华东师范大学数据科学与工程学院上机实践报告

课程名称：统计方法与机器学习　　**年级**：大三　　**上机实践成绩**：

指导教师：董启文　　**姓名**：刘蔚璁　　**学号**：10225501443

实验三：分类算法——蘑菇可食用性预测

[摘要]：本实验基于蘑菇数据集，分别使用逻辑回归、决策树和支持向量机（SVM）完成蘑菇可食用性预测任务。通过实验，我们对三种分类算法在离散特征占主导的二分类问题中的性能进行了系统评估。结果显示，决策树在精度、召回率和 F1 值等指标上均表现优异，达到 100%的准确率，而自行实现的逻辑回归和 SVM 算法存在少量差错。我们分析了每种算法的特点及其适用场景，为实际数据建模提供参考。

一、目标与要求

- 使用 Python 提供的逻辑回归、决策树和 SVM 算法在蘑菇数据集上完成分类任务
- 自行实现上述三种算法，重复实验并验证结果

二、实验数据

实验使用的是 UCI 存储库提供的蘑菇数据集（Mushroom Dateset），创建于 1981 年，汇集了来自不同生态系统和地理区域的蘑菇样本，旨在通过机器学习区分蘑菇的可食用性与毒性，是生物学领域著名的多变量二分类数据集。该数据集包含 8124 个样本，其中可食用的有 4208 个样本，占 51.8%；有毒的有 3916 个样本，占 48.2%。

- **数据来源**：http://archive.ics.uci.edu/dataset/73/mushroom
- **数据内容**

  每个样本包含 23 个特征，其中 22 个是描述蘑菇属性的字符型特征，描述了伞菌科和口蘑科蘑菇的物理特性（如蘑菇帽形状、颜色、气味等），1 个是目标变量，即蘑菇的食用性（“可食用”或“有毒”）。

- **样例展示**

```
X.head(5)
```

✓ 0.0s　　Python

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	x	s	n	t	p	f	c	n	k	e	...	s	w	w	p	w	o	p	k	s	u
1	x	s	y	t	a	f	c	b	k	e	...	s	w	w	p	w	o	p	n	n	g
2	b	s	w	t	l	f	c	b	n	e	...	s	w	w	p	w	o	p	n	n	m
3	x	y	w	t	p	f	c	n	n	e	...	s	w	w	p	w	o	p	k	s	u
4	x	s	g	f	n	f	w	b	k	t	...	s	w	w	p	w	o	e	n	a	g

5 rows × 22 columns

三、实验方法

- **算法选择**
  - **Logistic 回归**

    Logistic 回归是一种常见的分类模型，属于非线性回归模型，是研究因变量为二项分类或多项分类结果与某些影响因素之间关系的一种多重回归分析方法。

    对于将预测概率作为输出的分类模型，需要将输出值限定在 0 到 1 之间。因此引入具有良好归一化性质和可微性质的 Sigmoid 函数，函数表达式如下：

$$\mathrm{Sigmoid}(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

二分类 Logistic 回归模型指的是以下条件概率分布：

$$\mathbf{P}(Y=1|\mathbf{x}) = \frac{\mathrm{e}^{\mathbf{w}'\mathbf{x}+b}}{1+\mathrm{e}^{\mathbf{w}'\mathbf{x}+b}}$$

$$\mathbf{P}(Y=0|\mathbf{x}) = 1-\mathbf{P}(Y=1|\mathbf{x}) = \frac{\mathrm{e}^{\mathbf{w}'\mathbf{x}+b}}{1+\mathrm{e}^{\mathbf{w}'\mathbf{x}+b}}$$

其中，$\mathbf{x}\in\mathbb{R}^n$是输入，$Y\in\{0,1\}$是输出，$\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}$为参数。

对于给定的实例 $\mathbf{x}$，若 $\mathbf{P}(Y=1|\mathbf{x}) > \mathbf{P}(Y=0|\mathbf{x})$，则属于 *Y* = 1 的类，否则属于 *Y* = 0 的类。可以发现，二分类 Logistic 回归模型预测概率的计算即为线性层和 Sigmoid 激活层的连接，再根据分类阈值判决属于的类别。

- **决策树**

决策树是一种常见的机器学习算法，它通过一系列条件判断对样本进行分类或预测数值，从根节点开始，根据不同的属性值逐步将样本划分到不同的分支，直至到达叶节点，每个叶节点代表一个类别。

在生成决策树的过程中，定义了样本集合 *D* 的信息熵 Ent(*D*)，用于度量样本集合的纯度，一般认为，信息熵越大，样本集合的纯度越小，其公式如下：

$$\mathrm{Ent}(D) = -\sum_{k=1}^{N} p_k \log_2 p_k$$

其中，*N* 为样本的类别数，p_k 为样本集合 *D* 中第 *k* 类样本所占的比例。

假设离散属性 *a* 有 *V* 个可能取值，若使用 *a* 来对样本集合 *D* 进行划分，则第 *v* 个分支节点包含 *D* 中所有在 *a* 上取值为 a^v 的样本，记为 D^v。假设样本数越多的分支节点影响越大，定义用 *a* 对 *D* 进行划分获得的信息增益 Gain(*D*, *a*)，公式如下：

$$\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v)$$

在训练过程中，若结点上所有样本标签相同，则样本集合纯度已经无法再提升，可直接判决为该标签；若结点已无特征可分，将返回多数类。

- **支持向量机（SVM）**

SVM 是一类按监督学习方式对数据进行二分类的广义线性分类模型，求解的是对线性可分样本学习的最大边距超平面，即如下优化问题。

$$\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2, \quad \text{s.t. } y_i(\mathbf{w}'\mathbf{x}_i+b) \geq 1$$

其中，$\mathbf{x}_i\in\mathbb{R}^n$是样本特征，$y_i\in\{-1,1\}$是判决输出，$\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}$为参数。

支持向量机实际上构造了如下的两个超平面作为间隔边界，用以判决样本分类：

$$\mathbf{w}'\mathbf{x}_i+b \geq 1 \quad \Rightarrow \quad y_i = 1$$

$$\mathbf{w}'\mathbf{x}_i+b \leq -1 \quad \Rightarrow \quad y_i = -1$$

即所有在上间隔边界上方的样本属于正类，在下间隔边界下方的样本属于负类。其中，位于间隔边界上的正类和负类样本为支持向量。

- **数据预处理**

在分类任务中，对数据进行的预处理将直接影响特征的质量以及分类模型的效果。

- **标签分布**

样本是否均衡是决定模型性能的很重要部分，所以我们先查看标签分布：

```
y.value_counts()
```

```
poisonous
e           4208
p           3916
Name: count, dtype: int64
```

可以看出标签分布基本均衡。

- **数据清洗：缺失值处理**

缺失值指的是现有数据集中某些属性的值是不完全的，可能对模型的训练结果产生影响，我们统计数据集中每个特征的缺失值情况如下：

```
missing_values = X.isnull().sum()
print("每列的缺失值数量：")
print(missing_values)

total_missing = missing_values.sum()
print(f"数据总缺失值数量：{total_missing}")
```

```
每列的缺失值数量：
cap-shape                    0
cap-surface                  0
cap-color                    0
bruises                      0
odor                         0
gill-attachment              0
gill-spacing                 0
gill-size                    0
gill-color                   0
stalk-shape                  0
stalk-root                2480
stalk-surface-above-ring     0
stalk-surface-below-ring     0
stalk-color-above-ring       0
stalk-color-below-ring       0
veil-type                    0
veil-color                   0
ring-number                  0
ring-type                    0
spore-print-color            0
population                   0
habitat                      0
dtype: int64
数据总缺失值数量：2480
```

通过统计可知，缺失值都源自于 stalk-root，该特征描述了蘑菇的根部形态。接下来详细查看这一特征的取值分布情况：

```
print(X['stalk-root'].unique())
print(X['stalk-root'].value_counts())
```

```
['e' 'c' 'b' 'r' nan]
stalk-root
b    3776
e    1120
c     556
r     192
Name: count, dtype: int64
```

可以看出，缺失值在这一特征中所占的比重较大，不妨认为缺失值本身包含信息，不对其做删除或填充处理。

- **数据处理：字符型特征编码**

查看数据集后发现，该数据集中均为非数值数据，因此使用独热编码将分类变量转换为模型可用的数值形式，其作用是将类别型数据表示为二进制特征矩阵。

具体的代码实现和编码效果如下所示：

```
encoder = OneHotEncoder(sparse_output=False)
X_encoded = encoder.fit_transform(X)
X_encoded[0,:]
```
[24] ✓ 0.0s Python

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

由此可见，数据集中的字符型特征被转换为长度为 117 的 0-1 序列。

- **实验设计**

  - **数据划分**

    本次实验将数据集按 7: 3 的比例划分为训练集和测试集，直接使用 scikit-learn 提供的数据集划分函数进行划分，并固定随机种子以确保实验结果可复现。

  - **模型训练**

    该模型训练过程较简单，各算法的核心计算公式已经在算法选择部分给出。

    ✧ 对于 Logistic 回归，只需将网络定义为线性层和 Sigmoid 层的连接进行训练；

    ✧ 对于决策树算法，只需按照上述方法逐步迭代分支，直至得到每一种特征情况下的分类判决；

    ✧ 对于 SVM，只需学习对应的超平面参数，在训练过程中，采取学习率动态调整的方案。

  - **模型评估**

    利用测试集对模型性能进行评估，计算分类任务的主要指标：

    ✧ **准确率** $\text{Accuracy} = \dfrac{\text{预测正确的样本数}}{\text{总样本数}}$

    ✧ **精确率** $\text{Precision} = \dfrac{\text{被正确分类的正类样本数}}{\text{被预测为正类的样本总数}}$

    ✧ **召回率** $\text{Recall} = \dfrac{\text{被正确分类的正类样本数}}{\text{正类样本总数}}$

    ✧ **F1 值** $\text{F1} = 2 \cdot \dfrac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

四、实验结果

- **模型性能**

  - **性能指标（调用 Python 库）**

    计算指标时，将“可食用（e）”作为正类，得到不同算法的指标如下表所示。

算法	准确率	精确率	召回率	F1 值
Logistic 回归	99.96%	99.92%	100.00%	99.96%
决策树	100.00%	100.00%	100.00%	100.00%
SVM	100.00%	100.00%	100.00%	100.00%

可以看出 Python 提供的不同模型算法在该数据集上都有很好的表现，仅有 Logistic 回归出现了少量的差错，均为将负类错判为正类。

➢ **性能指标（自行实现算法）**

不直接调用 Python 提供的算法模型，根据公式实现相应的训练算法进行测试，得到的不同算法的指标如下表所示。

算法	准确率	精确率	召回率	F1 值
Logistic 回归	99.88%	99.84%	99.92%	99.88%
决策树	100.00%	100.00%	100.00%	100.00%
SVM	99.63%	99.28%	100.00%	99.64%

五、分析与讨论

● 不同算法在蘑菇数据集上的表现

实验结果表明，不论是直接调用 Python 提供的算法还是自行实现算法，三种算法都取得了较好的表现，对不同算法的实验结果及其分析如下：

✧ Logistic 回归算法在蘑菇数据集上存在微小的差错，可能是由于 Logistic 回归属于简单的广义线性模型，对于多变量非线性的分类问题表达能力有限，相比于更高级的机器学习算法，决策边界相对粗糙，可能不足以达到极致的准确判断。

✧ 决策树算法在测试集上始终保持 100%的准确率，是三种算法中表现最好的算法。这是由于决策树算法天然地可以处理离散特征，并且能够在进行分支时产生复杂的决策边界，能够对精细的特征进行准确学习。

✧ SVM 算法在测试集上存在极少差错，但整体在蘑菇数据集上表现较好，出现差错的原因可能是 SVM 算法将数据映射到高维空间中，对数据集中存在的缺失值比较敏感，并且训练参数的选取对 SVM 的训练效果也容易产生影响。

● 改进建议

✧ **特征工程：**可通过增加交互特征或高阶特征以提高模型对复杂关系的刻画能力。

✧ **数据预处理：**针对缺失的特征，除了我们采用的直接保留的方法之外，也可尝试用插值法或统计方法填充，分析其对模型性能的影响。

✧ **模型选取：**可考虑使用神经网络等高级机器学习算法，或结合多种算法（如逻辑回归+决策树）构建集成模型，提高预测性能的同时增强鲁棒性。