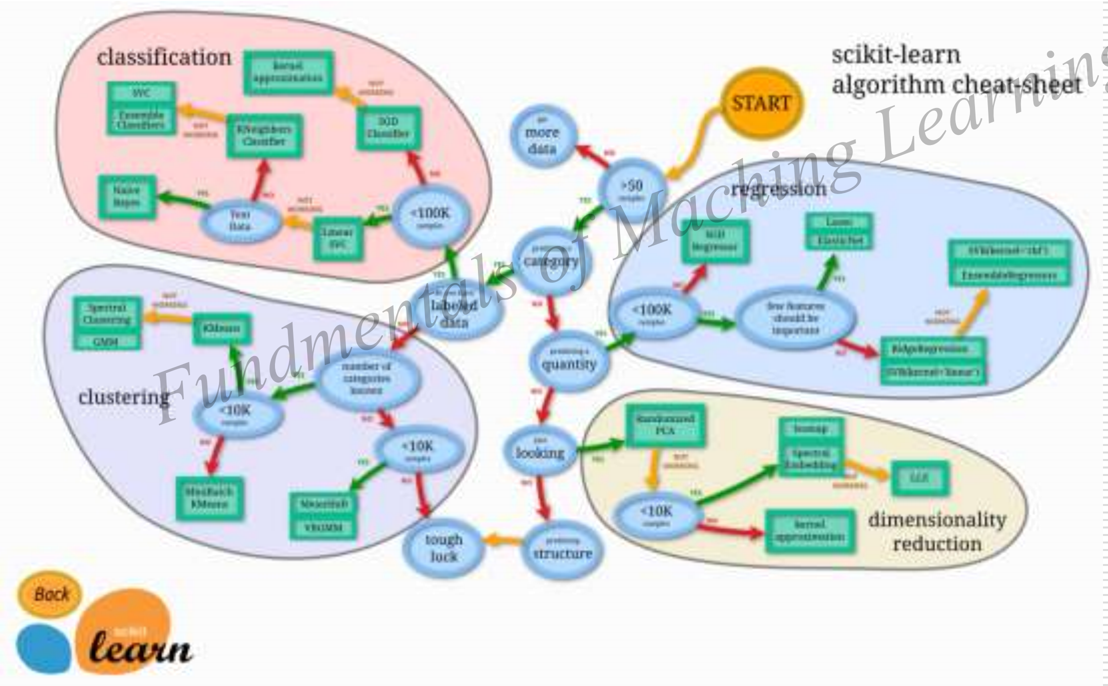
聚类器(cluster)

- sklearn.cluster.KMeans

初用 sklearn

选择估计器

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



初用sklearn

- sklearn构建模型(估计器)：

建立估计器对象
（类实例化）

参数(Parameters)

调用估计器对象
Methods
Parameters
Attributes

所有对象接口一致

初用sklearn

◆ 示例：构建KNN分类模型(估计器)


```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)

X = [[ 1,  2,  3],  # 2 samples, 3 features
     [11, 12, 13]]
y = [0, 1]  # classes of each sample

knn.fit(X, y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

初用 sklearn

◆ 示例：访问模型(估计器)参数

估计器设置的超参数和属性可通过实例的变量直接访问，区别是超参数的名称最后没有下划线 _，而属性的名称最后有下划线 _

- estimator.parameter
- estimator.attribute_

```
knn.n_neighbors
```

```
1
```

```
knn.classes_
```

```
array([0, 1])
```

初用sklearn

- 模型选择： sklearn.model_selection.*
  - train_test_split：划分数据集
  - GridSearchCV： 网格搜索最佳超参数
  - RandomizedSearchCV：随机搜索最佳超参数
  - cross_validate：交叉验证
  - learning_curve：绘制学习曲线
  - validation_curve：绘制验证曲线

初用sklearn

◆ 示例：模型选择-分割数据集

```
from sklearn.datasets import load_iris
iris = load_iris()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    iris['data'], iris['target'], random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, me
           metric_params=None, n_jobs=1, n_neighbors=1,
           weights='uniform')
```

```
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
```

```
X_train shape: (112, 4)
y_train shape: (112,)
```

```
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```

```
X_test shape: (38, 4)
y_test shape: (38,)
```

```
print("X_train score: {:.2f}".format(knn.score(X_train, y_train)))
print("X_test score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
X_train score: 1.00
X_test score: 0.97
```

初用sklearn

◆ 示例：模型选择--交叉验证

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=200, centers =3, random_state=8)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X, y)

from sklearn.model_selection import cross_validate
result = cross_validate(knn, X, y)  # defaults to 3-fold CV
print(result['test_score'])
```

```
[1. 1. 1.]
```

初用sklearn

◆ 示例：模型选择--自动搜索模型参数

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=200, centers =3, random_state=8)

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

# 优化参数k的取值范围
k_range = range(1, 30)
param_grid = {'n_neighbors':k_range}

knn = KNeighborsClassifier(n_neighbors=5)
grid = GridSearchCV(estimator = knn, param_grid = param_grid, cv=10, scoring='accuracy')
grid.fit(X, y)

print('网格搜索-最佳度量值:', grid.best_score_)
print('网格搜索-最佳参数：', grid.best_params_)
print('网格搜索-最佳模型：', grid.best_estimator_)
```

初用sklearn

□ 模型预测：predict()方法

- estimator.predict(X_test)：在测试集上预测
- estimator.predict(X_train)：在训练集上预测

```
y_pred = knn.predict(X_test)
print("X_test set predictions:\n {}".format(y_pred))
```

```
X_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]
```

```
knn.predict([X_test[0]])
```

```
array([2])
```

```
knn.predict_proba([X_test[0]])
```

```
array([[0., 0., 1.]])
```

初用sklearn

转换器(Transformer)

- 实现两类接口【fit() + transform()】，有两大类：

➢ 将分类型变量 (categorical) 编码成数值型变量 (numerical)

LabelEncoder \ OrdinalEncoder

➢ 规范化 (normalize) 或标准化 (standardize) 数值型变量

MinMaxScaler\StandardScaler

初用 sklearn

◆ 示例：特征缩放预处理

```
from sklearn.preprocessing import StandardScaler
X = [[0, 15],
     [1, -10]]
StandardScaler().fit(X).transform(X)
```

```
array([[-1.,  1.],
       [ 1., -1.]])
```

初用sklearn

流水线(Pipeline)

将多个估计器对象“连在一起”或“并在一起”使用

例如，两种形式流水线 (pipeline)

① 任意转换器序列

② 任意转换器序列 + 估计器

示例：构建管道对象

```python
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# create a pipeline object
pipe = make_pipeline(
StandardScaler(), KNeighborsClassifier(n_neighbors=5) )

# load the iris dataset and split it into train and test sets
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# fit the whole pipeline
pipe.fit(X_train, y_train)

# we can now use it like any other estimator
accuracy_score(pipe.predict(X_test), y_test)
```

```
0.9736842105263158
```

初用sklearn

sklearn建模流程

准备数据 → 选择模型 → 训练模型 评估模型 调整参数 → 模型预测

准备数据：
- 数据处理
- 特征工程
- ……

选择模型：
- 分类模型
- 回归模型
- 聚类模型
- ……

训练模型 评估模型 调整参数：
- 根据经验设定参数
- 交叉验证确定最优参数

模型预测：
- 预测
- 识别
- ……

案例-鸢尾花分类

建立鸢尾花分类模型

三种鸢尾花类型

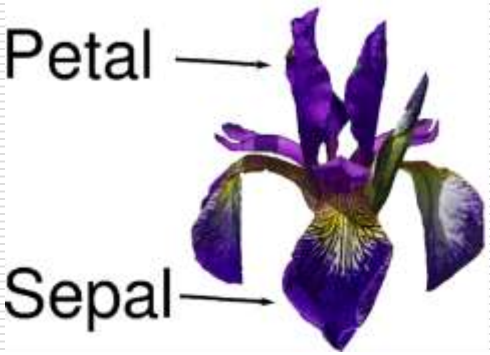
山鸢尾花 Setosa、变色鸢尾花 Versicolor、韦尔吉尼娅鸢尾花 Virginica

案例-鸢尾花分类

iris数据集加载

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# 加载 iris 数据集
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```



案例-鸢尾花分类

初识数据

```
print("First five rows of data:\n{}".format(iris_dataset['data'][:5]))
```

```
First five rows 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]]
```

```
print("Type of target: {}".format(type(iris_dataset['target'])))
```

```
Type of target: <class 'numpy.ndarray'>
```

```
print("Shape of target: {}".format(iris_dataset['target'].shape))
```

```
Shape of target: (150,)
```

```
print("Keys of iris_dataset: {}".format(iris_dataset.keys()))
```

```
Keys of iris_dataset: dict_keys(['feature_names', 'data', 'target', 'DESCR', 'target_names'])
```

```
print("Feature names: {}".format(iris_dataset['feature_names']))
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
print("Shape of data: {}".format(iris_dataset['data'].shape))
```

```
Shape of data: (150, 4)
```

```
print("Type of data: {}".format(type(iris_dataset['data'])))
```

```
Type of data: <class 'numpy.ndarray'>
```

```
print("Target names: {}".format(iris_dataset['target_names']))
```

```
Target names: ['setosa' 'versicolor' 'virginica']
```

```
print("Target:\n{}".format(iris_dataset['target']))
```

```
Target:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

案例-鸢尾花分类

数据集分割

```
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: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
```

```
X_train shape: (112, 4)
y_train shape: (112,)
```

```
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```

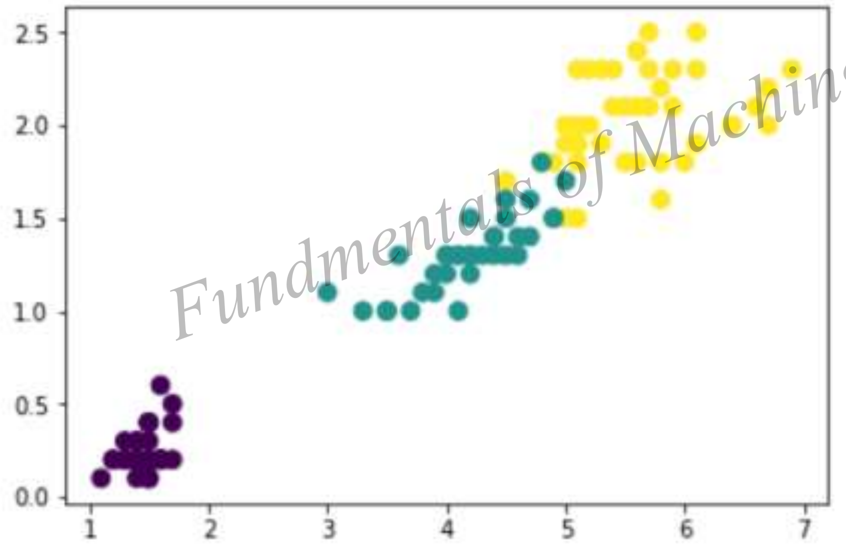
```
X_test shape: (38, 4)
y_test shape: (38,)
```

案例-鸢尾花分类

- 探索数据：散点图

```
plt.scatter(X_train[:, 2], X_train[:, 3], s=60, c=y_train, marker='o')
```

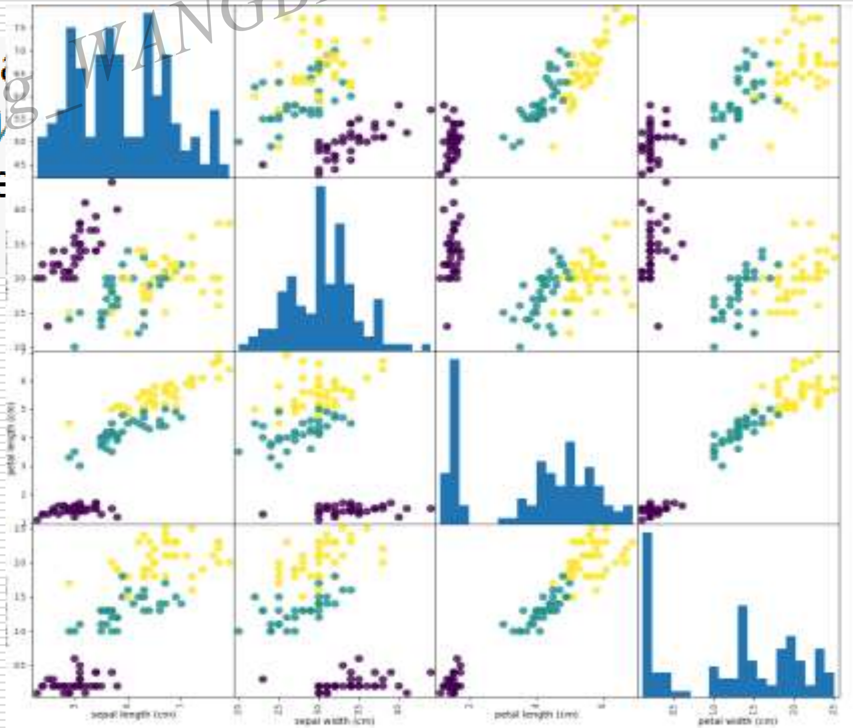
<matplotlib.collections.PathCollection at 0x13416b00>



案例-鸢尾花分类

□ 探索数据：绘制散点图矩阵

```
iris_dataframe = pd.DataFrame(X_train, columns=iris_da
# create a scatter matrix from the dataframe, color by y
grr = pd.scatter_matrix(iris_dataframe, c=y_train, figsize
                        hist_kwds={'bins': 20}, s=60, alpha=.8)
```



案例--鸢尾花分类

构建模型：kNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

案例-鸢尾花分类

☐ 评估模型：kNN

```
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
print("Test set score: {:.2f}".format(np.mean(y_pred == y_test)))
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
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]
Test set score: 0.97
Test set score: 0.97
```

案例-鸢尾花分类

模型预测

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
```

```
X_new.shape: (1, 4)
```

```
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target name: {}".format(
      iris_dataset['target_names'][prediction]))
```

```
Prediction: [0]
Predicted target name: ['setosa']
```

案例-鸢尾花分类

sklearn建模通用方法

- 训练模型：**estimator.fit(X_train, y_train)**
- 模型预测：**estimator.predict(X_new)、 predict_proba(X_new)**
- 模型评估：**stimator.score(X_test, y_test)**

模型评估方法

- **误差(error)**：预测值与样本真值间的差异（残差），也叫损失
- **经验误差(empirical error)**：在训练集上的误差，也称训练误差
- **泛化误差(generalization error)**：对未知数据的预测能力，通过在测试集上的误差评价
- **目标学习器**：泛化误差小的模型

模型评估方法

- **欠拟合(Under Fitting)**

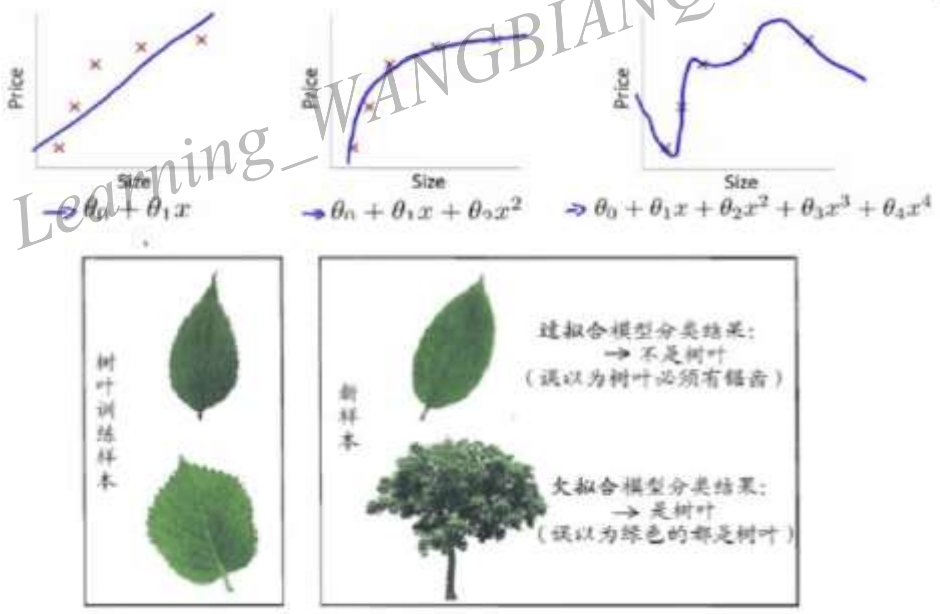
  对样本的一般性质尚未学好，

  模型的准确性不高

- **过拟合(Over Fitting)**

  表现为模型有更多参数，

  导致泛化能力下降



过拟合、欠拟合的直观类比

源自周志华《机器学习》

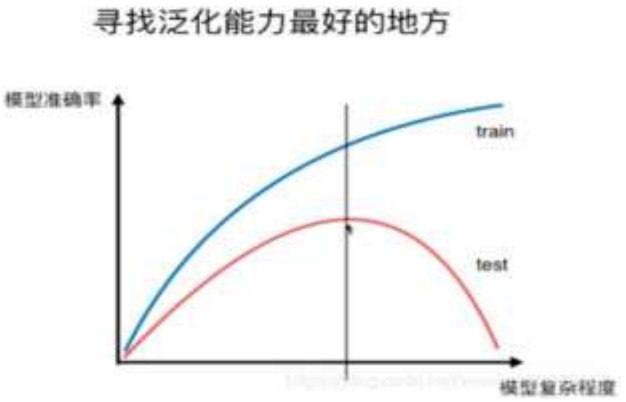
模型评估方法

模型泛化能力

- 训练误差：训练集的平均损失
- 测试误差：测试集的平均损失

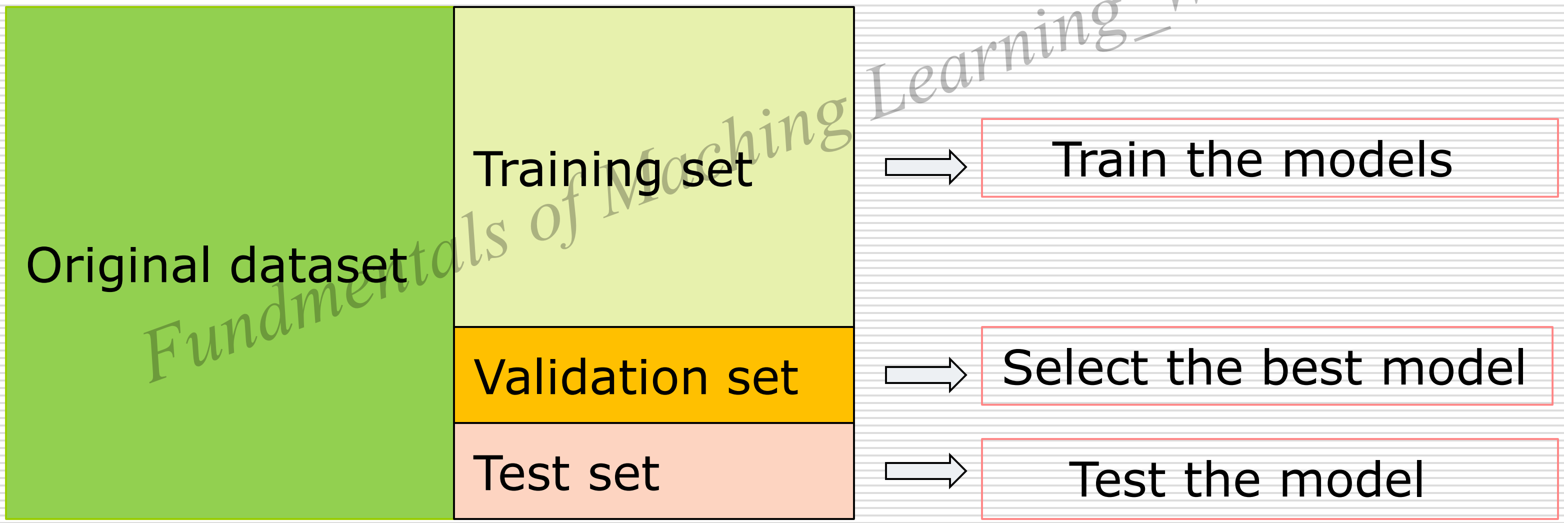
防止过拟合

- 数据集扩增
- 正则化限制模型复杂度
- Early Stopping、dropout
- 使用集成方法



模型评估方法

数据集分割：训练集 vs 验证集 vs 测试集



模型评估方法

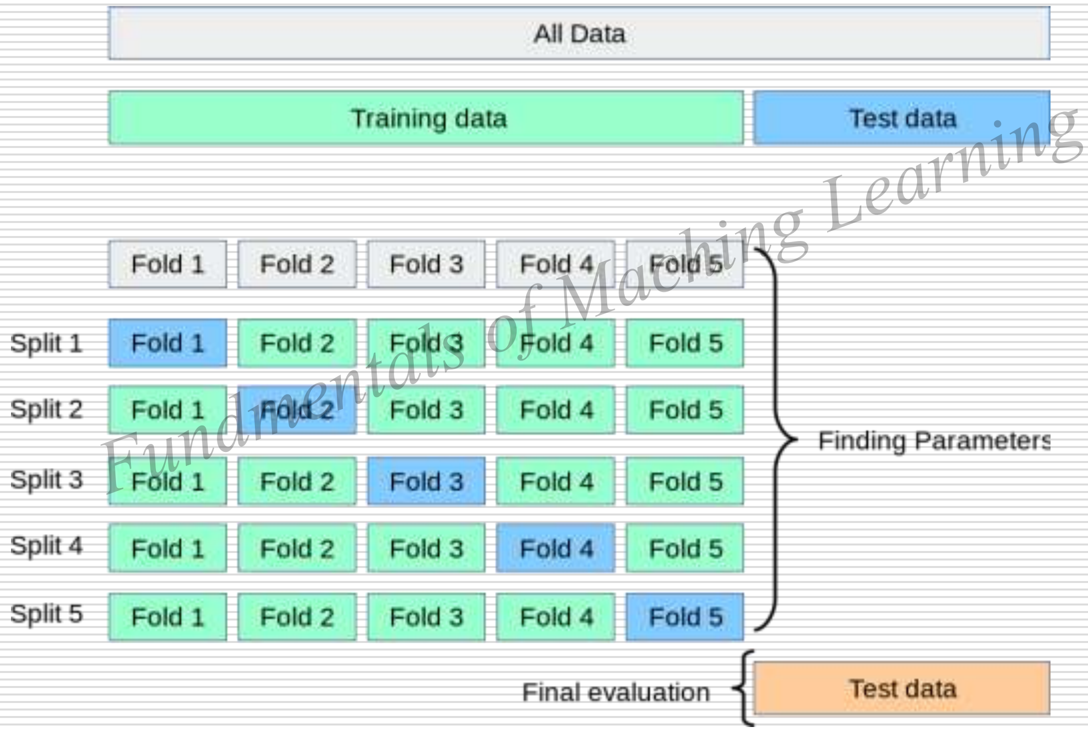
- **留出法**：直接将数据集***D***划分为两个互斥的集合，其中一个集合作为训练集***S***，另一个作为测试集***T***，即：

$$D = S \cup T,\ S \cap T = \Phi$$

  - 利用***S***训练模型，利用***T***评估测试误差，作为对泛化误差的评估
  - 通常，约**2/3**数据做训练集，其余**1/3**数据做测试集

模型评估方法

- S 折交叉验证(S-fold cross validation)：



返回平均结果

from sklearn.model_selection import cross_val_score

模型评估方法

模型选择哲学：

- 没有免费午餐定律(**No Free Lunch Theorem，NFL**）
- 奥卡姆剃刀定律(**Occam‘s Razor, Ockham’s Razor**）

  如无必要，勿增实体”即“简单有效原理”
- 集成学习(**Ensemble Learning**

模型评估方法

模型选择与评估

https://scikit-learn.org/stable/model_selection.html#model-selection

- Cross-validation: evaluating estimator performance
- Tuning the hyper-parameters of an estimator
- Metrics and scoring: quantifying the quality of predictions
- Model persistence
- Validation curves: plotting scores to evaluate models

模型评估指标

- 分类指标（Classification metrics）
- 回归指标（Regression metrics）
- 聚类指标（Clustering metrics）
- ……

模型评估指标

- **准确率**(accuracy)：对于给定的测试数据集，分类器正确分类的样本数与总样本数之比
  - 假设有100个样本，有99个正样本，一个负样本，模型将100个样本都判为正样本，请问模型的准确率是多少？

    这样的场景有：信用卡欺诈检测，离职员工检测等
  - 准确率指标的缺陷？

    未考虑数据不平衡性，将每个类同等对待

模型评估指标

- **sklearn.metrics.***

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

```
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = [0, 2, 1, 3]
y_true = [0, 1, 2, 3]

print(accuracy_score(y_true, y_pred, normalize=False))
print(accuracy_score(y_true, y_pred))
```

```
2
0.5
```

模型评估指标

- **二元分类模型的评估**：假设只有两类样本，即正例(**positive**)和负例(**negative**)。 通常以关注的类为正类，其他类为负类。

混淆矩阵(Confusion matrix)

真实情况	预测结果	
	正例	反例
正例	*TP*（真正例）	*FN*（假反例）
反例	*FP*（假正例）	*TN*（真反例）

```python
from sklearn.metrics import confusion_matrix
y_true = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 0, 0, 0, 0, 1, 1, 1]

confusion_matrix(y_true, y_pred)
```

```
array([[4, 1],
       [2, 3]], dtype=int64)
```

TP(true positive), FP(false positive)
TN(true negative), FN(false negative)

模型评估指标

- **精确率(precision)和召回率(recall)：**
  - 精确率： $p = \dfrac{TP}{TP+FP}$
  - 召回率： $R = \dfrac{TP}{TP+FN}$
  - 精确率和召回率的均值： $F_1 = \dfrac{2TP}{2TP+FP+FN}$

模型评估指标

- **分类报告(Classification report)**：显示每个类的分类性能包括每个类别的精确率、召回率、F1值等

sklearn.metrics.classification_report

```
from sklearn.metrics import classification_report
y_true = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 0, 0, 0, 0, 1, 1, 1]
target_names = ['class 0', 'class 1', 'class 2']

print(classification_report(y_true, y_pred, target_names=target_names))
```

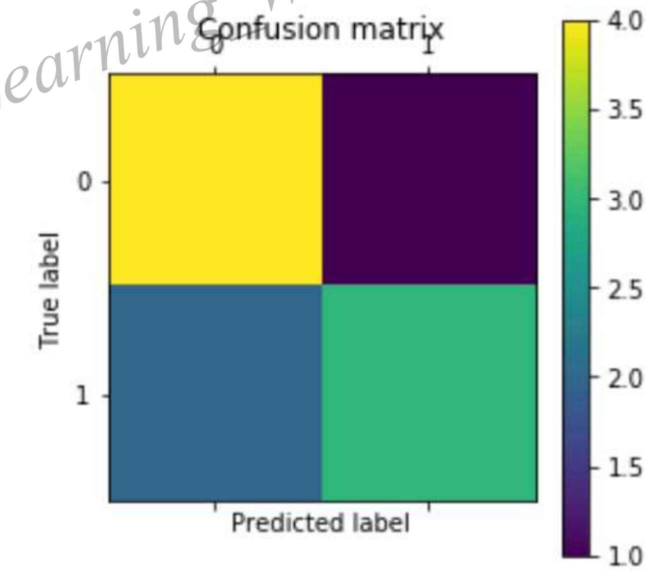
```
             precision    recall  f1-score   support

    class 0       0.67      0.80      0.73         5
    class 1       0.75      0.60      0.67         5

avg / total       0.71      0.70      0.70        10
```

模型评估指标

- **混淆矩阵可视化**：热图(**heatmap**)直观地展现类别的混淆情况

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
%matplotlib inline
y_test = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 0, 0, 0, 0, 1, 1, 1]
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
plt.matshow(confusion_matrix)
plt.title('Confusion matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
[[4 1]
 [2 3]]
```



模型评估指标

混淆矩阵可用于多元分类模型的评价

如果是多分类的呢？举一个三分类的例子：

Confusion Matrix		Predict		
		0	1	2
Real	0	a	b	c
	1	d	e	f
	2	g	h	i

$$Specificity_{class0} = \frac{e+i}{d+e+g+i},$$

$$Sensitivity_{class0} = Recall_{class0} = \frac{a}{a+b+c}$$

$$Precision_{class0} = \frac{a}{a+d+g},$$

```
from sklearn.metrics import confusion_matrix
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]

confusion_matrix(y_true, y_pred)
```

```
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
```

模型评估指标

混淆矩阵可用于多元分类模型的评价

```
from sklearn.metrics import classification_report
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]
target_names = ['class 0', 'class 1', 'class 2']

print(classification_report(y_true, y_pred, target_names=target_names))
```

```
             precision    recall  f1-score   support

    class 0       0.67      1.00      0.80         2
    class 1       0.00      0.00      0.00         1
    class 2       0.67      0.67      0.67         3

avg / total       0.56      0.67      0.60         6
```

模型评估指标

二元分类评估指标：ROC和AUC

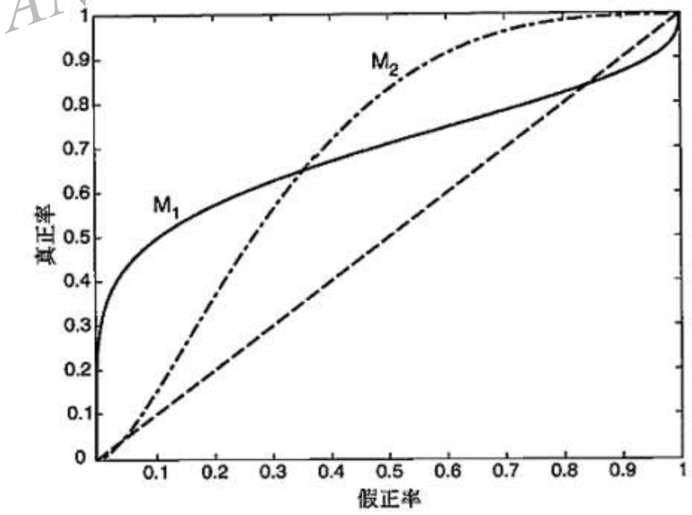
ROC(receiver operating characteristic curve)：

- TPR和FPR 之间的一种图形化表示
- 每条曲线对应一个分类模型

AUC(area under curve)：模型曲线下面的阴影面积与理想模型曲线下面阴影面积的比值

AUC越接近于1，表示模型预测能力越高

- TPR = TP/(TP + FN) = R
- FPR = FP/(TN + FP)



3个不同分类器的ROC曲线

模型评估指标

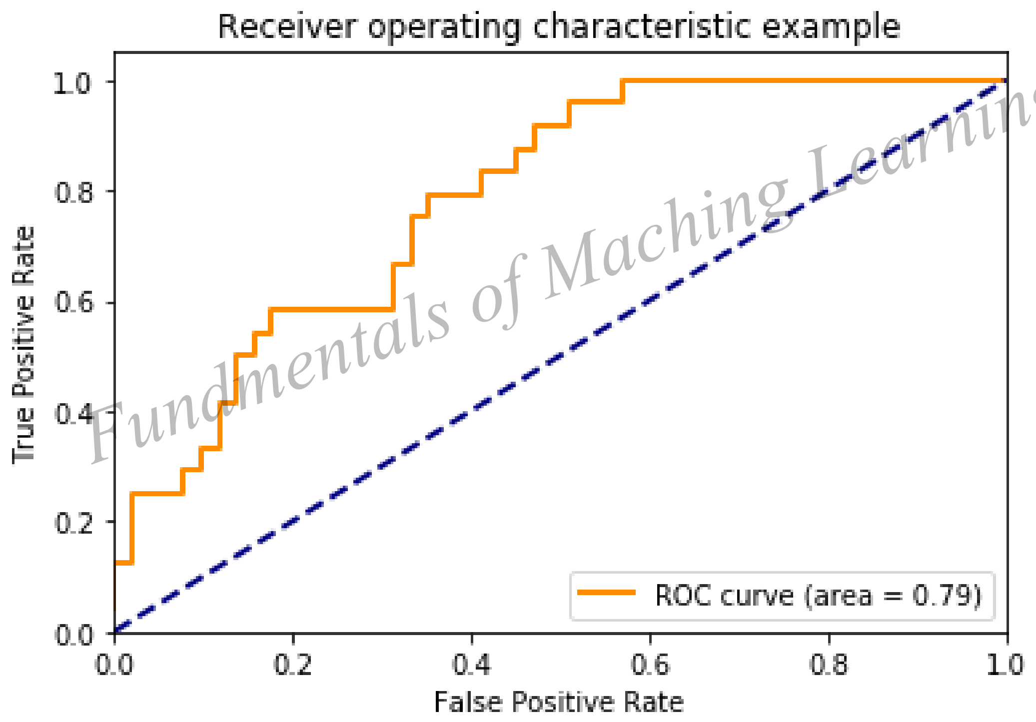
sklearn.metrics中的ROC曲线

```python
# roc_curve计算ROC曲线
import numpy as np
from sklearn.metrics import roc_curve
y = np.array([1, 1, 2, 2])
scores = np.array([0.1, 0.4, 0.35, 0.8])
FPR, TPR, thresholds = roc_curve(y, scores, pos_label=2)

print(FPR)
print(TPR)
print(thresholds)
```

```
[0.  0.5 0.5 1. ]
[0.5 0.5 1.  1. ]
[0.8  0.4  0.35 0.1 ]
```

模型评估指标

◆ 示例：**sklearn.metrics中的ROC曲线**



模型评估指标

- sklearn.metrics.roc_auc_score

```python
# roc_auc_score函数计算ROC曲线之下的面积，它也被称为AUC或AUROC
# 通过计算之下的面积曲线信息被归一化到1内
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])

print(roc_auc_score(y_true, y_scores))
```

```
0.75
```

模型评估指标

sklearn中的回归模型

- Linear Regression
- Decision Tree Regressor
- SVM Regressor
- K Neighbors Regressor
- Random Forest Regressor
- Adaboost Regressor
- Gradient Boosting Random Forest Regressor
- bagging Regressor
- ExtraTree Regressor

模型评估指标

sklearn中的回归模型部分指标

- 平均绝对误差：metrics.mean_absolute_error
- 均方误差：metrics.mean_squared_error
- 中位数绝对误差：metrics.median_absolute_error
- 解释方差分：metrics.explained_variance_score
- R方得分：metrics.r2_score

通常尽量保持均方误差最低，而且解释方差分最高

模型评估指标

R2：回归平方和占总离差平方和的比例

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- 反映回归直线的拟合程度，取值范围在 [0 , 1] 之间
- $R^2 \to 1$，说明回归方程拟合的越好
- $R^2 \to 0$，说明回归方程拟合的越差
- 判定系数等于相关系数的平方，即$R^2 = (r)^2$

模型评估指标

sklearn.metrics.r2_score

```python
from sklearn.metrics import r2_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
print(r2_score(y_true, y_pred))

y_true = [[0.5, 1], [-1, 1], [7, -6]]
y_pred = [[0, 2], [-1, 2], [8, -5]]
print(r2_score(y_true, y_pred, multioutput=

y_true = [[0.5, 1], [-1, 1], [7, -6]]
y_pred = [[0, 2], [-1, 2], [8, -5]]
```

```python
print(r2_score(y_true, y_pred, multioutput='uniform_average'))
print(r2_score(y_true, y_pred, multioutput='raw_values'))
print(r2_score(y_true, y_pred, multioutput=[0.3, 0.7]))
```

```
0.9486081370449679
0.9382566585956417
0.9368005266622779
[0.96543779 0.90816327]
0.9253456221198156
```

模型评估指标

sklearn.metrics.explained_variance_score

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

```
# 解释方差值: Explained variance score

from sklearn.metrics import explained_variance_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
print(explained_variance_score(y_true, y_pred))

y_true = [[0.5, 1], [-1, 1], [7, -6]]
y_pred = [[0, 2], [-1, 2], [8, -5]]
print(explained_variance_score(y_true, y_pred, multioutput='raw_values'))
print(explained_variance_score(y_true, y_pred, multioutput=[0.3, 0.7]))
```

```
0.9571734475374732
[0.96774194 1.        ]
0.9903225806451612
```

偏差与方差均衡

- **模型的期望泛化误差分解：**

$$E(f:D)=bias^2(x)+\mathrm{var}(x)+\varepsilon^2$$

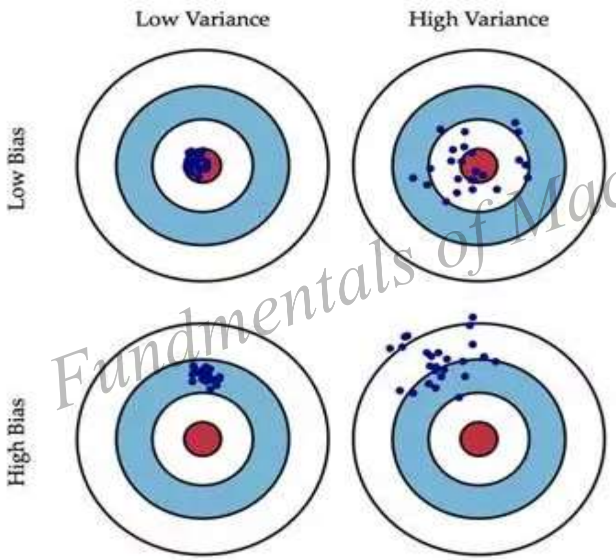
即，泛化误差分为偏差、方差与噪声之和

泛化性能由学习算法，数据的充分性以及学习任务本身的难度共同决定

源自周志华《机器学习》

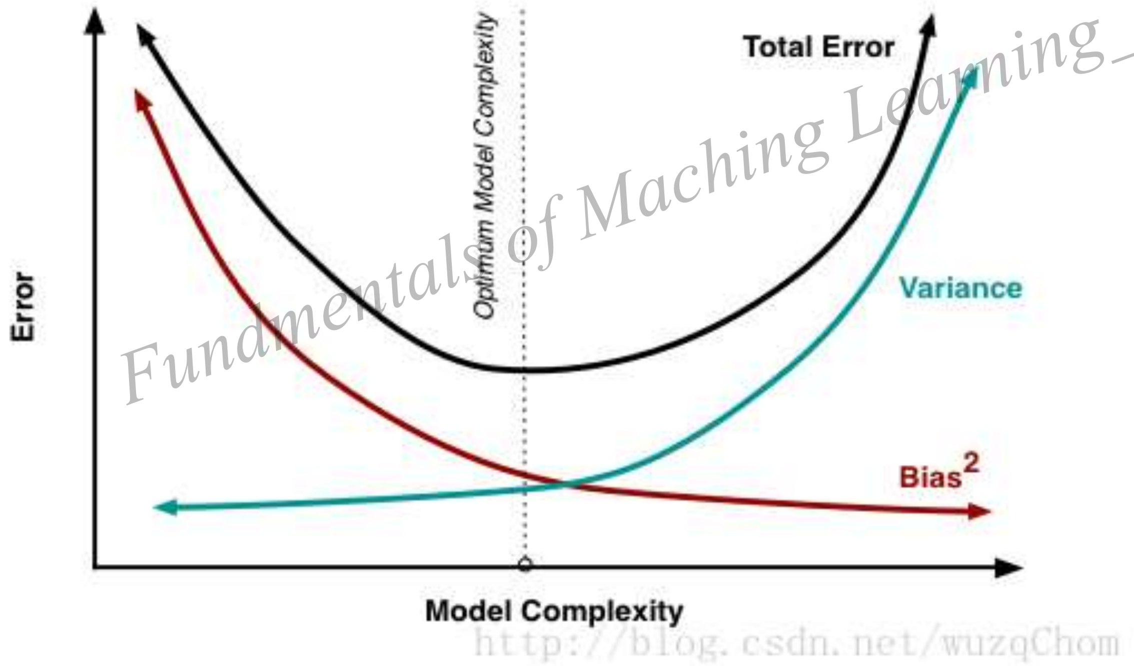
模型评估方法

偏差与方差



偏差与方差均衡

- 均衡偏差与方差



偏差与方差均衡

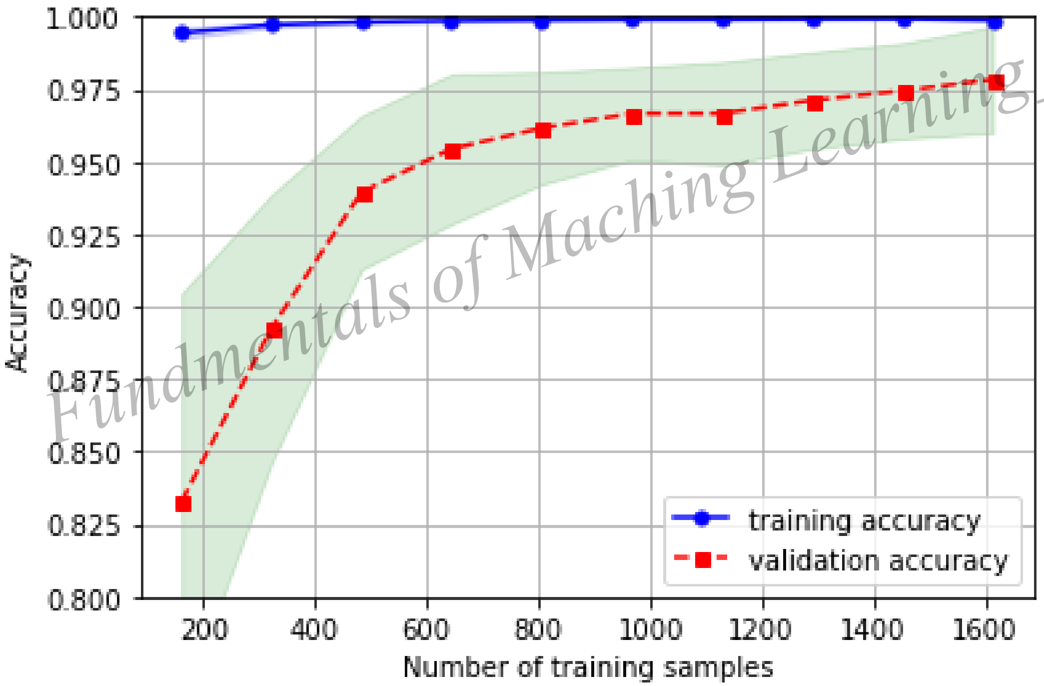
学习曲线(learning curve)

sklearn.model_selection.learning_curve(estimator, X, y, *, groups=None, train_sizes=array([0.1, 0.33, 0.55, 0.78, 1.]), cv=None, scoring=None, exploit_incremental_learning=False, n_jobs=None, pre_dispatch='all', verbose=0, shuffle=False, random_state=None, error_score=nan, return_times=False)[source]
Learning curve.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html#sklearn.model_selection.learning_curve

偏差与方差均衡

◆ 示例：学习曲线



偏差与方差均衡

验证曲线(Validation curve)

sklearn.model_selection.validation_curve(estimator, X, y, *, param_name, param_range, groups=None, cv=None, scoring=None, n_jobs=None, pre_dispatch='all', verbose=0, error_score=nan)

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.validation_curve.html#sklearn.model_selection.validation_curve

偏差与方差均衡

◆ **示例：验证曲线**