机器学习与深度学习

——sklearn入门

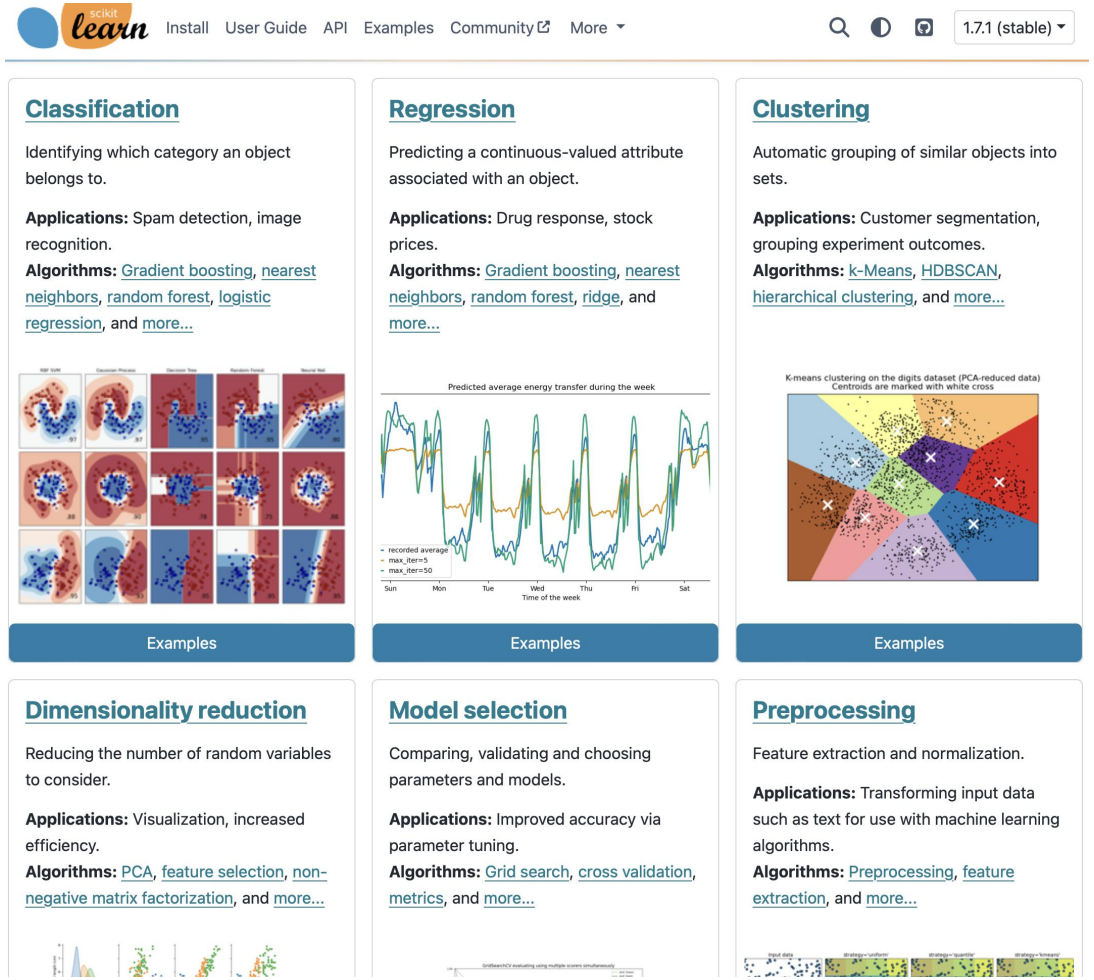
Personal Website: https://www.miaopeng.info/
Email: miaopeng@stu.scu.edu.cn
Github: https://github.com/MMeowhite
Youtube: https://www.youtube.com/@pengmiao-bmm

目录章节

CONTENTS

sklearn是什么？

- sklearn：Python中最常用的机器学习工具包（官网：https://scikit-learn.org/stable/）。

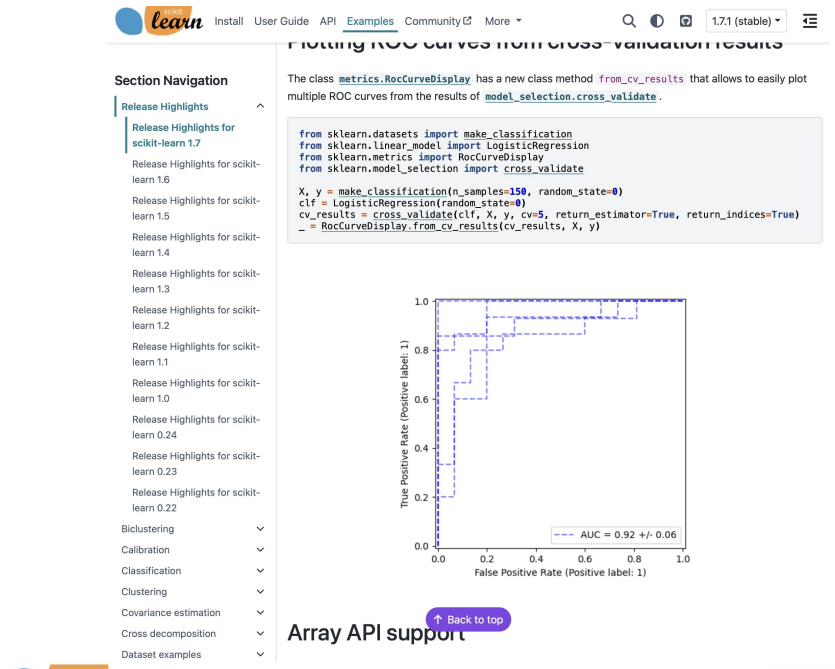


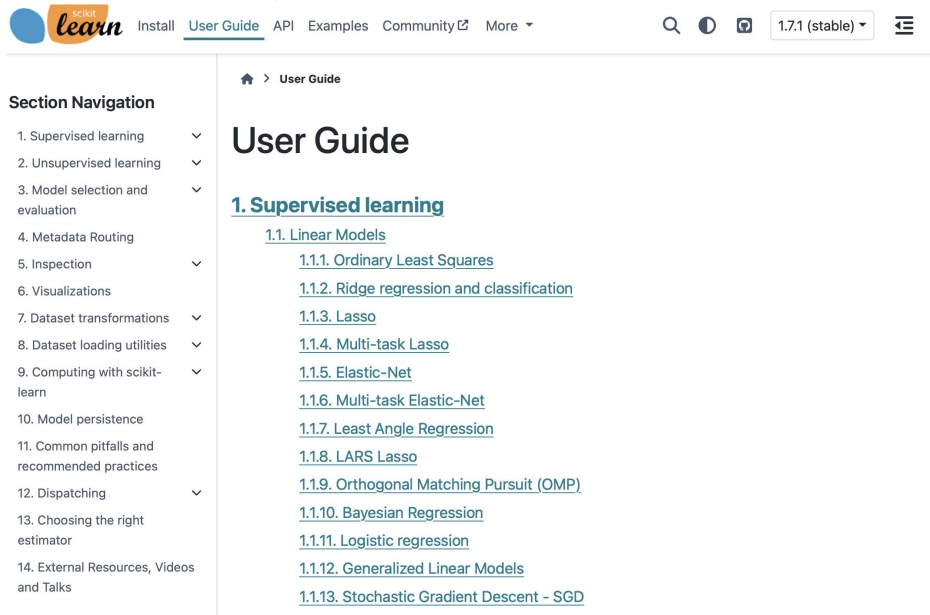
- 用于预测性数据分析的简单高效工具。
- 人人可用，可在多种场景下复用
- 基于 NumPy、SciPy 和 matplotlib 构建
- 开源，支持商业用途 —— BSD 许可证

Scikit-learn 是 Python 生态里最流行的机器学习工具，几乎所有入门/原型开发都会用到它。

sklearn的特点

特点	说明	示例代码片段
统一简洁的API	训练不同模型调用接口几乎一样，易学易用	model = LogisticRegression() model.fit(X_train, y_train)
丰富的算法库	多种分类、回归、聚类算法一应俱全	from sklearn.ensemble import RandomForestClassifier, ...
良好的文档支持	官方文档详尽，示例代码丰富，入门无压力	scikit-learn.org
高效且可扩展	基于 NumPy/SciPy，速度快，能处理大规模数据	from sklearn.preprocessing import StandardScaler，用于数据标准化





sklearn能做什么？

- sklearn主要用于**分类、回归、聚类、降维、模型选择、特征工程**：

scikit-learn Install User Guide API Examples Community More 1.7.1 (stable)

Section Navigation

1.1. Linear Models

The following are a set of methods intended for regression in which the target value is expected to be a linear combination of the features. In mathematical notation, if $\hat{y}$ is the predicted value.

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

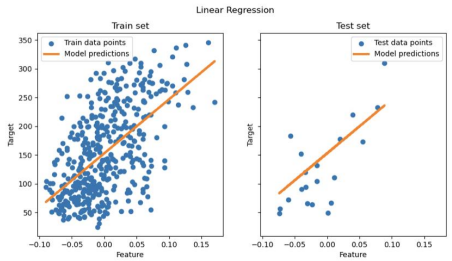
Across the module, we designate the vector $w = (w_1, \ldots, w_p)$ as `coef_` and w_0 as `intercept_`.

To perform classification with generalized linear models, see Logistic regression.

1.1.1. Ordinary Least Squares

`LinearRegression` fits a linear model with coefficients $w = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w ||Xw - y||_2^2$$

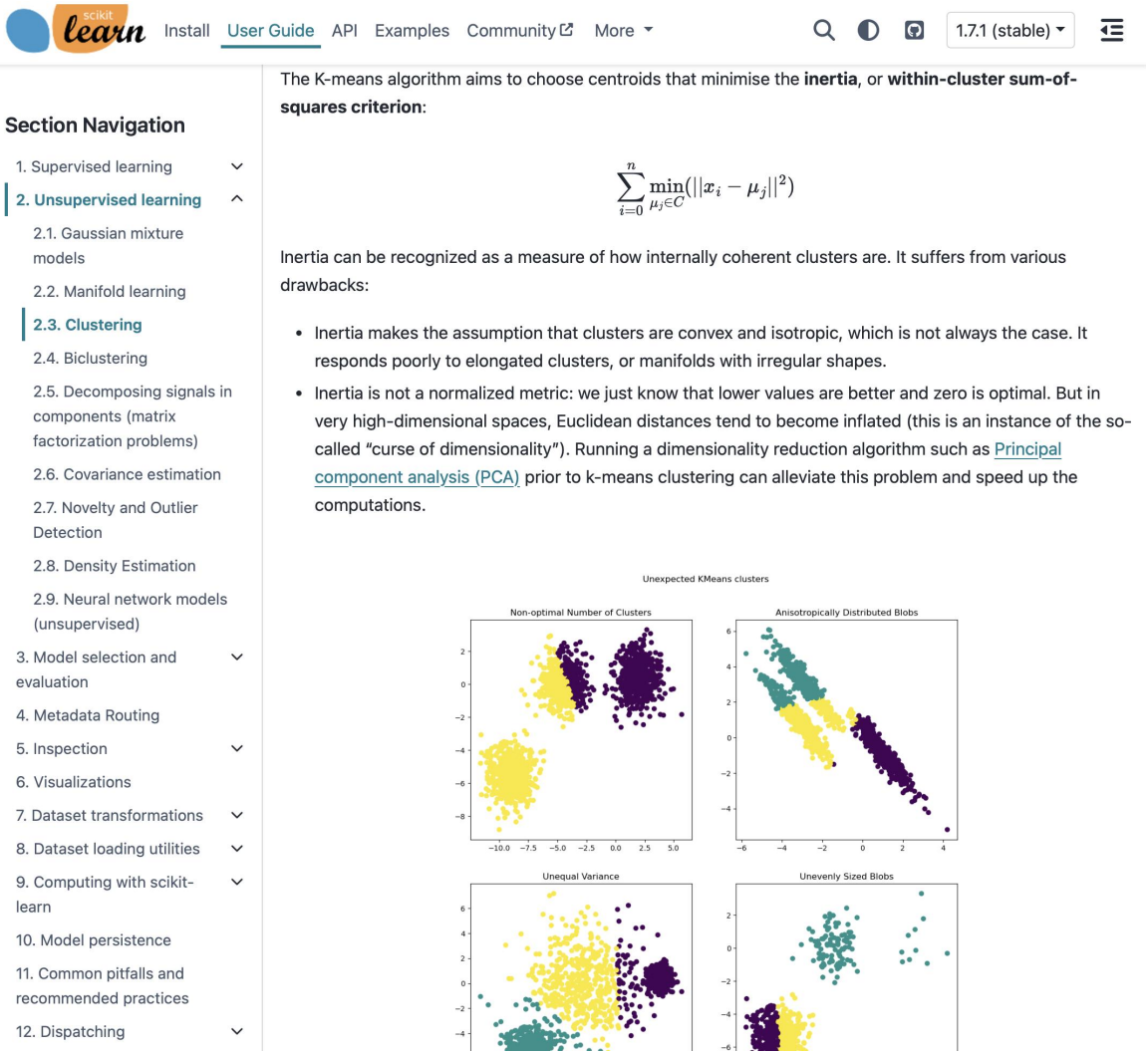


`LinearRegression` takes in its `fit` method arguments `X`, `y`, `sample_weight` and stores the coefficients w of the linear model in its `coef_` and `intercept_` attributes:

- 分类：
  - 将数据划分到不同类别，例如用**逻辑回归**预测一封邮件是不是垃圾邮件。
- 回归：
  - 预测连续数值，例如用**线性回归**预测房价。

sklearn能做什么？

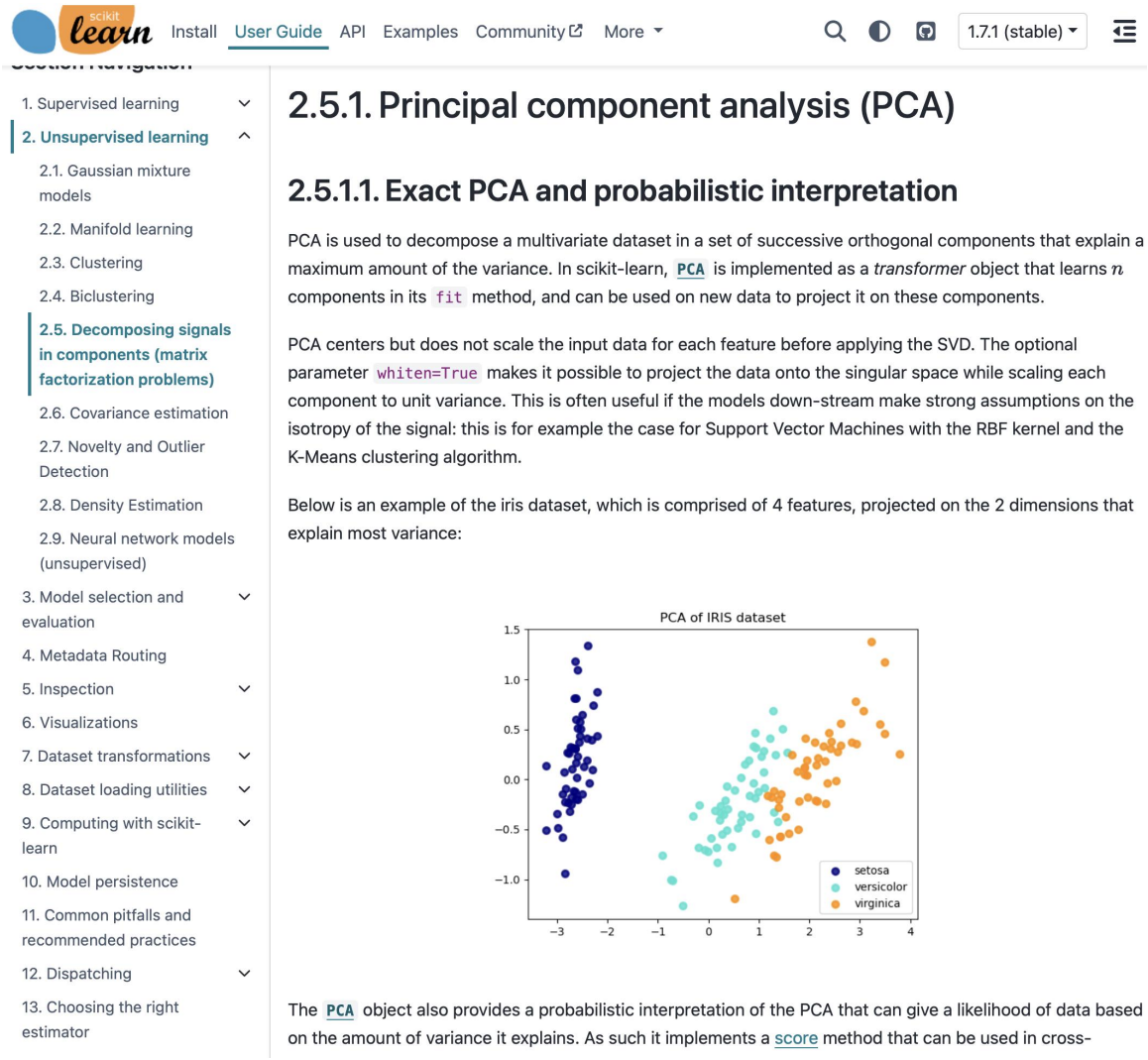
- sklearn主要用于**分类、回归、聚类、降维、模型选择、特征工程**：



- 聚类：
  - 无监督地把数据分组，例如用 K-Means 把用户分为不同群体。

sklearn能做什么？

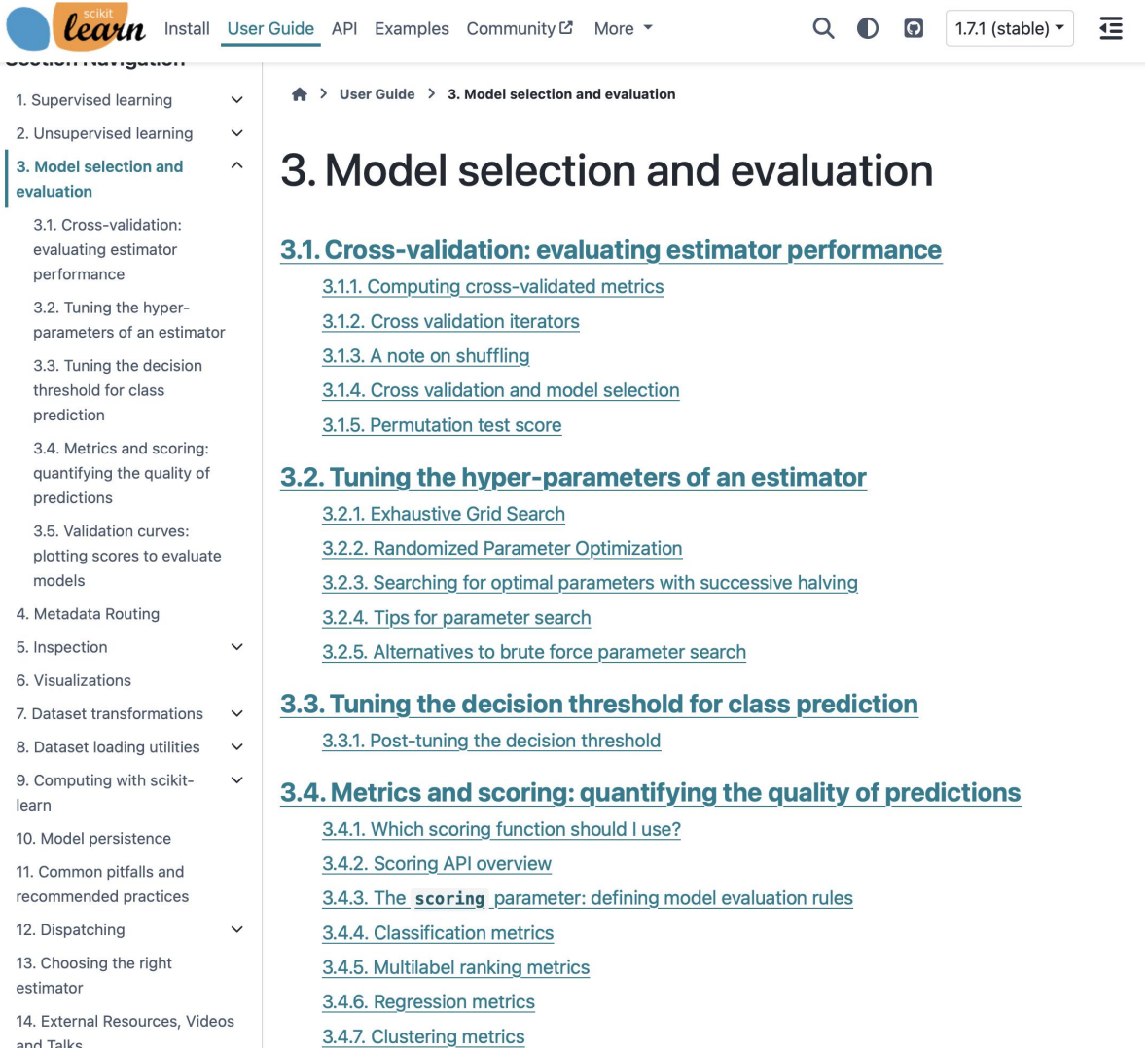
- sklearn主要用于**分类、回归、聚类、降维、模型选择、特征工程**：



- 降维：
  - 压缩特征数量，例如用PCA提取数据主要成分，方便可视化或建模。

sklearn能做什么？

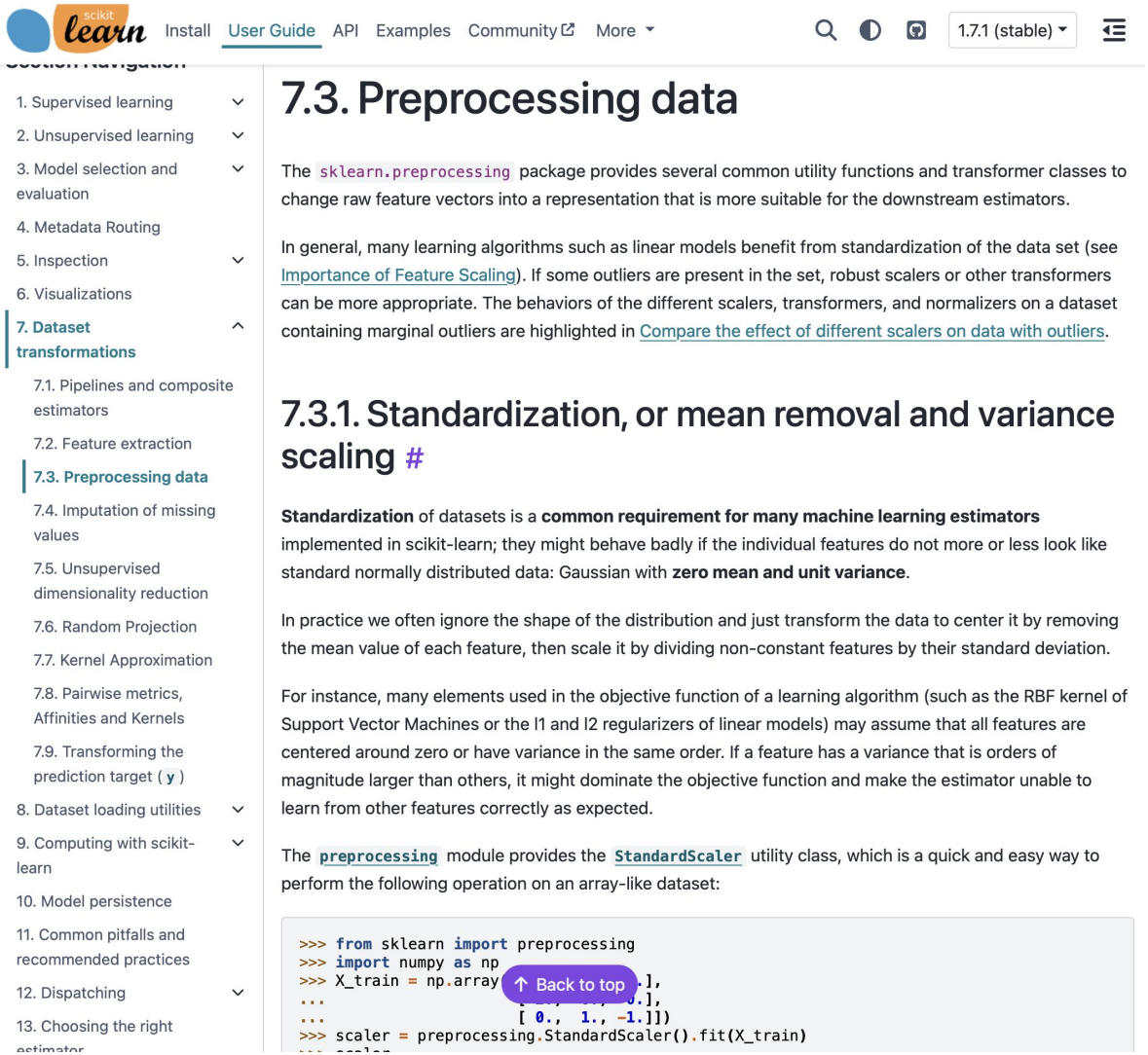
- sklearn主要用于**分类、回归、聚类、降维、模型选择、特征工程**：



- 模型选择：
  - 自动选择最优参数或模型，例如用 GridSearchCV 做超参数调优。

sklearn能做什么？

- sklearn主要用于**分类、回归、聚类、降维、模型选择、特征工程**：



- 降维：
  - 数据预处理与特征构造，例如用 StandardScaler 标准化特征或 OneHotEncoder 做独热编码。

▶ 小结

- sklearn：1）Python中最常用的机器学习库；2）统一 API，上手容易；3）适合原型验证&基线模型。
- sklearn的特点：
  - 统一简洁的 API：不同模型调用方式类似。
  - 丰富的算法库：分类、回归、聚类、降维一应俱全。
  - 高效 & 易扩展：基于 NumPy/SciPy，运行快。
  - 完善的文档 & 社区：学习资料丰富。
- sklearn的应用：1）分类：垃圾邮件识别、图像分类；2）回归：房价预测、销售量预测；3）聚类：用户分群、市场细分；4）降维：PCA 数据压缩、可视化；5）模型选择：交叉验证、网格搜索；6）特征工程：数据预处理、特征构造。

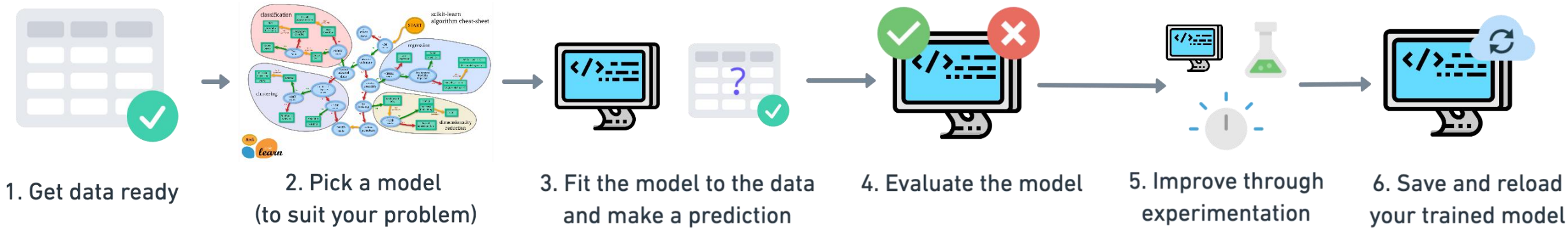
从数据到模型，Scikit-learn 带你走好机器学习第一步。

目录章节

CONTENTS

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型

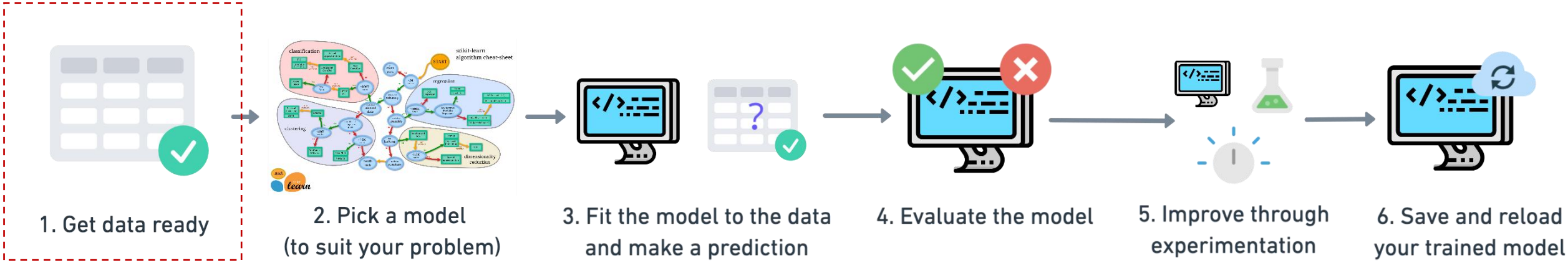


- 用 Scikit-learn 完成一个机器学习项目，就像走一条清晰的路线：先加载数据并完成预处理，然后选择合适的模型，进行训练并做出预测，再评估模型效果，如果效果不理想，可以通过调参和优化提升性能，最后，把满意的模型保存下来，方便后续直接加载使用。

用 Scikit-learn，你可以按流程完成机器学习：从数据预处理，到建模、评估、优化，再到模型保存与复用。

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



- 一切机器学习从数据开始：用 Scikit-learn 快速加载标准数据集或读取自有数据。

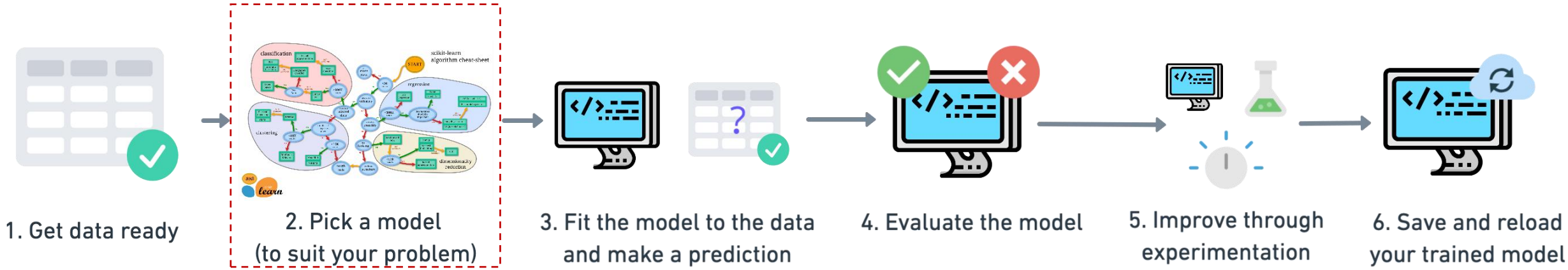
```python
from sklearn.datasets import load_iris
data = load_iris()
X, y = data.data, data.target
# 展示数据集的形状以及数据类型
print(f"特征数据形状：{X.shape}，标签数据形状：{y.shape}")
print(f"特征数据类型：{X.dtype}，标签数据类型：{y.dtype}")

# 展示前5个样本的特征数据和标签
print(f"前5个样本的特征数据：{X[:5]}，前5个样本的标签：{y[0:5]}")

# 展示特征名和标签名
print(f"特征名称：{data.feature_names}，\n标签名称：{data.target_names}")
```

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



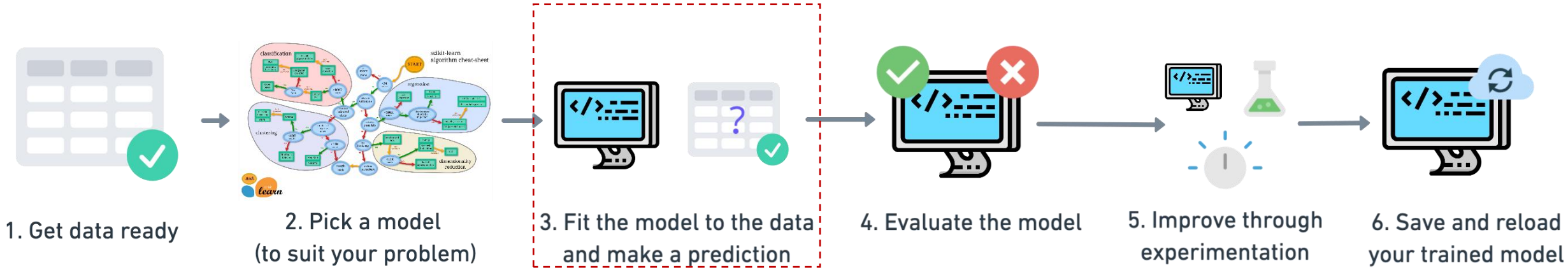
- 根据任务类型选择合适的算法，是机器学习建模的核心。

任务类型	常用模型
分类	LogisticRegression、RandomForestClassifier
回归	LinearRegression、GradientBoostingRegressor
聚类	KMeans、DBSCAN
降维	PCA、TruncatedSVD

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
print(f"模型的超参数: {model.get_params()}")
```

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



- 通过fit训练模型，用predict得到预测结果。

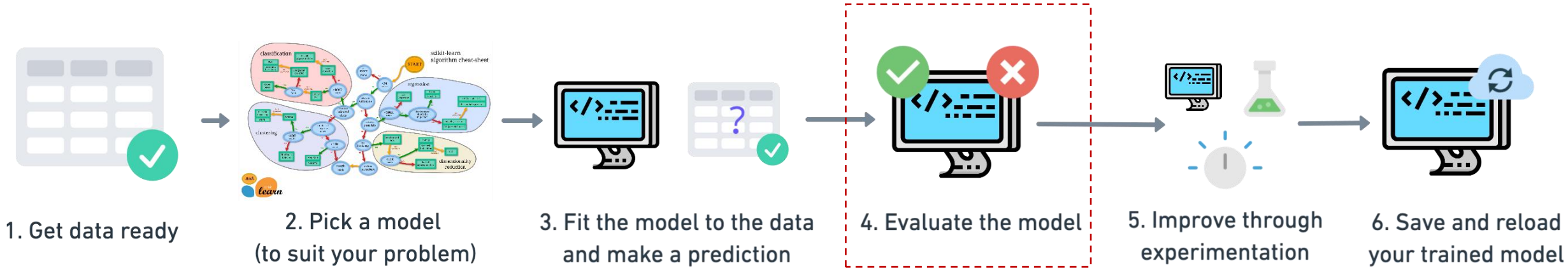
任务类型	常用模型
分类	LogisticRegression、RandomForestClassifier
回归	LinearRegression、GradientBoostingRegressor
聚类	KMeans、DBSCAN
降维	PCA、TruncatedSVD

```
# 训练模型
model.fit(X_train, y_train)

# 预测
y_pred = model.predict(X_test)
print(y_pred[:5])  # 打印前5个预测结果
```

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



- 没有评估，就不知道模型好坏，这时就需要用合适的指标量化模型表现。

任务类型	常用模型
分类	accuracy_score、precision、recall、f1_score
回归	mean_squared_error、r2_score
聚类	silhouette_score、adjusted_rand_score

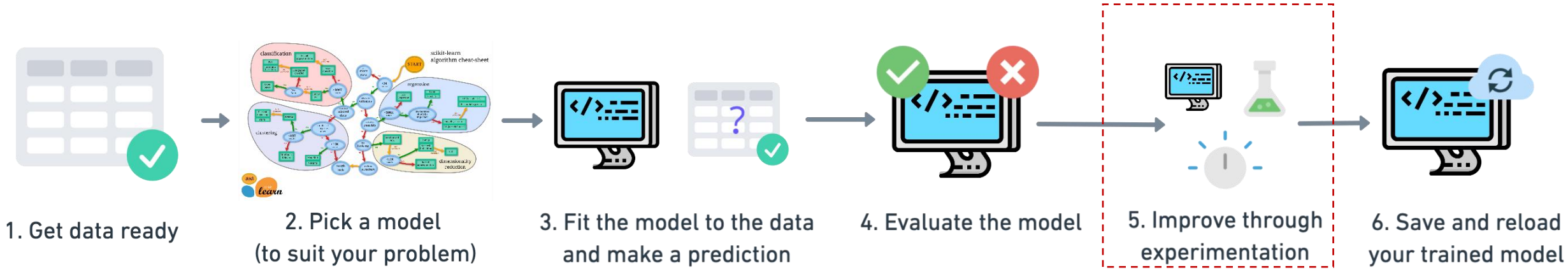
```python
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, classification_report

print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
print("F1 Score:", f1_score(y_test, y_pred, average='macro'))
print("Accuracy:", accuracy_score(y_test, y_pred))

# 综合以上的部分报告
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

sklearn的核心工作流

- sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



- 如果模型效果不好，可以调超参、换模型、集成模型、加特征等。

方法类别	常用手段	工具/函数示例
调参优化	网格搜索、随机搜索	GridSearchCV、RandomizedSearchCV
特征工程	特征选择、特征缩放、特征组合	SelectKBest、RFE、StandardScaler、PolynomialFeatures
集成学习	Bagging、Boosting、Stacking	RandomForestClassifier、GradientBoostingClassifier、StackingClassifier
数据优化	数据增强、过采样/欠采样	SMOTE（imbalanced-learn）、自定义数据增强方法

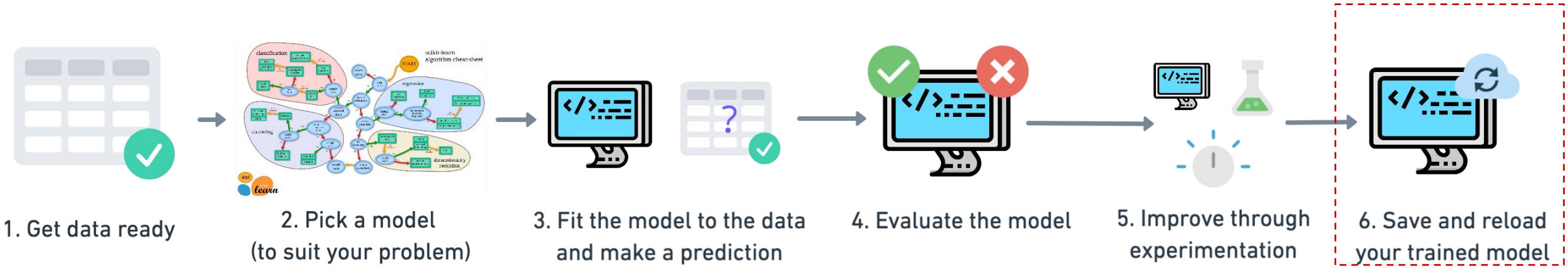
```python
from sklearn.model_selection import GridSearchCV
# 超参数调优，这里选择50、100、200个树作为超参数
param_grid = {'n_estimators': [50, 100, 200]}

# 使用网格搜索进行超参数调优，通过5折交叉验证
grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
# 训练模型
grid.fit(X_train, y_train)

# 输出最佳超参数
print("最佳参数：", grid.best_params_)
print("最佳得分：", grid.best_score_)
best_model = grid.best_estimator_
```

▶ sklearn的核心工作流

➢ sklearn的核心工作流程：1.加载数据与预处理 → 2.选择模型 → 3.训练与预测 → 4.评估 → 5.提升模型性能 → 6.保存与加载预训练模型



- 模型训练好了，别让它只活在内存里，保存下来才能复用与部署。

- 保存方法：1）joblib（推荐用于大模型/含 numpy 数组）；2）pickle（通用保存方法）

```python
import joblib

# 保存之前训练的最佳模型
joblib.dump(best_model, 'best_model.pkl')

# 加载模型
loaded_model = joblib.load('best_model.pkl')

# 使用加载的模型进行预测
y_pred_loaded = loaded_model.predict(X_test)
print("加载模型的预测结果：", y_pred_loaded[:5])  # 打印前5个预测结果
print("加载模型的准确率：", accuracy_score(y_test, y_pred_loaded))

print("\nClassification Report:\n", classification_report(y_test, y_pred_loaded))
```

▶ 小结

➢ 回顾一下：Sklearn 的精髓，其实就是一个清晰的机器学习流水线：

步骤	关键操作	常用工具
加载与预处理	导入数据、缺失值处理、特征缩放、分割数据（训练集/测试集）	pandas、StandardScaler、SimpleImputer、train_test_split
选择模型	分类/回归/降维/聚类	LogisticRegression、RandomForest、PCA
训练与预测	拟合模型、生成预测	model.fit()、model.predict()
评估模型	评估性能、找问题	accuracy_score、classification_report、r2_score
提升性能	调参、特征工程、集成学习	GridSearchCV、FeatureUnion、RandomForest
保存与加载	模型落地可用	joblib.dump()、joblib.load()

“Scikit-learn 的工作流程，就是把数据一步步送入：预处理 → 建模 → 训练 → 评估 → 优化 → 保存，形成一个可迭代的机器学习闭环。”

目录章节

CONTENTS

常用模块与算法概览

- Scikit-learn提供了从数据预处理到模型训练、评估与优化的全流程工具。它内置了丰富的算法，包括分类、回归、聚类、降维等常用方法。
- 通过统一的接口与模块化设计，快速搭建和验证机器学习模型变得高效便捷。

- **易用性**：统一的 API 设计，调用简单，降低上手门槛。
- **模块化**：涵盖预处理、建模、评估、优化等全流程，组合灵活。
- **算法丰富**：内置分类、回归、聚类、降维等主流算法，开箱即用。
- **高效稳定**：基于 NumPy/SciPy，性能优良，适合中小规模数据实验。
- **良好生态**：文档完善，社区活跃，易于与 Pandas、Matplotlib、深度学习框架配合。

参考资料：https://www.cnblogs.com/hongbao/p/17660040.html

▶ 常用模块与算法概览：数据预处理

- 数据预处理的包主要通过from sklearn.preprocessing import xxxx进行导入。
- 1）**标准化/归一化**
  - StandardScaler：均值=0，方差=1（常用于SVM、逻辑回归等）
  - MinMaxScaler：数据压缩到[0,1]区间（常用于神经网络、KNN）。

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- 2）**类别编码**
  - OneHotEncoder：将类别变量转换为独热向量（适合无序分类）
  - OrdinalEncoder：将类别映射为整数（适合有序分类）

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
X_encoded = encoder.fit_transform(X_categorical)
```

- 3）**缺失值处理**
  - SimpleImputer：填补缺失值（均值/中位数/众数）

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_filled = imputer.fit_transform(X)
```

▶ 常用模块与算法概览：特征选择

- 特征选择的包主要通过from sklearn.feature_selection import xxxx进行导入。
- 1）**过滤法（Filter）**
  - VarianceThreshold：去除低方差特征（信息量小）。
  - SelectKBest：根据统计指标（如卡方检验、F值）选择前 K 个特征。

```
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_
selector = SelectKBest(score_func=f_classif, k=10)
X_selected = selector.fit_transform(X, y)
```

- 2）**包装法（Wrapper）**
  - RFE（递归特征消除）：用模型迭代地去除权重较低的特征。

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
selector = RFE(model, n_features_to_select=5)
X_selected = selector.fit_transform(X, y)
```

- 3）**嵌入法（Embedded）**
  - L1正则化（Lasso）：通过稀疏化权重自动选择特征。
  - 树模型特征重要性。

```
from sklearn.linear_model import Lasso
model = Lasso(alpha=0.01)
model.fit(X, y)
importance = model.coef_  # 非零权重对应的特征即为选中
```

▶ 常用模块与算法概览：分类算法

- 特征选择的包主要通过from sklearn.linear_model import xxxx, from sklearn.svm import xxxx, from sklearn.ensemble import xxxx进行导入。

- 1）**逻辑回归（Logistic Regression）**
  - 用于二分类问题，输出概率

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 2）**K近邻（K-Nearest Neighbors, KNN）**
  - 根据邻近样本投票分类

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 3）**支持向量机（SVM）**
  - 寻找最优超平面，支持线性与非线性分类。

```
from sklearn.svm import SVC
model = SVC(kernel='rbf')  # rbf核函数
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 4）**决策树（Decision Tree）**
  - 基于特征条件分支进行分类

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 5）**随机森林（Random Forest）**
  - 多棵决策树的集成，提高泛化能力
  - 树模型特征重要性。

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

▶ 常用模块与算法概览：回归算法

- 特征选择的包主要通过from sklearn.linear_model import xxxx, from sklearn.ensemble import xxxx进行导入。

- 1）线性回归（Linear Regression）
  - 预测连续变量的线性模型。

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 2）岭回归（Ridge Regression）
  - 加入L2正则化的线性回归，防止过拟合。

```
from sklearn.linear_model import Ridge
model = Ridge(alpha=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 3）Lasso回归（Lasso Regression）
  - 寻找最优超平面，支持线性与非线性分类。

```
from sklearn.linear_model import Lasso
model = Lasso(alpha=0.1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 4）随机森林回归（Random Forest Regressor）
  - 基于特征条件分支进行分类。

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- 5）梯度提升回归（Gradient Boosting Regressor）
  - 通过逐步减小残差提升模型性能。

```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor(n_estimators=100)
```

常用模块与算法概览：聚类/降维算法

- 特征选择的包主要通过from sklearn.cluster import xxxx, from sklearn.decomposition import xxxx, from sklearn.discriminant_analysis import xxxx进行导入。
- 1）聚类（Cluster）
  - K-Means：基于质心的划分算法，常用且简单。
  - DBSCAN：基于密度的聚类，能识别噪声和任意形状簇。
  - 层次聚类：通过建立层次树结构进行聚类，支持凝聚和分裂方式
- 2）降维（Decomposition）
  - 主成分分析（PCA）：线性降维，通过提取主成分保留最大方差。
  - 线性判别分析（LDA）：监督式降维，最大化类间距，适合分类任务。

```python
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3, random_state=42)
model.fit(X)
labels = model.labels_
```

```python
from sklearn.cluster import DBSCAN
model = DBSCAN(eps=0.5, min_samples=5)
labels = model.fit_predict(X)
```

```python
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3)
labels = model.fit_predict(X)
```

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
```

```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_reduced = lda.fit_transform(X, y)
```

常用模块与算法概览：模型选择与评估

- 特征选择的包主要通过from sklearn.model_selection import xxxx, from sklearn.metrics import xxxx进行导入。
- 1）**交叉验证（Cross validation）**
  - 多次划分数据训练和测试，评估模型稳定性。

```python
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
scores = cross_val_score(model, X, y, cv=5)
print("Cross-validation scores:", scores)
print("Mean score:", scores.mean())
```

- 2）**超参数调优（GridSearchCV）**
  - 穷举法搜索最佳参数组合。

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20
grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
```

- 3）**评估指标（Evaluation）**
  - 分类准确率（accuracy_score）
  - 均方误差（mean_squared_error）
  - 其他：精确率、召回率、F1分数等。

```python
from sklearn.metrics import accuracy_score, precision_score, recall_sc
from sklearn.metrics import mean_squared_error, r2_score

# 假设 y_test 是真实值，y_pred 是预测值

# 分类指标
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
print("F1 Score:", f1_score(y_test, y_pred, average='macro'))

# 回归指标
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

▶ 小结

➢ 掌握常用模块和算法，能高效完成不同任务的机器学习建模与实践：

模块	功能	常用方法/算法
preprocessing	数据预处理	StandardScaler（标准化）、MinMaxScaler（归一化）、OneHotEncoder、OrdinalEncoder、SimpleImputer（缺失值处理）
feature_selection	特征选择	VarianceThreshold（过滤法）、SelectKBest、RFE（递归特征消除）
linear_model	线性模型	LinearRegression、LogisticRegression、Ridge、Lasso
svm	支持向量机	SVC（分类）、SVR（回归）
neighbors	邻近算法	KNeighborsClassifier、KNeighborsRegressor
tree	决策树模型	DecisionTreeClassifier、DecisionTreeRegressor
ensemble	集成学习	RandomForestClassifier、RandomForestRegressor、GradientBoosting
cluster	聚类	KMeans、DBSCAN、AgglomerativeClustering
decomposition	降维	PCA（主成分分析）、LDA（线性判别分析）
model_selection	模型选择与验证	train_test_split、cross_val_score、GridSearchCV
metrics	模型评估	accuracy_score、precision_score、recall_score、f1_score、mean_squared_error、r2_score

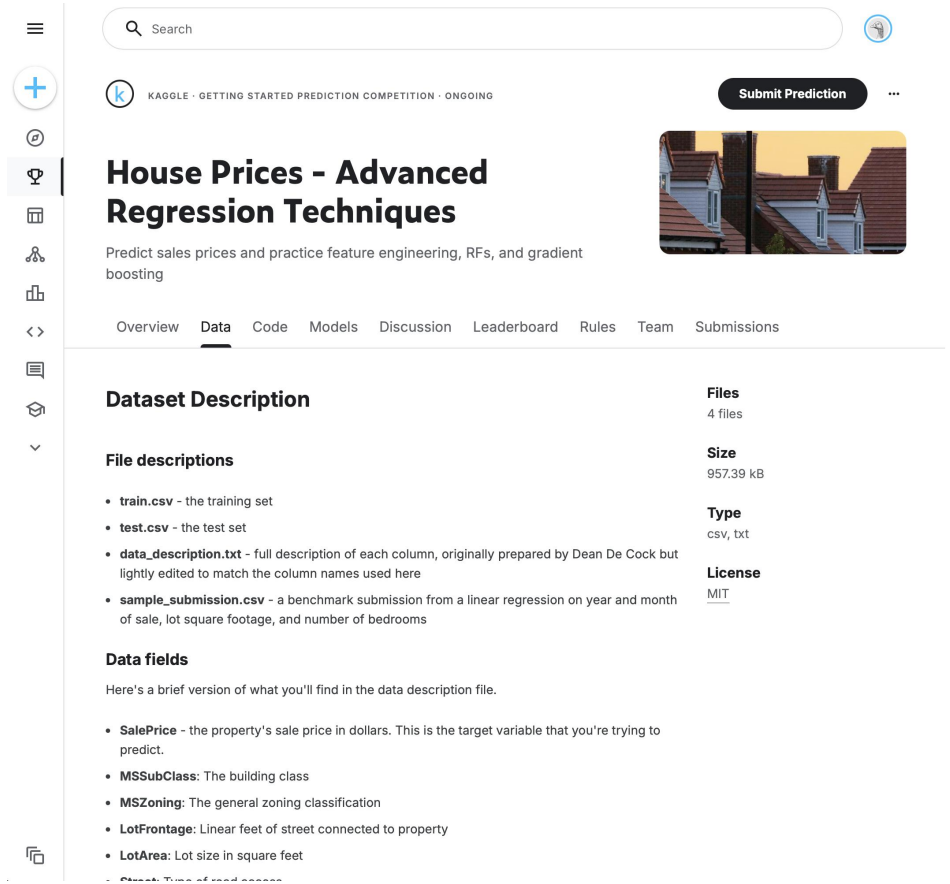
Scikit-learn 提供了从数据预处理、特征选择到分类、回归、聚类、降维等一整套机器学习工具，涵盖模型训练、评估与调优全流程。

目录章节

CONTENTS

房价预测概述

- 任务目标：**给定各种住房的特征，预测其房价的多少**。这里直接采用Kaggle项目：https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data



- 该数据集包含了房屋的结构特征、地理信息及装修情况等 79 个变量，是一个综合性很强的回归问题。
- 通过对这些特征的建模与分析，我们不仅能够预测房价，还能探索影响房价的关键因素。

房价预测步骤：导入数据

➢ 首先需要导入房价数据集，并加载其中的特征与目标值，为后续建模做准备。

```python
import pandas as pd
import numpy as np

# 读取数据
train_data = pd.read_csv('house-prices-advanced-regression-techniques/train.csv')
test_data = pd.read_csv('house-prices-advanced-regression-techniques/test.csv')

# 提取训练集标签和测试集的 Id
y = train_data["SalePrice"]
test_ids = test_data["Id"]

# 删除不需要的列（Id， 标签列）
train_data.drop(["SalePrice", "Id"], axis=1, inplace=True)
test_data.drop("Id", axis=1, inplace=True)

# 对数化处理
y = np.log1p(y)
data = pd.concat([train_data, test_data], axis=0)

# 打印信息，初步查看信息的特征决定下一步需要做什么
data.head()
data.info()
data.describe()
```

```
 5   Alley          198 non-null    object
 6   LotShape       2919 non-null   object
 7   LandContour    2919 non-null   object
 8   Utilities      2917 non-null   object
 9   LotConfig      2919 non-null   object
 10  LandSlope      2919 non-null   object
 11  Neighborhood   2919 non-null   object
 12  Condition1     2919 non-null   object
 13  Condition2     2919 non-null   object
 14  BldgType       2919 non-null   object
 15  HouseStyle     2919 non-null   object
 16  OverallQual    2919 non-null   int64
 17  OverallCond    2919 non-null   int64
 18  YearBuilt      2919 non-null   int64
 19  YearRemodAdd   2919 non-null   int64
...
 77  SaleType       2918 non-null   object
 78  SaleCondition  2919 non-null   object
dtypes: float64(11), int64(25), object(43)
memory usage: 1.8+ MB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea
count	2919.000000	2433.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2896.000000
mean	57.137718	69.305795	10168.114080	6.089072	5.564577	1971.312778	1984.264474	102.201312
std	42.517628	23.344905	7886.996359	1.409947	1.113131	30.291442	20.894344	179.334253
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000
25%	20.000000	59.000000	7478.000000	5.000000	5.000000	1953.500000	1965.000000	0.000000
50%	50.000000	68.000000	9453.000000	6.000000	5.000000	1973.000000	1993.000000	0.000000

● 注：对数化处理的意义：1）缓解数据分布的偏态（让数据更接近正态分布）；2）降低极端值影响（稳定方差）；3）改善模型拟合效果。

这是整个建模流程的起点，确保数据能被正确读取和理解。

房价预测步骤：数据预处理

- 在建模前，需要对原始数据进行清洗与转换，包括缺失值处理、编码类别变量、特征缩放等步骤。

```
# 数据预处理
def fill_missing(col):
    if col.dtype == 'object':
        return col.fillna(col.mode()[0] )
    else:
        return col.fillna(col.mean())

data=data.apply(fill_missing)
data.isnull().sum()
```

```
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
                ..
SaleCondition    0
HouseAge         0
HasPool          0
HasGarage        0
HasBsmt          0
Length: 83, dtype: int64
```

```
from sklearn.preprocessing import OrdinalEncoder

OE=OrdinalEncoder()
obj_data=data.select_dtypes('object')
# 对这些类别型特征进行拟合并转换，将每个类别映射为一个整数（从0开始）
obj_data=OE.fit_transform(obj_data)
data[data.select_dtypes('object').columns]=obj_data
data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape
0	60	3.0	65.0	8450	1.0	0.0	3.0
1	20	3.0	80.0	9600	1.0	0.0	3.0
2	60	3.0	68.0	11250	1.0	0.0	0.0
3	70	3.0	60.0	9550	1.0	0.0	0.0
4	60	3.0	84.0	14260	1.0	0.0	0.0

5 rows × 83 columns

数据预处理能提高模型的稳定性和预测精度，是建模前不可或缺的一环。

房价预测步骤：划分数据集

- 将数据集划分为训练集与测试集，确保模型能够在未见过的数据上验证效果。

```python
# 划分数据集
from sklearn.model_selection import train_test_split

X = data.iloc[:len(y), :]
X_test = data.iloc[len(y):, :]

x_train, x_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=42)

# 查看划分后的数据集形状
print(f"训练集形状: x_train: {x_train.shape}, y_train: {y_train.shape}")
print(f"验证集形状: x_valid: {x_valid.shape}, y_valid: {y_valid.shape}")
print(f"测试集形状: X_test: {X_test.shape}")
```

```
训练集形状: x_train: (1168, 83), y_train: (1168,)
验证集形状: x_valid: (292, 83), y_valid: (292,)
测试集形状: X_test: (1459, 83)
```

通过合理的划分，我们可以评估模型的泛化能力，避免过拟合。

▶ 房价预测步骤：模型选择及训练

➢ 房价预测是一个回归问题，常见模型有：

- 线性回归（基准模型，简单可解释）
- Ridge/Lasso（带正则化，防止过拟合）
- 随机森林 / Gradient Boosting（XGBoost / LightGBM / CatBoost）（效果好，适合非线性关系）
- 神经网络（可以尝试，但需要更多调参）

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# 实例化线性回归模型
LR=LinearRegression()
# 训练模型
LR.fit(x_train,y_train)
# 预测
y_pred_lr=LR.predict(x_valid)
# 计算均方根误差（RMSE）
rmse_lr=np.sqrt(mean_squared_error(y_valid,y_pred_lr))
print(f"线性回归模型的RMSE：{rmse_lr}")
```

```
线性回归模型的RMSE：0.16575068239246504
```

◆ 根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

▶ 房价预测步骤：模型选择及训练

➢ 房价预测是一个回归问题，常见模型有：

- 线性回归（基准模型，简单可解释）
- Ridge/Lasso（带正则化，防止过拟合）
- 随机森林 / Gradient Boosting（XGBoost / LightGBM / CatBoost）（效果好，适合非线性关系）
- 神经网络（可以尝试，但需要更多调参）

```python
from sklearn.linear_model import Ridge
# 实例化岭回归模型
ridge = Ridge(alpha=1.0)  # alpha是正则化参数
# 训练模型
ridge.fit(x_train, y_train)
# 预测
y_pred_ridge = ridge.predict(x_valid)
# 计算均方根误差（RMSE）
rmse_ridge = np.sqrt(mean_squared_error(y_valid, y_pred_ridge))
print(f"岭回归模型的RMSE: {rmse_ridge}")
```

```
岭回归模型的RMSE: 0.15845075242802503
```

◆ 根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

▶ 房价预测步骤：模型选择及训练

➢ 房价预测是一个回归问题，常见模型有：

- 线性回归（基准模型，简单可解释）
- Ridge/Lasso（带正则化，防止过拟合）
- 随机森林 / Gradient Boosting（XGBoost / LightGBM / CatBoost）（效果好，适合非线性关系）
- 神经网络（可以尝试，但需要更多调参）

```python
from sklearn.ensemble import RandomForestRegressor

# 实例化随机森林回归模型
RF = RandomForestRegressor(max_depth=100)
# 训练模型
RF.fit(x_train,y_train)
# 预测
y_pred_rf=RF.predict(x_valid)
# 计算均方根误差（RMSE）
rmse_rf=np.sqrt(mean_squared_error(y_valid,y_pred_rf))
print(f"随机森林回归模型的RMSE: {rmse_rf}")
```

```
随机森林回归模型的RMSE: 0.14710417585780383
```

◆ 根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

▶ 房价预测步骤：模型选择及训练

➢ 房价预测是一个回归问题，常见模型有：

- 线性回归（基准模型，简单可解释）
- Ridge/Lasso（带正则化，防止过拟合）
- 随机森林 / Gradient Boosting（XGBoost / LightGBM / CatBoost）（效果好，适合非线性关系）
- 神经网络（可以尝试，但需要更多调参）

```python
from xgboost import XGBRegressor

# 实例化XGBoost回归模型
xgb = XGBRegressor(
    n_estimators=1000, # n_estimators: 树的数量
    learning_rate=0.01, # learning_rate: 学习率
    max_depth=5, # max_depth: 树的最大深度
    subsample=0.7, # subsample: 用于训练的样本比例
    colsample_bytree=0.5, # colsample_bytree: 用于训练的特征比例
    random_state=42 # random_state: 随机种子
)

# 训练模型
xgb.fit(x_train,y_train)
# 预测
y_pred_xgb=xgb.predict(x_valid)
# 计算均方根误差 (RMSE)
rmse_xgb=np.sqrt(mean_squared_error(y_valid,y_pred_xgb))
print(f"XGBoost回归模型的RMSE: {rmse_xgb}")
```

```
XGBoost回归模型的RMSE: 0.13271680733542823
```

◆ 根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

根据任务需求选择合适的回归模型，如线性回归、随机森林或梯度提升等。

▶ 房价预测步骤：保存模型

➢ 在模型训练完成后，可以将其保存为文件，方便后续快速加载与复用。这样无需重新训练，就能直接在新数据上进行预测，提高效率。

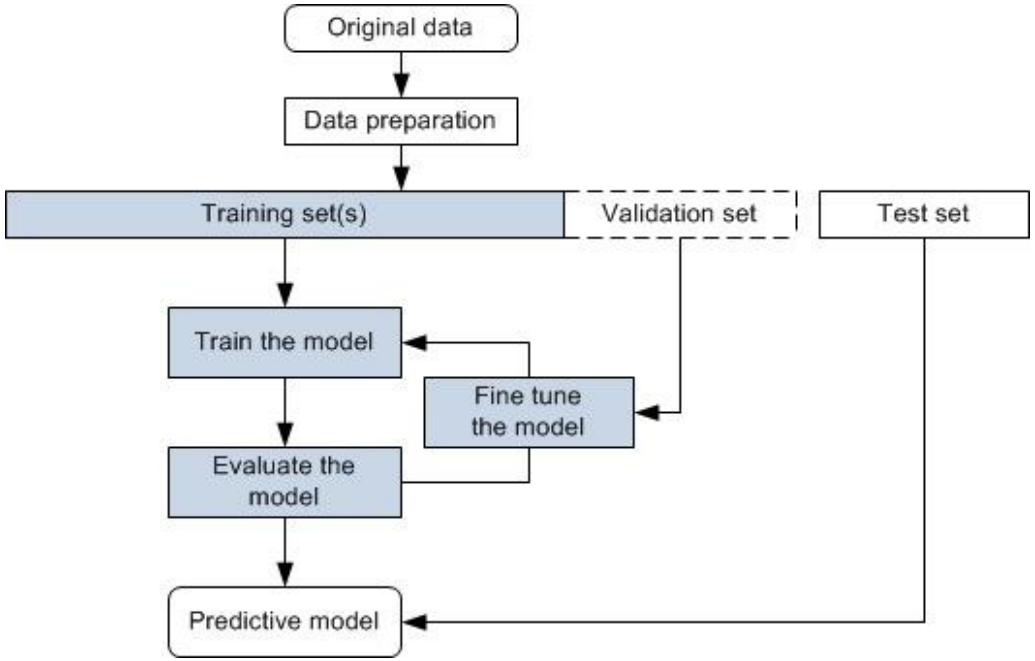
```
# 使用joblib保存模型
import joblib
# 保存之前训练的各个模型
joblib.dump(LR, 'house-prices-advanced-regression-techniques/lr_model.pkl')
joblib.dump(ridge, 'house-prices-advanced-regression-techniques/ridge_model.pkl')
joblib.dump(RF, 'house-prices-advanced-regression-techniques/rf_model.pkl')
joblib.dump(xgb, 'house-prices-advanced-regression-techniques/xgb_model.pkl')

# 使用pickle保存模型
import pickle
# 保存之前训练的各个模型
with open('house-prices-advanced-regression-techniques/lr_model.pkl', 'wb') as f:
    pickle.dump(LR, f)
with open('house-prices-advanced-regression-techniques/ridge_model.pkl', 'wb') as f:
    pickle.dump(ridge, f)
with open('house-prices-advanced-regression-techniques/rf_model.pkl', 'wb') as f:
    pickle.dump(RF, f)
with open('house-prices-advanced-regression-techniques/xgb_model.pkl', 'wb') as f:
    pickle.dump(xgb, f)
```

保存模型，把训练好的智慧装进文件，随时调用，省时高效，方便他人复现。

▶ 小结

- 房价预测的完整流程：
  - 导入数据：获取并加载房价及相关特征数据。
  - 数据预处理：清洗数据，处理缺失值和类别编码。
  - 划分数据集：分为训练集和测试集，评估模型泛化能力。
  - 模型选择与训练：根据任务选择合适模型，学习特征与房价关系。
  - 模型评估与优化：用测试集验证性能，调参提升效果。
  - 保存模型：将训练好的模型存储，方便复用和部署。



掌握这六步，便能系统地完成一个房价预测项目，从数据的获取与清洗，到模型的训练与调优，再到最终模型的保存和部署，实现从原始数据到精准预测的完整闭环。

目录章节

CONTENTS

目录章节

CONTENTS

总结

- **什么是sklearn？**
  - ✓ 定义：Python机器学习库，封装了主流算法与工具。
  - ✓ 特点：统一API、易用、高效、与Numpy/Pandas无缝衔接。
  - ✓ 总之，用最少的代码，完成数据到模型的全流程。
- **sklearn的核心工作流：**
  - ✓ 加载数据与预处理 → 选择模型 → 训练与预测 → 评估 → 提升模型性能 → 保存与加载预训练模型
- **常用模块与算法：**
  - ✓ sklearn通过 preprocessing、feature_selection、linear_model、svm、ensemble、cluster、decomposition、model_selection 和 metrics 等模块，覆盖了从数据处理到模型训练与评估的全流程。

sklearn集成多种机器学习模块和算法，涵盖数据预处理、特征选择、建模、评估与调优，助力快速高效完成各种机器学习任务。

感谢聆听

Personal Website: https://www.miaopeng.info/

Email: miaopeng@stu.scu.edu.cn

Github: https://github.com/MMeowhite

Youtube: https://www.youtube.com/@pengmiao-bmm