注意：DBSCAN无predict方法，只有fit_predict方法

In [ ]:

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN().fit(iris.data)
dbscan.labels_
```

In [ ]:

```
dbscan.components_
```

In [ ]:

```
dbscan.fit_predict(iris.data)
```

In [ ]:

```
# 增大距离参数
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps = 1).fit(iris.data)
dbscan.labels_
```

6.5 实战练习

将iris数据集的案例顺序彻底随机化，然后重新使用BIRCH方法进行聚类。列出随机化以后聚类结果和真实类别间的交叉表，并且和按照原顺序得到的BIRCH聚类结果和真实类别的交叉表作比较，思考案例顺序随机化处理在BIRCH方法中的重要性。

> 提示：交叉表描述可使用Pandas中的功能完成，也可以使用7.1.1中将要介绍的混淆矩阵完成。案例随机化可以使用Pandas中的功能完成，如果对Pandas不熟悉，也可以参考8.2.1节中的程序实现方式。

将iris数据集的案例顺序彻底随机化，使用K-Means方法进行聚类操作，并比较随机化前后的聚类结果，思考为什么会和BIRCH方法存在这种差异。

7 评估模型效果

评分方法缩写	Sklearn中的具体函数
Classification（分类）	
'accuracy'	metrics.accuracy_score
'average_precision'	metrics.average_precision_score
'f1'	metrics.f1_score
'precision' etc.	metrics.precision_score
'recall' etc.	metrics.recall_score
'roc_auc'	metrics.roc_auc_score
Clustering（聚类）	
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score
'adjusted_rand_score'	metrics.adjusted_rand_score
'completeness_score'	metrics.completeness_score
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score
'homogeneity_score'	metrics.homogeneity_score
'mutual_info_score'	metrics.mutual_info_score
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score
'v_measure_score'	metrics.v_measure_score
Regression（回归）	
'explained_variance'	metrics.explained_variance_score
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
'r2'	metrics.r2_score

7.1 类别预测模型

7.1.1 分类结果的呈现：混淆矩阵

可在该矩阵的基础上进一步计算分类别的各种指标

sklearn.metrics.confusion_matrix(y_true, y_pred, labels = None, sample_weight = None)

输出预测类别和实际类别的交叉矩阵（混淆矩阵）
函数返回：array, shape = [n_classes, n_classes]

In [ ]:

```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier().fit(iris.data, iris.target)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(iris.target, mlp.predict(iris.data))
cm
```

In [ ]:

```
# 自定义类别顺序
confusion_matrix(iris.target, mlp.predict(iris.data), labels = [2, 1, 0])
```

In [ ]:

```
# 可以考虑用热图的形式加以呈现
sns.heatmap(cm, cmap = sns.color_palette("Blues"), annot = True)
```

7.1.2 分类结果的评价：评分

sklearn中的常见评分函数：

```
precision_score: 计算准确率
recall_score: 计算召回率
average_precision_score: 计算预测值的AP（平均准确率）
    实际对应了precision-recall曲线下的面积
f1_score: 计算F1值，也被称为平衡F-score或F-measure
fbeta_score: 计算F-beta score，即准确率和召回率的加权协调均数
precision_recall_curve: 计算不同概率阀值的precision-recall对
precision_recall_fscore_support: 为每个类计算precision, recall, F-measure和 su
pport
```

7.1.2.1 准确率与召回率

sklearn.metrics.precision_recall_curve(

```
y_true : array, shape = [n_samples]，因变量的真值
probas_pred : array, shape = [n_samples]，模型给出的每个案例的估计概率
pos_label = None : 阳性类别的标签
sample_weight = None : 案例权重
```

)返回：

```
precision : array, shape = [n_thresholds + 1]，准确率
    第i个元素为score >= thresholds[i]时的准确率，最后一个元素为1
recall : array, shape = [n_thresholds + 1]，召回率
    第i个元素为score >= thresholds[i]时的召回率，最后一个元素为0
thresholds : array, shape = [n_thresholds <= 不重复的预测概率列表长度]
    用于计算上述两个率的不重复的概率值列表，为升序排列
```

注意：该函数只能用于二值预测模型，多类预测问题需要转化为二值问题进行评估。

In [ ]:

```
import numpy as np
from sklearn.metrics import precision_recall_curve

y_true = np.array([0, 0, 0, 0, 0,
                   1, 1, 1, 1, 1])
y_scores = np.array([0.1, 0.2, 0.25, 0.4, 0.6,
                     0.2, 0.45, 0.6, 0.75, 0.8])
precision, recall, thresholds = precision_recall_curve(
     y_true, y_scores)
```

In [ ]:

```
len(precision), len(recall), len(thresholds)
```

In [ ]:

```
thresholds
```

In [ ]:

```
precision
```

In [ ]:

```
plt.scatter(x = thresholds, y = precision[:-1])
```

In [ ]:

```
plt.scatter(x = thresholds, y = recall[:-1])
```

In [ ]:

```
plt.scatter(x = recall, y = precision)
```

In [ ]:

```
sns.lineplot(recall, precision)
```

In [ ]:

```
# 多值预测问题转化为二值问题进行评估
from sklearn.preprocessing import binarize

y = binarize(iris.target.reshape(-1, 1))
```

In [ ]:

```
pd.DataFrame(mlp.predict_proba(iris.data))[0].head()
```

In [ ]:

```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier().fit(iris.data, y)
```

In [ ]:

```
from sklearn.metrics import precision_recall_curve

precision, recall, thresholds = precision_recall_curve(y,
    pd.DataFrame(mlp.predict_proba(iris.data))[0], pos_label = 0)
```

In [ ]:

```
plt.scatter(x = thresholds, y = recall[:-1])
```

In [ ]:

```
plt.scatter(x = recall, y = precision)
```

7.1.2.2 给出预测值的平均准确率

AP: average precision，所有实际阳性案例的准确率平均值。

sklearn.metrics.average_precision_score(

```
    y_true : array, shape = [n_samples] or [n_samples, n_classes]
    y_score :  array, shape = [n_samples] or [n_samples, n_classes]
    average = 'macro' : string, 具体的汇总平均方法
        [None, 'micro', 'macro' (default), 'samples', 'weighted']
        None : 分各类别报告指标，不进行平均
        'binary' : 只报告pos_label参数指定类别的结果，仅对两分类有效
        'micro' : 直接对整个样本计算各个指标，显然小类在计算中会相对更重要
        'macro' : 直接基于整个样本的tp，tf等计算评分指标，不考虑样本比例不平衡
        'weighted' : 分类别计算各个指标，然后按照各类别的真实案例数作加权平均
        'samples' : 适用于多标签问题，直接计算每个样本的指标，然后再作加权平均
    sample_weight = None
```

)

In [ ]:

```
from sklearn.metrics import average_precision_score

average_precision = average_precision_score(y_true, y_scores)
average_precision
```

7.1.3 评分结果的汇总输出

F1 score

F1 score = 2 *(命中率* * 召回率) / (命中率 + 召回率)

sklearn.metrics.f1_score(

```
    y_true : 各案例的真实类别
    y_pred : 各案例的预测类别
    pos_label = 1 : str or int, 两分类时希望报告的类别，多分类时该参数无效
    average = 'binary' : 指定多分类时的指标平均方法
        [None, 'binary', 'micro', 'macro', 'samples', 'weighted']
        None : 分各类别报告指标，不进行平均
        'binary' : 只报告pos_label参数指定类别的结果，仅对两分类有效
        'micro' : 直接对整个样本计算各个指标，显然小类在计算中会相对更重要
        'macro' : 直接基于整个样本的tp，tf等计算评分指标，不考虑样本比例不平衡
        'weighted' : 分类别计算各个指标，然后按照各类别的真实案例数作加权平均
        'samples' : 适用于多标签问题，直接计算每个样本的指标，然后再作加权平均
    labels = None, sample_weight = None
```

)返回：阳性类别的f1_score（对于两分类），所有类别加权平均后的f1_score（多分类）

In [ ]:

```
from sklearn.metrics import f1_score

mlp = MLPClassifier().fit(iris.data, iris.target)
f1_score(iris.target, mlp.predict(iris.data), average = 'micro')
```

分类结果的汇总报告

sklearn.metrics.classification_report(

```
    y_true, y_pred, labels = None, target_names = None
    sample_weight = None, digits = 2
```

)返回：每个类别的precision, recall, F1 score

In [ ]:

```
from sklearn.metrics import classification_report

# 使用print函数让结果自动格式化
print(classification_report(iris.target, mlp.predict(iris.data)))
```

7.1.4 ROC曲线分值

计算各拆分点的相应指标

sklearn.metrics.roc_curve(

```
    y_true, y_score, pos_label = None, sample_weight = None
    drop_intermediate = True : 是否丢弃一些偏离ROC曲线的次优阈值
        该参数对于创建更简明的ROC曲线非常有用
```

)返回:

```
    fpr : array, shape = [>2], 假阳性率
    tpr : array, shape = [>2], 真阳性率
    thresholds : array, shape = [n_thresholds]
        用于计算上述两个率的不重复的概率值列表，为降序排列
```

In [ ]:

```
from sklearn.metrics import roc_curve

roc_curve(y_true, y_scores)
```

In [ ]:

```
fpr, tpr, thresholds = roc_curve(y_true, y_scores)
```

In [ ]:

```
sns.lineplot(fpr, tpr, ci = None)
plt.ylim(0, 1.01)
plt.plot([0, 1], [0, 1])
```

计算ROC曲线下面积

sklearn.metrics.roc_auc_score(

```
    y_true, y_score, average = 'macro', sample_weight = None
```

)

In [ ]:

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_true, y_scores)
```

7.2 回归类模型

sklearn中常用的连续结果评分指标:

```
    mean_squared_error : 即残差平方的均值
    mean_squared_log_error : 原指标自然对数的残差平方均值
        适用于对数正态分布的指标
    mean_absolute_error : 即残差绝对值的均值

    explained_variance_score : 模型所解释的方差占总方差的比例
    r2_score : 决定系数
```

sklearn.metrics.r2_score(

```
    y_true, y_pred, sample_weight = None

    multioutput = 'uniform_average' : 多个目标变量时指标的平均方式
        'raw_values' : 直接返回原始的评分
        'uniform_average' : 返回原始评分的直接平均值
        'variance_weighted' : 返回评分按照方差加权的平均值
```

)

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LinearRegression()

reg.fit(boston.data, boston.target)

y_pred = reg.predict(boston.data)
```

In [ ]:

```
# 无偏估计时两个指标的结果完全相同
from sklearn.metrics import explained_variance_score
from sklearn.metrics import r2_score

print(explained_variance_score(boston.target, y_pred))
print(r2_score(boston.target, y_pred))
```

7.3 聚类模型

7.3.1 存在ground truth class

sklearn.metrics.adjusted_rand_score(labels_true, labels_pred)

sklearn.metrics.mutual_info_score(labels_true, labels_pred, contingency = None)

In [ ]:

```
from sklearn.metrics import adjusted_rand_score

adjusted_rand_score(iris.target, kmeans.labels_)
```

In [ ]:

```
from sklearn.metrics import mutual_info_score

mutual_info_score(iris.target, kmeans.labels_)
```

In [ ]:

```
from sklearn.metrics import mutual_info_score

mutual_info_score(kmeans.labels_, iris.target)
```

7.3.2 无ground truth class

sklearn.metrics.silhouette_score(

```
    X, labels, metric = 'euclidean', sample_size = None
    random_state = None, **kwds
```

)

In [ ]:

```
from sklearn.metrics import silhouette_score

silhouette_score(iris.data, kmeans.labels_)
```

7.4 与随机预测结果相比较（虚拟估计）

7.4.1 虚拟分类

class sklearn.dummy.DummyClassifier(

```
    strategy = 'stratified' : str, 预测值的生成策略
        stratified : 通过考虑训练集中各类的分布来生成随机预测
        most_frequent : 总是预测训练集中最常见的类标签
        prior : 总是预测训练集中先验概率最大的一类，predict_proba返回预测概率
        uniform : 随机产生预测类别
        constant : 总是预测用户提供的常数类标签，这种方法对评估小类别很有效

    random_state = None,
    constant = None : int/str/形如[n_outputs]的array，用户提供的常数类标签
```

)

DummyClassifier类的属性：

```
    classes_ : array or list of array of shape = [n_classes]
    n_classes_ : array or list of array of shape = [n_classes]
    class_prior_ : array or list of array of shape = [n_classes]
    n_outputs_ : int,
    outputs_2d_ : bool, 输出是否为2d数组
    sparse_output_ : bool
```

In [ ]:

```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier().fit(iris.data, iris.target)
mlp.score(iris.data, iris.target)
```

In [ ]:

```
from sklearn.dummy import DummyClassifier

clf = DummyClassifier(random_state = 0)
clf.fit(iris.data, iris.target)
clf.score(iris.data, iris.target)
```

7.4.2 虚拟回归

class sklearn.dummy.DummyRegressor(

```
    strategy = 'mean' : 预测值的生成策略
        mean : 总是预测为训练集目标变量的平均值.
        median : 总是预测为训练集目标变量的中位数.
        quantile : 总是预测为用户提供的训练集目标变量的某一分位数
        constant : 总是预测为由用户提供的常数值

    constant = None : int / float / 形如[n_outputs]的array
    quantile = None : float in [0.0, 1.0]，用户指定的百分位数
```

)

DummyRegressor类的属性：

```
    constant_ : float or array of shape [n_outputs]
    n_outputs_ : int,
    outputs_2d_ : bool,
```

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LinearRegression().fit(boston.data, boston.target)
reg.score(boston.data, boston.target)
```

In [ ]:

```
from sklearn.dummy import DummyRegressor

reg = DummyRegressor().fit(boston.data, boston.target)
reg.score(boston.data, boston.target)
```

7.5 实战练习

以均数为标准将boston数据的因变量拆分为二分类，分别拟合不同的分类预测模型，并进行完整的模型效果评价。

思考在身边可能遇到的聚类分析案例中，有无可能出现ground truth class的情形。

8 数据的拆分

使用建模数据的结果进行模型效果评价，很难避免过拟合发生。

8.1 二分法拆分

sklearn.model_selection.train_test_split(

```
    *arrays : 等长度的需要拆分的数据对象
        格式可以是lists, numpy arrays, scipy稀疏矩阵或者pandas数据框
        显然，对于有监督类模型，X和y需要按相同标准同时进行拆分
    test_size = 0.25 : float, int, None, 用于验证模型的样本比例，范围在0~1
        为None时所有样本都将用于训练
    train_size = None : float, int, or None, 用于训练模型的样本比例，0~1
        为None时自动基于test_size计算
    random_state = None
    shuffle = True : 是否在拆分前对样本做随机排列
    stratify = None : array-like or None，是否按指定类别标签对数据做分层拆分
```

)返回 : 对输入对象进行拆分后的list, length = 2 * len(arrays)