sklearn模块

数据预处理 preprocessing

- MinMaxScaler 归一化（缩放到0-1）
- StandardScaler 标准化（均值0方差1）
- Binarizer 二值化（指定阈值，全部分为0和1）
- OneHotEncoder 热独编码，去掉数值大小影响
- OrdinalEncoder 数值编码，一个数字对应一个类别，数字有大小之分
- PolynomialFeatures 多项式特征

基类和实用函数 base

>>> BaseEstimator
该方法适用于简单估计器以及嵌套对象（例如Pipeline）
比如在自定义分类器中作为闭包，用于调整参数等功能，暂不深入讨论

>>> TransformerMixin
继承fit_transform()方法

>>> clone
可以克隆一个模型实例

数据集 datasets

取数据函数
>>> fetch_openml('数据集名称',版本,,,)

加载鸢尾花数据
>>> load_iris()

生成卫星数据
>>> make_moons(n_numbers, shuffle, noise, random_state)
生成数量，是否打乱，噪声大小，随机种子

生成聚类点
>>> make_blobs(n_samples=1000, centers=((4, -4), (0, 0)))
1. n_samples 样本数量
2. centers 中心点
3. cluster_std 类别标准差
4. random_state 随机数

MNIST手写数字数据集
load_digits()

支持向量机 svm

>>> SVC(C=1)
支持向量机分类
C越高泛化能力越低，容易过拟合

>>> decision_function(数据)
对数据得出评估分数

>>> .classes_
是属性，按大小储存分类类别

>>> SVR()
支持向量机回归
1. kernel 核技巧，可以使用linear（建议用 LinearSVR），poly，rbf，sigmoid
2. gamma 与拟合程度有关，过拟合时降低，只对'rbf','poly','sigmoid'起作用
3. C 与容忍度负相关，过拟合时降低
4. degree 核技巧为poly时指定多项式阶数
5. coef0 核函数的常数项，只对'poly','sigmoid'有用
6. probability 是否采用概率输出

>>> LinearSVC(C=1, loss='hinge')
线性分类支持向量机
C越高泛化能力越低，容易过拟合

>>> LinearSVR()
线性SVM回归

>>> OneClassSVM(kernel = 'rbf', gamma = 0.001, nu = 0.03)
单类支持向量机（异常值检测）
高斯RBF核技巧，nu为异常值百分比

多分类 muticlass

可以将其他分类器传入该分类器中，实现其他分类器指定OvO或者OvR策略

>>> OneVsOneClassifier()
一对一分类器

>>> OneVsRestClassifier()
一对余分类器

最近邻搜索 neighbors

.kneighbors
返回指定实例在训练集中最近邻居的距离和索引

>>> KNeighborsClassifier()
K近邻分类器

分解器 decomposition

主成分分析
>>> PCA(n_components = 2, svd_solver='randomize')
1. n_components 目标维数
2. svd_solver设置为randomize为随机搜索，设置为full为完全SVD搜索

增量PCA
>>> IncrementalPCA
参数与PCA类似，用于大型和在线数据集

内核
>>> PCAkernelPCA()
1. n_components 目标维数
2. kernel 核技巧 如高斯RBF（rbf, linear, sigmoid等，同SVM）
3. gamma 与拟合程度有关，过拟合时降低

流形模块 manifold

LLE局部线性嵌入
>>> LocallyLinearEmbedding()
1. n_components 目标维度
2. n_neighbors 近邻数量

聚类算法（无监督） cluster

>>> KMeans()
1. n_clusters 指定聚类数
2. init 指定初始中心点
3. n_init 中心点随机初始化次数
实例属性：
1. labels_ 标签
2. cluster_centers_ 中心点坐标

小批量KMeans
>>> MiniBatchKMeans()
参数与KMeans类似

混合算法（无监督） mixture

高斯混合模型
>>> GaussianMixture()
1. n_components 核心实例数（聚类数）
2. n_init 中心点随机初始化次数，默认为1，建议调高避免次优解

流水线 pipeline

>>> Pipepline([('名称', 分类/转换器1),('名称', 分类/转换器2)])
列表嵌套元组，元组包含分类/转换器名称和转换器本身

>>> make_pipeline

线性模型 linear_model

线性回归
>>> LinearRegression()

梯度下降回归
>>> SGDRegressor(max_iter=1000, tol=1e-3, penalty=None, eta0=0.1)
迭代次数，tol容忍度，正则类型（ 'l2' 等同岭回归， 'l1' 为Lasso回归），eta0学习率

梯度下降分类
>>> SGDClassifier(random_state=42)
设定随机数种子才能复现，否则训练是随机的，结果每次都不一样

岭回归
>>> Ridge(alpha=1 , solver='cholesky', random_state=42)

弹性网络
>>> ElasticNet()

逻辑回归
>>> LogisticRegression()
默认使用Logic回归，参数设定multi_class='multinomial'时使用Softmax回归
其他设置：multi_class="ovr", solver="lbfgs"等

推算处理缺失值 impute

填补缺失值
>>> SimpleImputer(mising_values=nan，strategy='mean/median/most_frequent(众数)')
一般可以直接写 imputer = SimpleImputer(strategy='median') 创建一个实例（估算器）

训练数据
>>> imputer.fit(面板数据)
用该实例训练数据（算出中位数）

填充数据
>>> X = imputer.transform(面板数据)
X是np数组，还需转化为DF数据

度量模块 metrics

均方误差
>>> mean_squared_error(真实值, 预测值)
计算真实值和预测值之间的方差

R^2
>>> r2_score(真实值, 预测值)

精确性分数
>>> accuracy_score(真实标签，预测标签)

混淆矩阵
>>> confusion_matrix(真实标签，预测标签)
对于多分类问题，一般没有0值，完美的二分类器则仅对角线有值

精度，查准率
>>> precision_score(真实标签，预测标签)

召回率，查全率
>>> recall_score(真实标签，预测标签)

F_1分数
>>> f1_score(真实标签，预测标签)

PR（精度/召回率）曲线
>>> precision_recall_curve(真实标签,预测分数)
返回精度、召回率、阈值，用于作PR图

ROC曲线
>>> roc_curve(真实标签，预测分数)
返回FPR（假正率），TPR（真正率），thresholds

AUC分数（即ROC曲线下方面积）
>>> roc_auc_score(真实标签，预测分数)

注意，这里的参数是（真实标签，预测分数），而非（真实标签，预测标签）

轮廓分数
>>> silhouette_score(实例, 标签)
用于KMeans中心点评判

复合估计器 compose

列转换器
ColumnTransformer

模型选择器 model_selection

随机抽样
>>> train_test_split(数据集，测试集比例，随机数种子，stratify=)
stratify=某一列可以使抽样保持分布

分层随机抽样（按照分组数生成若干不重复的【训练集/测试集】数据对）
>>> StratifiedShuffleSplit(分组数，测试集比例，随机数种子)
生成的是“切割规则”而非数据本身，与原数据集对应，利用for循环进行赋值

.split(x,y)将x按照y中数据类别比例进行等比划分

分层随机抽样误差小
随机抽样通常有偏

交叉分数（K折验证）
>>> cross_val_score(实例, x, y, scoring='neg_mean_squared_error', cv=10)
scoring指定评分规则，这里为均方误差，也可以是accuracy（严格匹配，用于二分类问题）等，cv是折数

交叉预测（K折预测）
>>> cross_val_predict(模型，训练集，训练集标签，cv=折数, method='方法')
返回每个折叠的预测，用于得出混淆矩阵，method参数可选，如'decision_function'就返回预测分数，'predict_proba' 返回概率，否则返回预测值

网格搜索
>>> GridSearchCV(模型，超参数字典，CV=折数，scoring='neg_mean_squared_'， return_train_score=True)
其中超参数字典表示方法暂不做讨论
该方法将对指定的超参数进行遍历验证搜索

随机搜索
>>> RandomizedSearchCV() 【暂不详细讨论】
对超参数进行随机搜索，而非指定数值搜索

决策树 tree

决策树回归
>>> DecisionTreeRegressor()

决策树分类
>>> DecisionTreeClassifier()
1. max_depth 控制深度
2. min_samples_split 分裂前节点必须有的最小样本数
3. min_samples_leaf 叶节点必须有的最小样本数量
4. min_weight_fraction_leaf 加权实例总数站别
5. max_leaf_modes 最大叶节点数量
6. max_features 每个节点评估的最大特征数量

决策树可视化
>>> export_graphviz()

集成算法 ensemble

随机森林回归
>>> RandomForestRegressor()

极端随机树回归
>>> ExtraTreesRegressior()

随机森林分类
>>> RandomForestClassifier()
1. n_estimators 决策树数量
2. max_leaf_nodes 最大叶节点
3. n_jobs 指定CPU个数

极端随机树分类器
>>> ExtraTreesClassifier()

多算法投票分类器
>>> VotingClassifier()
1. estimators=[('name_1', clf_1), ('name_2', clf_2), ... ,]
2.voting='hard/soft' 硬投票或软投票（概率）

单算法多采样分类器
>>> BaggingClassifier()
1. 分类器
2. n_estimators 指定分类器个数
3. max_samples 每次从样本抽取多少个实例
4. bootstrap ， True为有放回， False不放回（也就是pasting）
5. n_jobs , 指定可使用的CPU个数， -1为全部可用

梯度提升回归
>>> GradientBoostingRegressor()
参数与决策树基本一致
学习率越低，决策树越多，泛化效果较好

梯度提升分类
>>> GradientBoostingClassifier()