2.2 基于sklearn构建完整的分类项目

2.2.1 数据准备

使用经典的分类项目数据集：鸢尾花数据集。

```
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
feature = iris.feature_names
data = pd.DataFrame(X,columns=feature)
data['target'] = y
print(data.head())
```

2.2.2 评价指标

2.2.2.1 二分类问题

- 混淆矩阵
  - 真阳性TP：预测值为正，实际值为正；
  - 真阴性TN：预测值为负，实际值为负；
  - 假阳性FP：预测值为正，实际值为负；
  - 假阴性FN：预测值为负，实际值为正；
- 准确率：$ACC = \frac{TP+TN}{FP+FN+TP+TN}$
- 精确率：$PRE = \frac{TP}{TP+FP}$
- 召回率：$REC = \frac{TP}{TP+FN}$
- F1值：$F1 = 2 * \frac{PRE*REC}{PRE+REC}$
- ROC曲线：以FP为横轴，TP为纵轴的曲线
- AUC：ROC曲线下方的面积

2.2.2.2 多分类问题

假设对于一个单标签多分类问题，有*c*个类，分别记为*1*、*2*、*…*、*c*。

- TP_j指分类*j*的真正率（True Positive）；
- FP_j指分类*j*的假正率（False Positive);
- TN_j指分类*j*的真负率（True Negative);
- FN_j指分类*j*的假负率（False Negative);

分类*j*的准确率、精确率、召回率、F1值计算方法是：

- 准确率：$ACC_j = \frac{TP_j+TN_j}{FP_j+FN_j+TP_j+TN_j}$
- 精确率：$PRE_j = \frac{TP_j}{TP_j+FP_j}$
- 召回率：$REC_j = \frac{TP_j}{TP_j+FN_j}$

- F1值：$F1_j = 2 * \frac{PRE_j * REC_j}{PRE_j + REC_j}$

基于Micro方法的三种评估指标不需要区分类别，直接使用总体样本计算。这种计算方式使得评价指标受样本量更大的类别影响更大。其公式为：

- Micro精确率：$PRE_{micro} = \frac{\sum_{j=1}^{c} TP_j}{\sum_{j=1}^{c} TP_j + \sum_{j=1}^{c} FP_j}$
- Micro召回率：$REC_{micro} = \frac{\sum_{j=1}^{c} TP_j}{\sum_{j=1}^{c} TP_j + \sum_{j=1}^{c} FN_j}$
- MicroF1值：$F1_{micro} = 2 * \frac{PRE_{micro} * REC_{micro}}{PRE_{micro} + REC_{micro}}$

> 对于多分类问题，准确率、Micro精确率、召回率、F1值是相等的。

基于Macro方法的三种评估指标均等地看待所有类别的影响，对于每个类别的评价指标求平均，得到总体的评价指标，因此结果容易受到稀有类别的影响。其公式为：

- 精确率：$PRE_{macro} = \frac{1}{c} \sum_{j=1}^{c} \frac{TP_j}{TP_j + FP_j}$
- 召回率：$REC_{macro} = \frac{1}{c} \sum_{j=1}^{c} \frac{TP_j}{TP_j + FN_j}$
- F1值：$F1_{macro} = 2 * \frac{PRE_{macro} * REC_{macro}}{PRE_{macro} + REC_{macro}}$

2.2.3 模型训练

2.2.3.1 逻辑回归

使用sigmoid函数作为预测为正类的概率。也即：

$$p_1 = P(y = 1|x) = \frac{1}{1 + e^{-w^T x}}$$

假设标签数据服从伯努利分布，也即：

$$P(y|x) = \begin{cases} p1, & \text{if } y = 1 \\ 1 - p1, & \text{if } y = 0 \end{cases}$$

将两条公式结合起来，

使用**极大似然估计**来对参数w进行估计，其核心思想是给定自变量X和参数w，出现观测值Y的概率最大。

$$\begin{aligned} \max L(w) &= \log P(Y|X; w) \\ &= \log \prod_{i=1}^{n} P(y_i|x_i; w) \\ &= \sum_{i=1}^{n} \log P(y_i|x_i; w) \\ &= \sum_{i=1}^{n} y_i \log p_1 + (1 - y_i) \log(1 - p_1) \end{aligned}$$

使用梯度下降法进行求解。

```
from sklearn.linear_model import LogisticRegression
log_iris = LogisticRegression()
log_iris.fit(X,y)
log_iris.score(X,y)
```

2.2.3.2 决策树

决策树有三种常见的算法：ID3、C4.5、CART。这三种模型的主要区别是分裂节点选择的指标不同。

ID3使用信息增益。

- 熵$H(D)=\sum\limits_{x\in D}-p\log p$：表示随机变量的不确定性。
- 条件熵$H(D|A)$：在一个条件下，随机变量的不确定性。
- 信息增益$g(D,A)=H(D)-H(D|A)$：熵 - 条件熵。表示在一个条件下，信息不确定性减少的程度。

C4.5使用信息增益率。

- 数据集D关于特征A的值的熵$H_A(D)=-\sum\limits_{i=1}^{n}\frac{|D_i|}{|D|}\log\frac{|D_i|}{|D|}$
- 信息增益率$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$

CART使用基尼系数。

- 基尼系数$G(D)=1-\sum\limits_{x\in D}p^2$
- 条件基尼系数$G(D,A)=\sum\limits_{i=1}^{n}\frac{|D_i|}{|D|}G(D_i)$

```
# criterion:{“gini”, “entropy”}, default=”gini”
from sklearn.tree import DecisionTreeClassifier
tree_iris = DecisionTreeClassifier(criterion='gini'
, min_samples_leaf=5)
tree_iris.fit(X,y)
tree_iris.score(X,y)
```

2.2.3.3 SVM

我们的目标是找到一个支撑超平面，这个超平面能够最大程度地将正负两类样本点分开。也就是说，距离超平面最近的两个点（分属于两种类别）的间隔最大。

假设最近点到超平面的距离为δ，也即$w^Tx+b=\delta$，同时缩放w和b，可以将其转化为$w^Tx+b=1$。也就是说，距离超平面最近的两个点满足：

$$
\begin{aligned}
w^Tx_1+b=1\\
w^Tx_2+b=-1
\end{aligned}
$$

稍加转化可以得到，

$$
\begin{aligned}
\left(w^Tx_1+b\right)-\left(w^Tx_2+b\right)&=2\\
w^T\left(x_1-x_2\right)&=2\\
w^T\left(x_1-x_2\right)=\|w\|_2\|x_1-x_2\|_2\cos\theta&=2\\
\|x_1-x_2\|_2\cos\theta&=\frac{2}{\|w\|_2}\\
d_1=d_2=\frac{\|x_1-x_2\|_2\cos\theta}{2}&=\frac{\frac{2}{\|w\|_2}}{2}=\frac{1}{\|w\|_2}\\
d_1+d_2&=\frac{2}{\|w\|_2}
\end{aligned}
$$

假设正类样本真实标签为1，负类样本真实标签为-1（这里为了简化模型，与一般模型的设计不太一样，其他模型常将正类设置为1，负类设置为0）

因此SVM可以被表示为：

$$\min\ L(w,b)=\frac{1}{2}||w||^2$$
$$s.t.\begin{cases} w^Tx_i+b\geq 1, & \text{if } y_i=1 \\ w^Tx_i+b\leq -1, & \text{if } y_i=-1 \end{cases}$$

可以将两个条件合并为 $y_i(w^Tx_i+b)\geq 1$，从而模型转化为：

$$\min\ L(w,b)=\frac{1}{2}||w||^2$$
$$s.t.\ y_i(w^Tx_i+b)\geq 1,\qquad i=1,\ldots,n$$

通过拉格朗日乘子法求解。

加入软约束

如果数据集中存在一些噪声数据，则数据不能被完全可分，此时我们可以对原始损失函数添加一个软约束。模型具体形式为：

$$\min\ L(w,b,\xi)=\frac{1}{2}||w||^2+C\sum_{i=1}^{n}\xi_i$$
$$s.t.\begin{cases} y_i(w^Tx_i+b)\geq 1-\xi_i, & i=1,\ldots,n \\ \xi_i\geq 0, & i=1,\ldots,n \end{cases}$$

非线性支持向量机

对于完全不线性可分的数据集，一种可能的方式是将数据投影到更高维的空间上，此时在低维空间中线性不可分的数据集可能会编程线性可分的。我们使用核函数来将数据映射到高维空间，常用的核函数有：

- 多项式核函数（Polynomial Kernel）：$K(\mathbf{x}_i,\mathbf{x}_j)=(\langle\mathbf{x}_i,\mathbf{x}_j\rangle+c)^d$
- 高斯核函数（Gaussian Kernel）：$K(\mathbf{x}_i,\mathbf{x}_j)=\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|_2^2}{2\sigma^2}\right)$
- Sigmoid核函数（Sigmoid Kernel）：$K(\mathbf{x}_i,\mathbf{x}_j)=\tanh\left(\alpha\mathbf{x}_i^\top\mathbf{x}_j+c\right)$
- 余弦相似度核函数：$K(\mathbf{x}_i,\mathbf{x}_j)=\frac{\mathbf{x}_i^\top\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}$

```
from sklearn.svm import SVC
svc_iris = SVC()
svc_iris.fit(X,y)
svc_iris.score(X,y)
```

> 由于iris是一个比较简单的数据集，以上所有模型的准确率均达到了0.97+。