6



# 数据预处理：**编码**
# **(Encoding)**

授课教师：[李阳源]

# SL02:今日学习目标



- 理解分类数据的特征与编码的必要性
- 学习主要编码技术
- 标签编码 (Label Encoding)
- 独热编码 (One-Hot Encoding)
- 掌握多类别处理方法
- 介绍使用Python进行编码的方法

# SL03: 什么是编码?



- **编码(Encoding)**: 将分类数据转换为机器可理解的数字形式的过程
- 大多数机器学习算法只能处理数值数据
- 将文本或分类数据数值化以用于模型训练

# SL04: 分类数据类型



1. **名义型(Nominal)**: 无序的类别
   - 例：性别(男，女)，血型(A, B, AB, O)，颜色(红，蓝，绿)

2. **顺序型(Ordinal)**: 有序的类别
   - 例：学历(小学，初中，高中，大学)
   - 例：满意度(非常不满意，不满意，一般，满意，非常满意)

# SL05: 为什么需要编码?



- 机器学习算法只能处理数值数据
- 无法直接使用分类变量
- 通过编码使其可用于模型训练
- 为了正确表示特征之间的关系
- 需要根据算法选择适当的编码方式

# SL06: 标签编码 (Label Encoding)



- 概念: 为每个类别分配唯一的整数值
- 优点: 简单且内存效率高
- 缺点: 应用于无序类别时可能产生错误的顺序关系

# SL07: 标签编码示例



| 原始数据 | 标签编码 |
| --- | --- |
| 红 | 0 |
| 蓝 | 1 |
| 绿 | 2 |
| 红 | 0 |
| 绿 | 2 |

- 注意数字大小没有意义 (0 < 1 < 2 关系不存在)
- 适合顺序型变量

# SL08: 标签编码应用场景



- 适合的情况:
  - 基于决策树的算法
  - 处理顺序型变量
  - 随机森林，XGBoost 等
- 不适合的情况:
  - 线性回归，逻辑回归
  - SVM，神经网络等基于距离的算法

# SL09: 独热编码 (One-Hot Encoding)



- **概念: 将每个类别表示为二进制向量**
- **每个类别转换为单独的列(特征)**
- **如果属于该类别则为1，否则为0**

**python**

**# 例: ['红', '蓝', '绿'] →**

**# [[1,0,0], [0,1,0], [0,0,1]]**

# SL10: 独热编码示例



| 原始数据 | 红(Red) | 蓝(Blue) | 绿(Green) |
|---|---|---|---|
| 红 | 1 | 0 | 0 |
| 蓝 | 0 | 1 | 0 |
| 绿 | 0 | 0 | 1 |
| 红 | 1 | 0 | 0 |
| 绿 | 0 | 0 | 1 |

- 表示类别间的平等关系
- 适合基于距离的算法

# SL11: 独热编码的优缺点



**优点:**

- 消除类别间错误的顺序关系
- 适合名义型变量
- 适合线性模型

**缺点:**

- 维度增加 (维度灾难)
- 生成稀疏矩阵
- 类别多时内存使用量增加

# SL12: 虚拟变量 (Dummy Variables)



- 独**热编码的变形**
- 将**N个类别表示为N-1个变量**
- **防止多重共线性**
- **设置基准类别**

**python**

**# 例: ['红', '蓝', '绿'] →**

**# [[1,0], [0,1], [0,0]] # 绿色为基准类别**

# SL13: 多类别处理 (Multi-category)



**多类别处理 (Multi-category)**

- **多标签**: 一个数据可能属于多个类别
- **多类别**: 存在多个可能的类别

**处理方法:**

1. 多标签二值化 (MultiLabelBinarizer)
2. 多热编码 (Multi-hot encoding)

# SL14: 多类别处理示例



- **原始数据:**

  **标签: ['体育, 户外活动', '音乐, 艺术', '烹饪, 食物, 旅行']**

- **多热编码结果:**

  **[1, 1, 0, 0, 0, 0] # 体育, 户外活动**

  **[0, 0, 1, 1, 0, 0] # 音乐, 艺术**

  **[0, 0, 0, 0, 1, 1, 1] # 烹饪, 食物, 旅行**

# SL15: Python中的标签编码



```python
from sklearn.preprocessing import LabelEncoder

# 数据示例
colors = ['红', '蓝', '绿', '红', '绿']

# 初始化并应用标签编码器
encoder = LabelEncoder()
encoded_colors = encoder.fit_transform(colors)

print(encoded_colors)  # 输出: [2 0 1 2 1]
print(encoder.classes_)  # 输出: ['蓝' '绿' '红']
```

# SL16: Python中的独热编码



```python
import pandas as pd

# 数据示例
colors = ['红', '蓝', '绿', '红', '绿']
df = pd.DataFrame({'颜色': colors})

# 使用pandas的get_dummies
one_hot = pd.get_dummies(df['颜色'])
print(one_hot)
```

**输出:**

| | 绿 | 红 | 蓝 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 |

# SL17: Scikit-learn的OneHotEncoder



```python
from sklearn.preprocessing import OneHotEncoder
import numpy as np

# 数据示例
colors = ['红', '蓝', '绿', '红', '绿']
colors = np.array(colors).reshape(-1, 1)

# 初始化并应用OneHotEncoder
encoder = OneHotEncoder(sparse=False)
encoded = encoder.fit_transform(colors)

print(encoded)
print(encoder.categories_)
```

# SL18: 编码选择指南



| 情况 | 推荐编码方式 |
| --- | --- |
| 顺序型变量 | 标签编码或顺序编码 |
| 名义型变量 | 独热编码 |
| 基于树的模型 | 标签编码 |
| 线性/基于距离的模型 | 独热编码 |
| 高维分类变量 | 频率/目标编码 |
| 文本数据 | 词嵌入 |

# SL19: 编码相关注意事项



1. **新类别处理: 测试数据中出现训练数据中没有的类别**
2. **维度灾难: 独热编码时随着类别增加而增加的维度**
3. **稀疏矩阵: 大型数据集的内存效率**
4. **数据泄露(Data Leakage): 使用目标编码时需要注意**

# SL20: 理解度测试



1. 以下哪种编码方式最适合名义型变量？

a) 标签编码 b) 独热编码 c) 顺序编码 d) 频率编码

2. 以下哪项不是独热编码的缺点？

a) 维度增加 b) 内存使用量增加 c) 产生顺序关系 d) 生成稀疏矩阵

# SL21: 总结与下节课预告



**今天学习的内容:**

- 分类数据的特征与编码必要性
- 标签编码与独热编码的概念及特点
- 多类别处理方法
- 使用Python实现编码

**下节课预告:**

- 文本预处理概述
- 分词、停用词去除、正则表达式等

# 6 PL01: 课程回顾



- 上周我们学习了数据缩放与归一化技术
- 我们使用了 sklearn.preprocessing 包中的各种工具
- 今天我们将学习如何处理分类数据

# PL02: 本次课程目标



- 理解分类数据编码的必要性
- 掌握 LabelEncoder 的使用方法
- 学习 OneHotEncoder 的应用场景
- 熟悉 pandas 的 get_dummies 函数
- 通过实例练习巩固所学知识

# PL03:为什么需要编码?



- 机器学习算法通常只能处理数值型数据
- 分类变量（如颜色、性别、国家等）无法直接用于计算
- 编码可以将分类变量转换为数值形式
- 不同的编码方式适用于不同的场景

# PL04: 分类数据类型



- **名义型变量（Nominal Variables）**：没有顺序关系的类别
  - 例如：颜色、国家、性别
- **顺序型变量（Ordinal Variables）**：有顺序关系的类别
  - 例如：教育程度（小学、中学、大学）、满意度评级（低、中、高）

# PL05: 常用编码方法



- 标签编码（Label Encoding）：将类别映射为整数
- 独热编码（One-Hot Encoding）：将类别转换为二进制向量
- 二进制编码（Binary Encoding）：将整数转换为二进制表示
- 目标编码（Target Encoding）：根据目标变量的平均值编码

# PL06: 开发环境配置



- 打开 PyCharm IDE
- 创建新项目 encoding_practice
- 创建新的 Python 文件 label_encoding.py
- 安装必要的库

```
pip install pandas numpy scikit-learn matplotlib
```

# PL07: 示例数据集



- 我们将使用一个简单的学生信息数据集
- 创建文件：

**student_data.csv**

**包含以下数据:**

```
id,name,gender,education,major,grade
1,张三,男,本科,计算机科学,A
2,李四,女,研究生,数据科学,B
3,王五,男,本科,人工智能,C
4,赵六,女,博士,计算机科学,A
5,钱七,男,研究生,数据科学,B
6,孙八,女,本科,人工智能,A
7,周九,男,博士,计算机科学,C
8,吴十,女,研究生,数据科学,B
```

# PL08: 加载数据



- 创建 load_data.py 文件

# PL09: 查看数据类型



- a 在 load_data.py 中添加代码：

# PL10: 什么是标签编码 (Label Encoding)?



- 将分类变量转换为数字标签
- 每个唯一的类别分配一个整数
- 例如：男=0, 女=1
- 适用于有序变量，如教育程度
- 存在的问题：可能引入不存在的顺序关系

# PL11: 使用 LabelEncoder



- 创建文件 label_encoding.py

# PL12: 运行标签编码示例



- 在 PyCharm 中运行 label_encoding.py
- 观察输出结果
- 注意编码是如何映射的
- 思考：这种编码方式对"教育程度"是否合适？为什么？

# PL13: LabelEncoder 的局限性



- 引入了不必要的顺序关系
  - 例如："计算机科学"=0, "数据科学"=1, "人工智能"=2，这暗示了某种顺序
- 数值之间的距离没有实际意义
- 在分类算法中可能造成错误的解释
- 对于名义变量，独热编码是更好的选择

# PL14: 什么是独热编码 (One-Hot Encoding)？



- 为每个类别创建一个二进制特征
- 只有一个值为1，其余为0
- 例如：[男, 女] 转换为 [1, 0] 和 [0, 1]
- 消除了类别之间的顺序关系
- 适用于名义变量

# PL15: 使用 OneHotEncoder



- 创建文件 onehot_encoding.py

# PL16: 运行独热编码示例



- 在 PyCharm 中运行 onehot_encoding.py
- 观察输出结果
- 注意每个类别如何转换为单独的列
- 思考：独热编码与标签编码相比有什么优势？

# PL17: Pandas 的 get_dummies 函数



- pandas 提供了更简单的方法：get_dummies
- 自动为分类变量创建独热编码
- 无需单独处理每个变量
- 可以同时处理多个分类变量

# PL18: 使用 get_dummies



- 创建文件 get_dummies_example.py

# PL19: 运行 get_dummies 示例



- 在 PyCharm 中运行 get_dummies_example.py
- 观察如何一次性处理多个分类变量
- 注意输出的列名格式
- 比较与 OneHotEncoder 的异同

# PL20: get_dummies 参数详解



- 创建文件 get_dummies_params.py

# PL21: 编码的实际应用场景



- 机器学习预处理: 几乎所有算法都需要数值输入
- 数据可视化: 分类数据可能需要编码后才能绘图
- 特征工程: 创建更有意义的特征
- 实例: 预测学生成绩

# PL22: 实战练习：学生成绩预测



- 创建文件 grade_prediction.py

# PL23: 处理高基数分类变量



- **高基数分类变量: 包含大量唯一值的分类变量**
  - **例如：国家、城市、产品ID**
- **独热编码会导致特征爆炸**
- **解决方案:**
  - **二进制编码 (Binary Encoding)**
  - **哈希编码 (Hash Encoding)**
  - **目标编码 (Target Encoding)**

# PL24: 编码方法比较



| 编码方法 | 优点 | 缺点 | 适用场景 |
| --- | --- | --- | --- |
| 标签编码 | 简单，保持单一特征 | 引入顺序关系 | 顺序型变量 |
| 独热编码 | 无顺序关系，直观 | 增加特征数量 | 低基数名义变量 |
| 二进制编码 | 特征数少于独热编码 | 不直观 | 高基数分类变量 |
| 目标编码 | 特征少，保留信息 | 可能过拟合 | 高基数分类变量 |

# PL25:实践技巧



- 选择合适的编码方法要考虑:
  - 变量的类型（名义型或顺序型）
  - 类别的数量（基数）
  - 机器学习算法的特性
- 在交叉验证中验证编码效果
- 考虑特征选择减少维度

# PL26: 作业练习



1. 创建一个包含至少5个分类特征的数据集
2. 应用标签编码和独热编码处理数据
3. 比较两种编码方法的区别
4. 思考每种编码方法适合什么类型的机器学习算法

# PL27: 课程总结



1. 分类数据编码是数据预处理的重要步骤
2. 标签编码适合顺序型变量
3. 独热编码适合名义型变量
4. Pandas 的 get_dummies 函数提供了简便的独热编码方法
5. 选择合适的编码方法需要考虑数据特性和算法需求

# PL28: 下节课预告



- 下周我们将学习字符串及文本处理实践
- 内容包括:
  - .str 方法
  - 正则表达式
  - 简单形态分析
- 请提前复习 Python 字符串基础知识

# PL29: 参考资源



1. Scikit-learn 官方文档: https://scikit-learn.org/stable/modules/preprocessing.html
2. Pandas 文档: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html
3. 《Python数据科学手册》- Jake VanderPlas
4. 《机器学习实战》- Peter Harrington