**人工智能理论及应用**

**实验报告**



**姓名：\_\_刘鹏程\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**班级：\_\_新一代 非全\_\_\_\_\_\_\_\_\_**

**学号：\_\_Y60250616\_\_\_\_\_\_\_\_\_\_\_**

# 基础实验

### 实验名称：分类问题实验——逻辑回归、决策树

### 实验用时：2小时

### 实验内容

1. 鸢尾花数据集介绍：包含 150 个样本，分为 3 个类别（山鸢尾、变色鸢尾、维吉尼亚鸢尾），每个样本有 4 个特征（花萼长度、花萼宽度、花瓣长度、花瓣宽度）；
2. 分类算法原理：逻辑回归基于 sigmoid 函数构建线性分类模型；K 近邻通过计算样本间距离，选取最近的 K 个样本投票确定类别；决策树通过特征分裂构建树状分类结构；
3. 模型评估指标：准确率为正确分类样本数占总样本数的比例；精确率为预测为正类的样本中实际为正类的比例；召回率为实际为正类的样本中被正确预测的比例；F1 值为精确率和召回率的调和平均数；混淆矩阵直观展示各类别样本的预测结果。

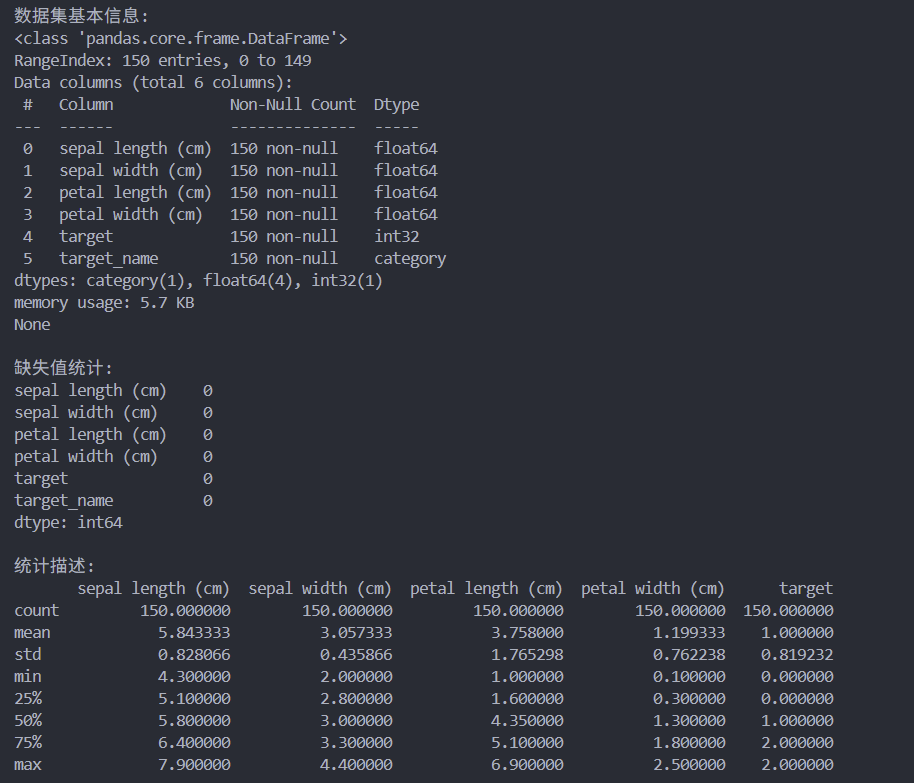
### 实验步骤

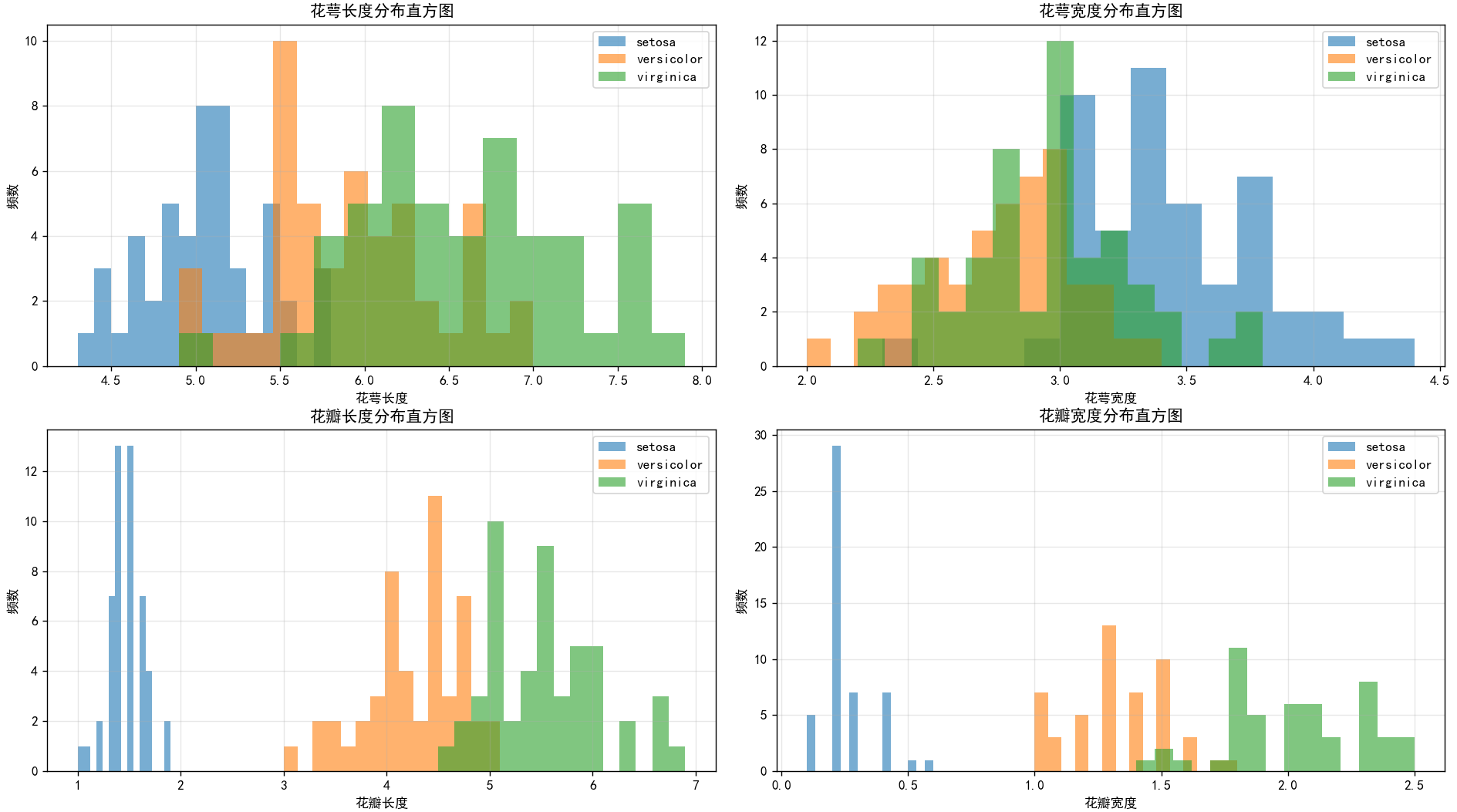
1. 数据加载与探索：使用 Scikit-learn 库加载鸢尾花数据集，通过 Pandas 查看数据基本信息（数据形状、缺失值、统计描述），使用 Matplotlib 绘制特征分布直方图、散点图，分析特征与类别的关系；​
2. 数据预处理：将数据集划分为训练集（70%）和测试集（30%），对特征进行标准化或归一化处理；​
3. 模型构建与训练：分别构建逻辑回归、K 近邻、决策树分类模型，使用训练集进行模型训练，调整模型关键参数（如 K 近邻的 K 值、决策树的深度）；​
4. 模型预测与评估：使用测试集进行模型预测，计算各模型的准确率、精确率、召回率、F1 值，绘制混淆矩阵，对比分析不同模型的性能；​
5. 实验总结：记录实验过程中的关键步骤、结果，分析模型性能差异的原因，总结实验收获与体会。​

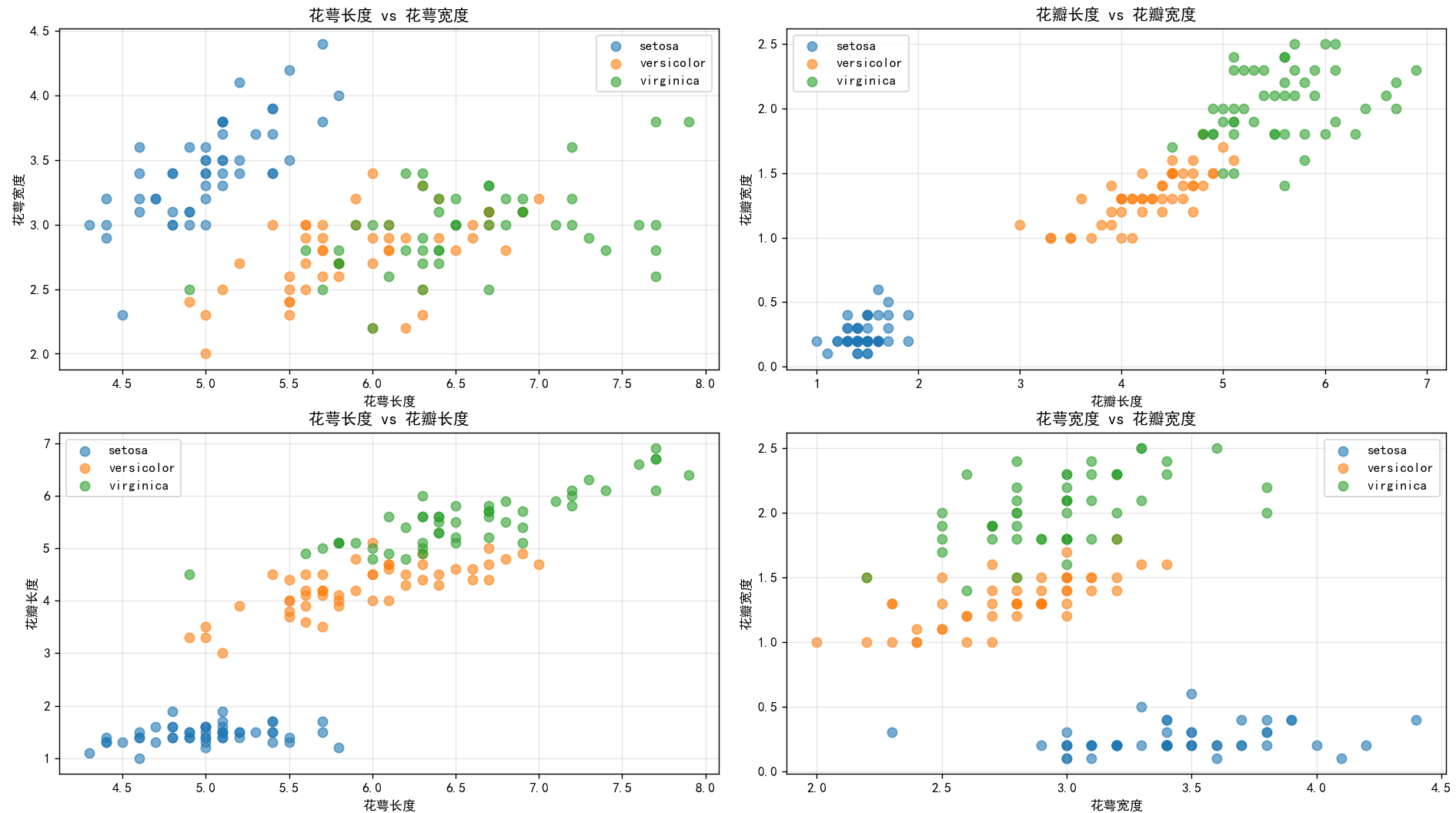
### 实验分析

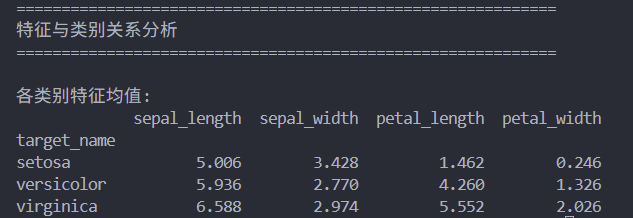
1. 数据加载与探索



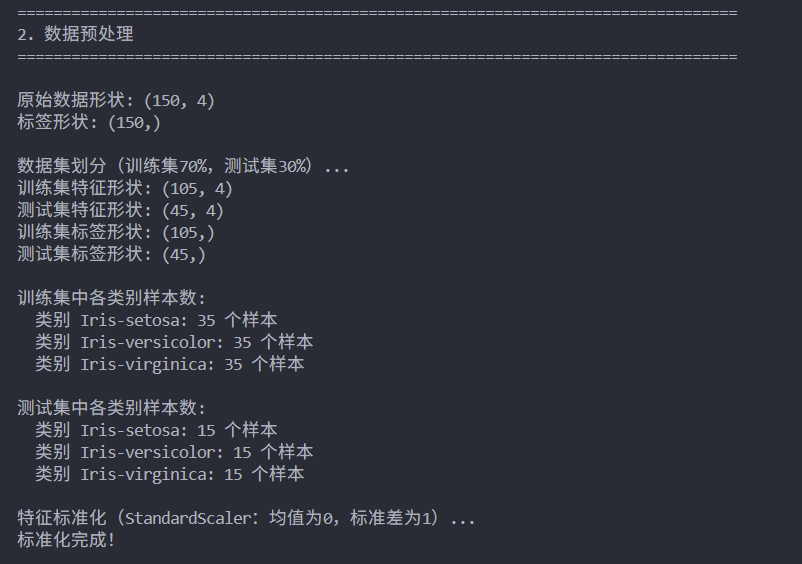




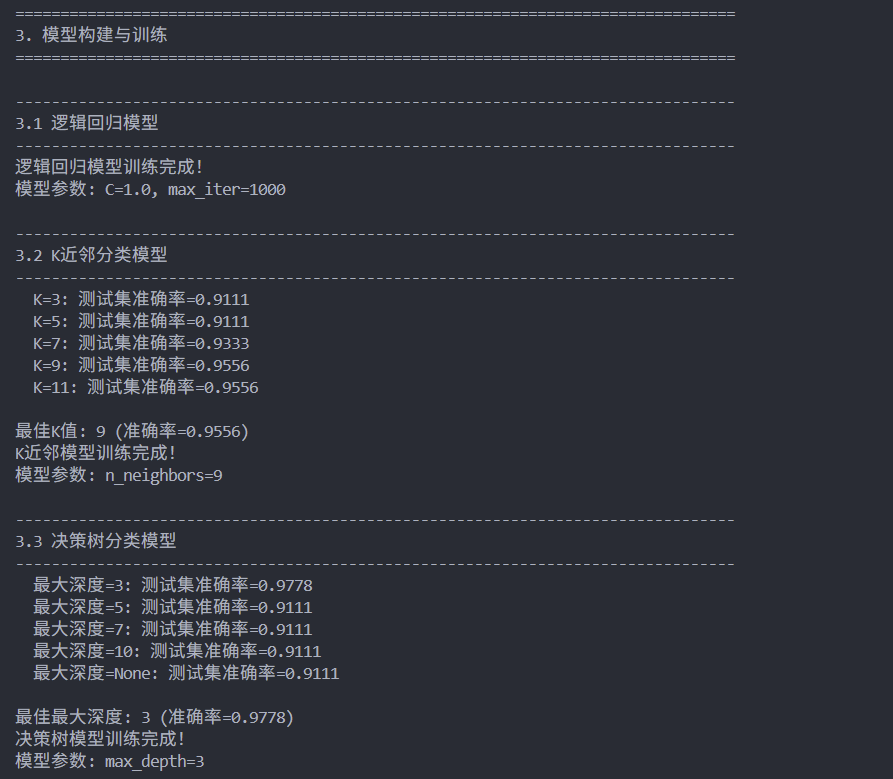




1. 数据预处理

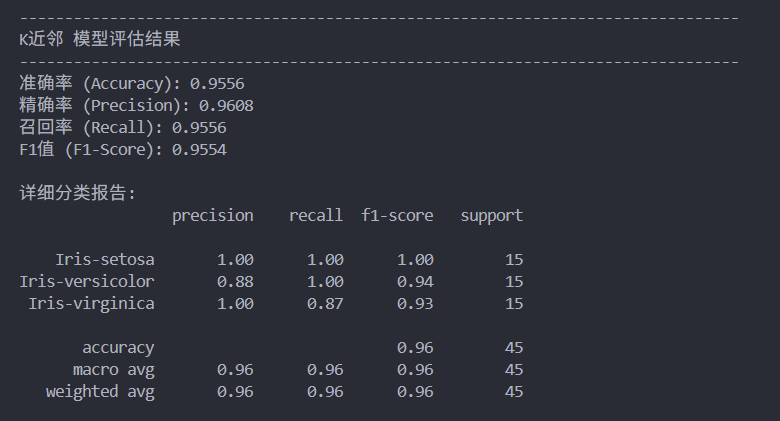


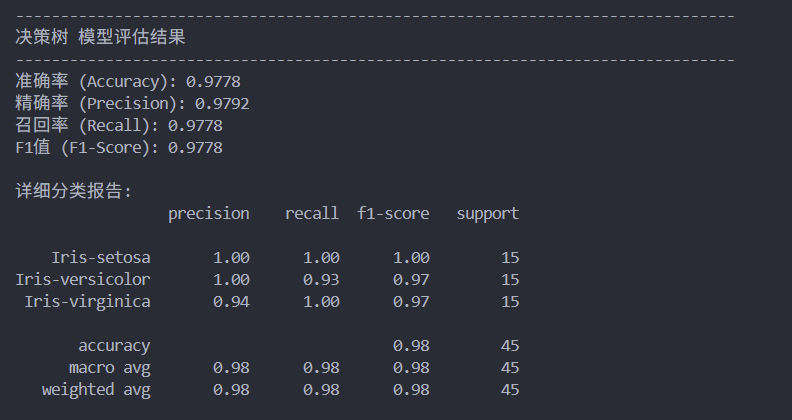
1. 模型构建与训练



1. 模型预测与评估

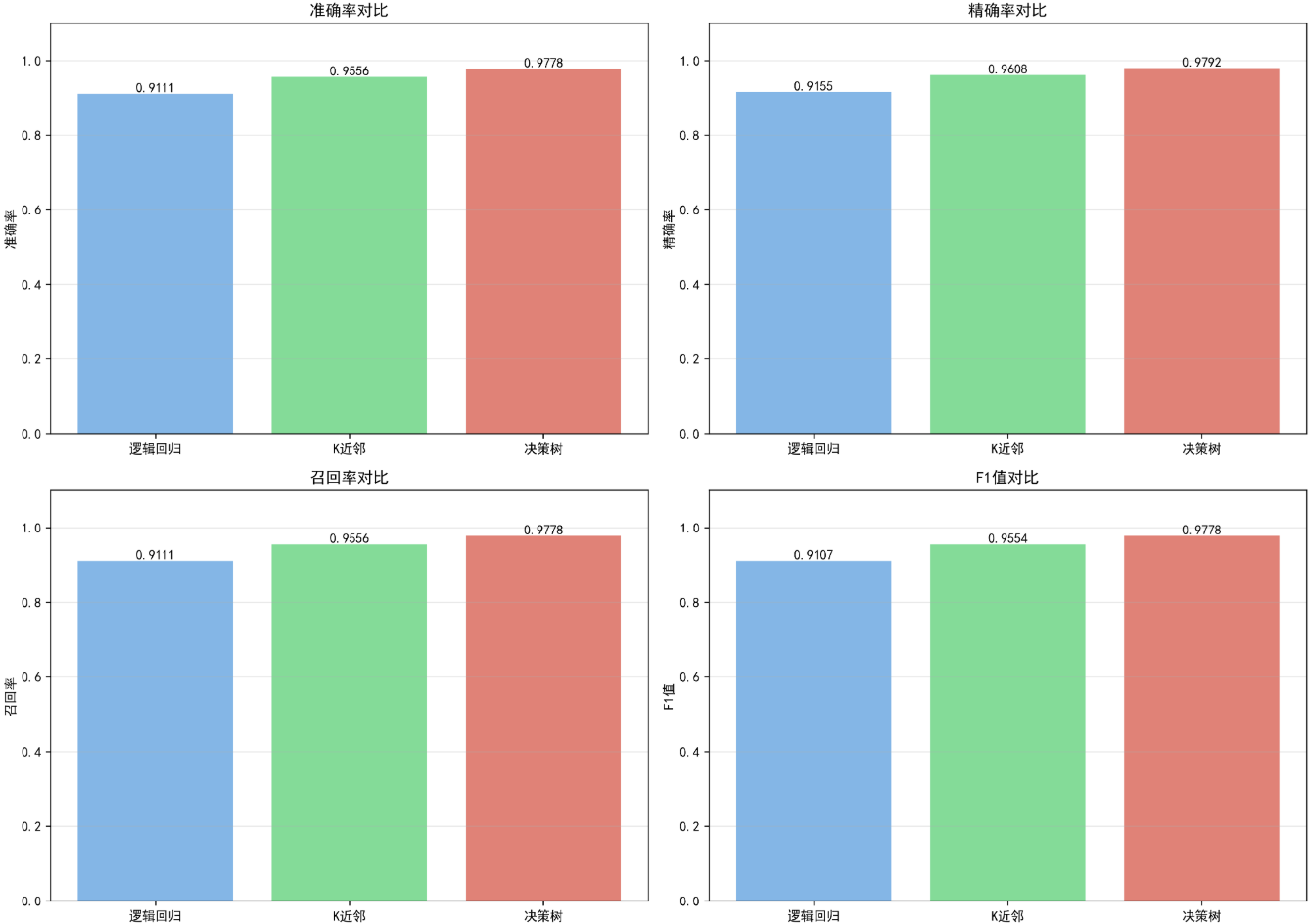












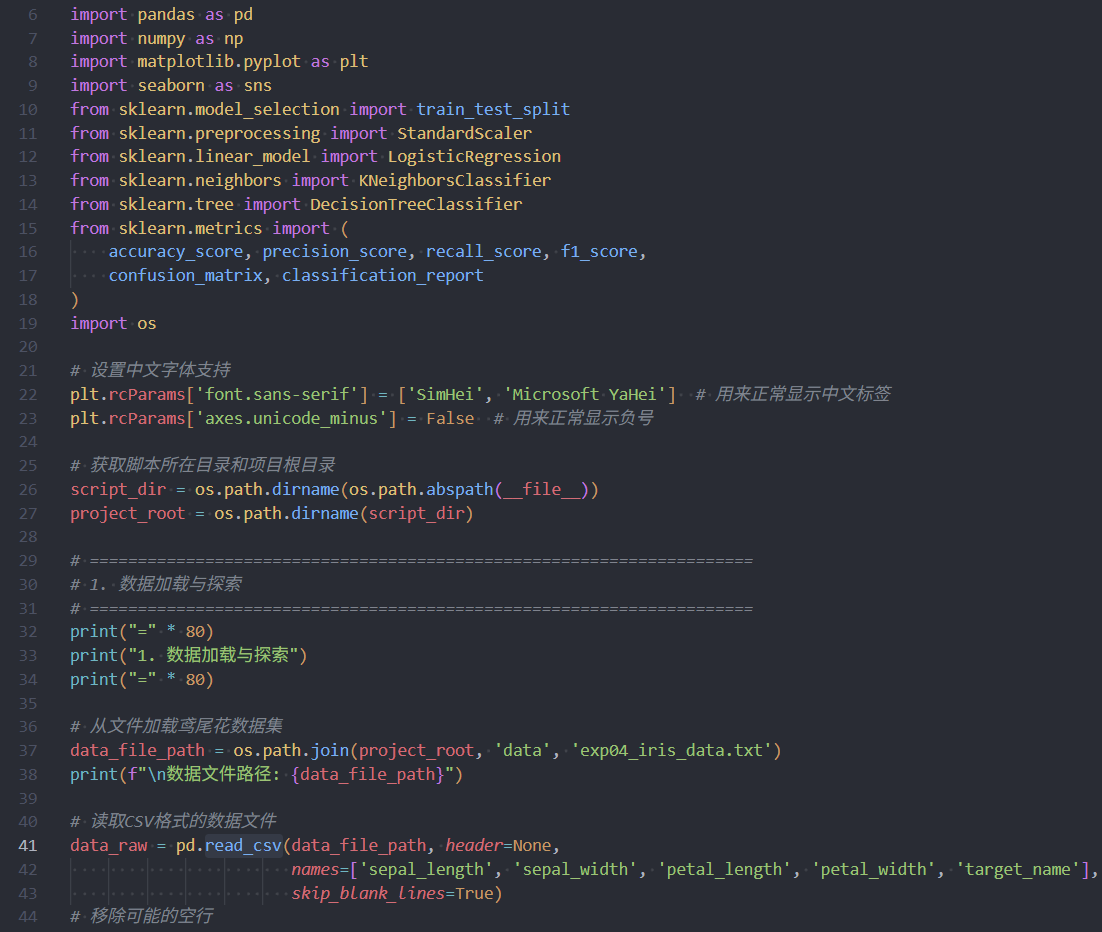
1. 实验总结

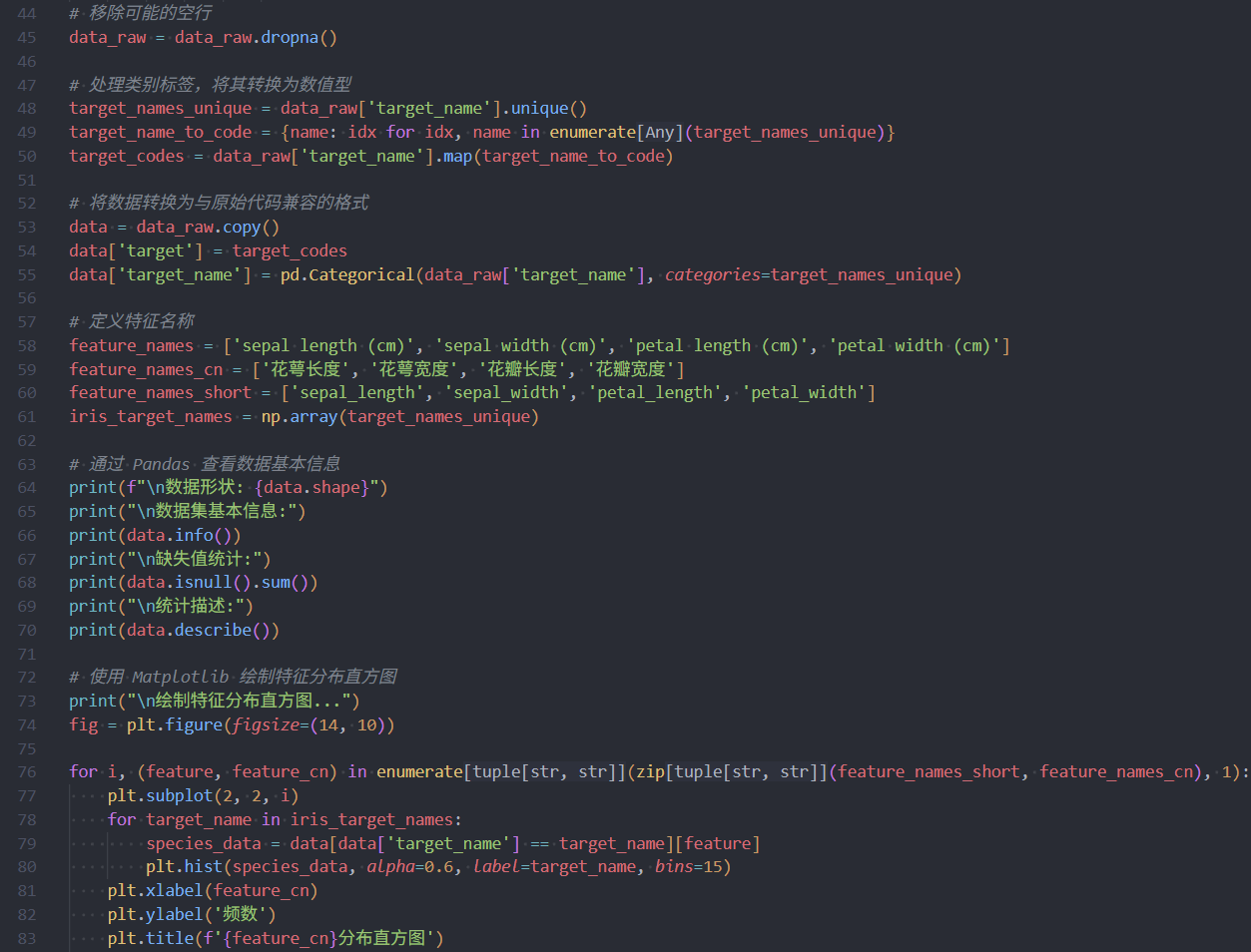
决策树在该数据集中表现最好（97.78%），其原因如下：

* 非线性建模能力强，能捕捉复杂边界
* 自动特征选择，专注于花瓣特征（重要性55%和45%）
* 深度控制（max\_depth=3）避免过拟合
* 仅1个误判样本

花瓣长度和宽度是区分类别的关键（决策树仅使用这两个特征）

### 程序截图









### 实验名称：神经网络基础——BP 神经网络

### 实验用时：3小时

### 实验内容

1. BP 神经网络：是一种多层前馈神经网络，通常包含输入层、隐藏层（1 层或多层）、输出层，激活函数采用 Sigmoid、ReLU 等可微函数；​

* 前向传播：输入数据通过输入层、隐藏层计算，得到输出层的预测结果；​
* 反向传播：根据预测结果与真实值的误差，从输出层反向计算各层的误差项，利用梯度下降法更新各层的权重和偏置；​

1. 损失函数：分类问题常用交叉熵损失函数，回归问题常用均方误差损失函数。

### 实验步骤

1. 数据：鸢尾花数据集
2. BP 神经网络实现：​

* 定义 BP 神经网络类，设置输入层维度、隐藏层维度（如 1 层隐藏层，10 个神经元）、输出层维度，初始化各层权重和偏置；​
* 实现前向传播：计算隐藏层输出（激活函数用 ReLU）、输出层输出（分类问题用 Sigmoid/Softmax，回归问题无激活函数）；​
* 实现反向传播：计算输出层误差、隐藏层误差，计算权重和偏置的梯度，使用梯度下降法更新参数（设置学习率、迭代次数）；​
* 使用数据集鸢尾花分类训练 BP 神经网络，记录训练过程中的损失值变化，绘制损失曲线；​

1. BP 神经网络框架实现（PyTorch）：​

* 使用 PyTorch 构建 BP 神经网络模型（Sequential 模型）；​
* 配置模型训练参数（优化器如 Adam、损失函数、评估指标）；​
* 训练模型，使用验证集监控模型性能，防止过拟合（如早停法）；​
* 评估模型在测试集上的性能，对比手动实现与框架实现的结果差异​

1. 实验分析：分析学习率、迭代次数、隐藏层神经元数量对模型训练效果和性能的影响。

### 实验分析

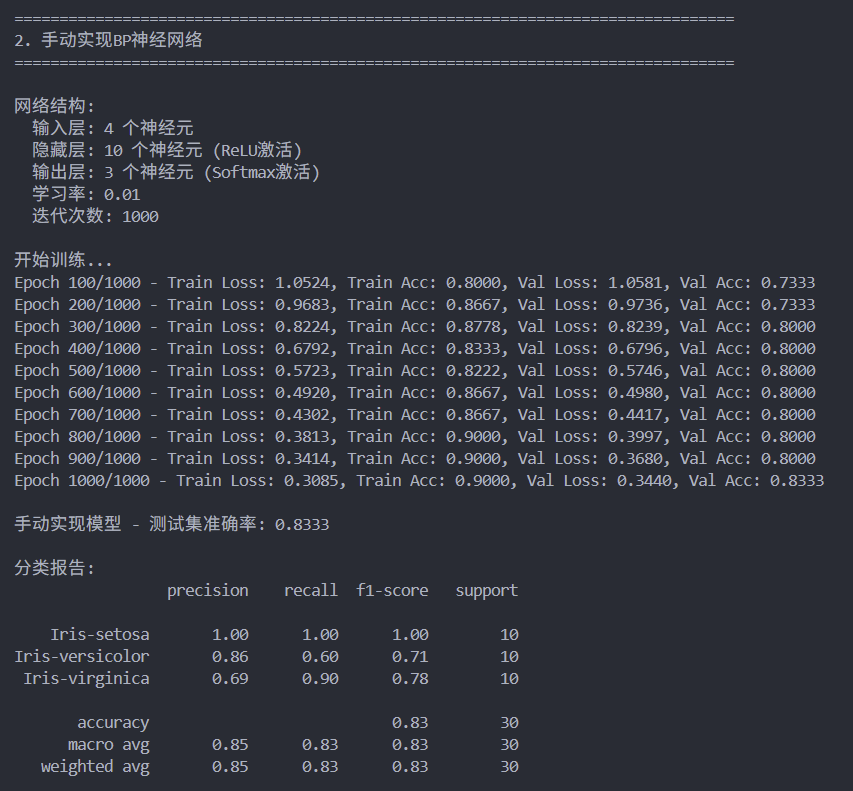
1. 手动实现BP神经网络

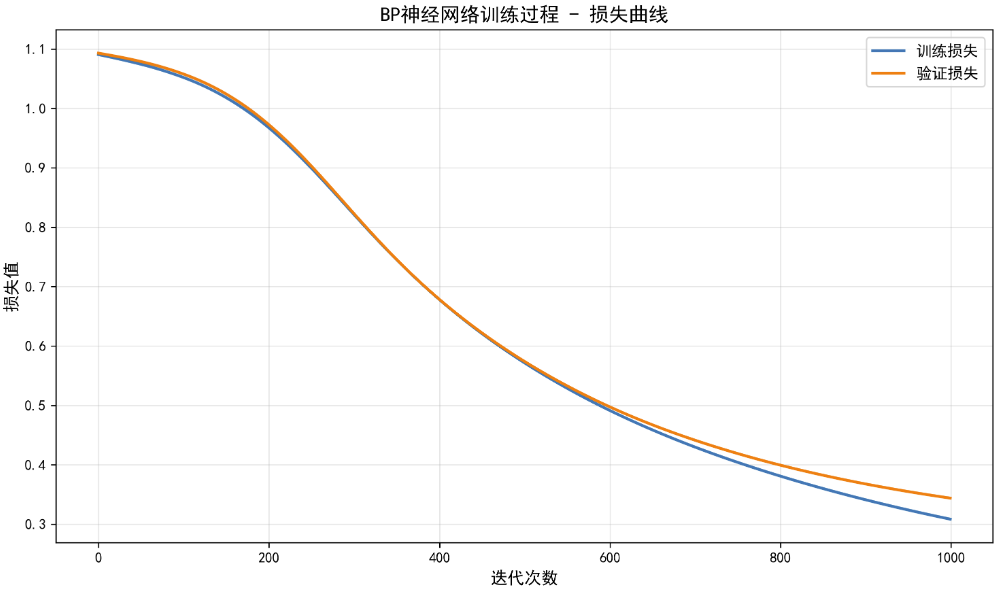
- 网络结构：输入层(4) → 隐藏层(10, ReLU) → 输出层(3, Softmax)

- 实现前向传播和反向传播算法

- 使用梯度下降法更新参数

- 记录训练过程中的损失值变化





1. PyTorch框架实现

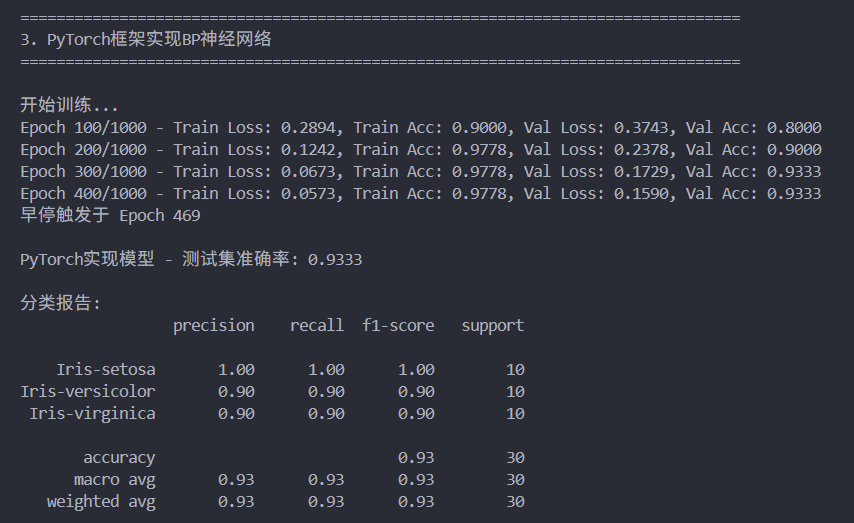
- 使用Sequential模型构建网络

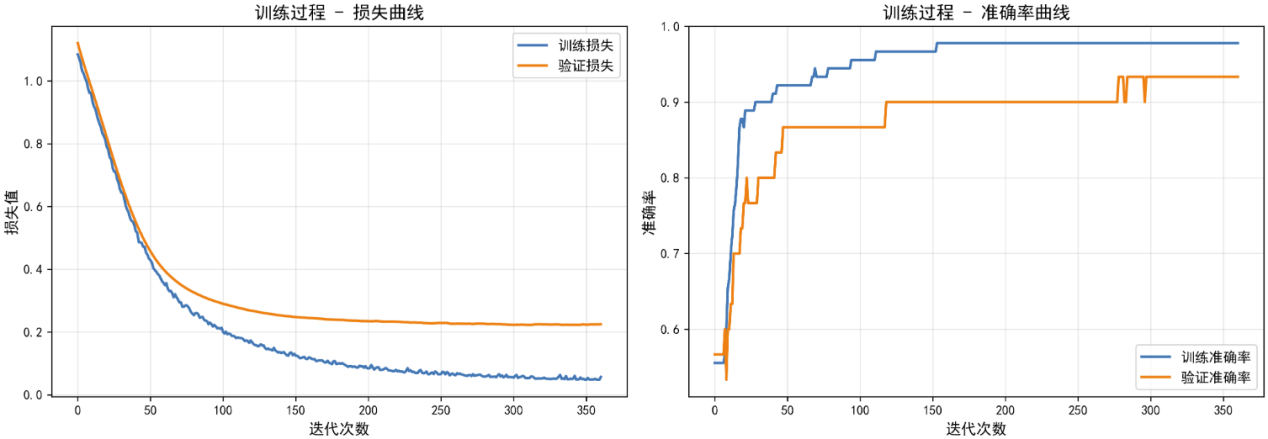
- 配置Adam优化器

- 使用交叉熵损失函数

- 实现早停法防止过拟合

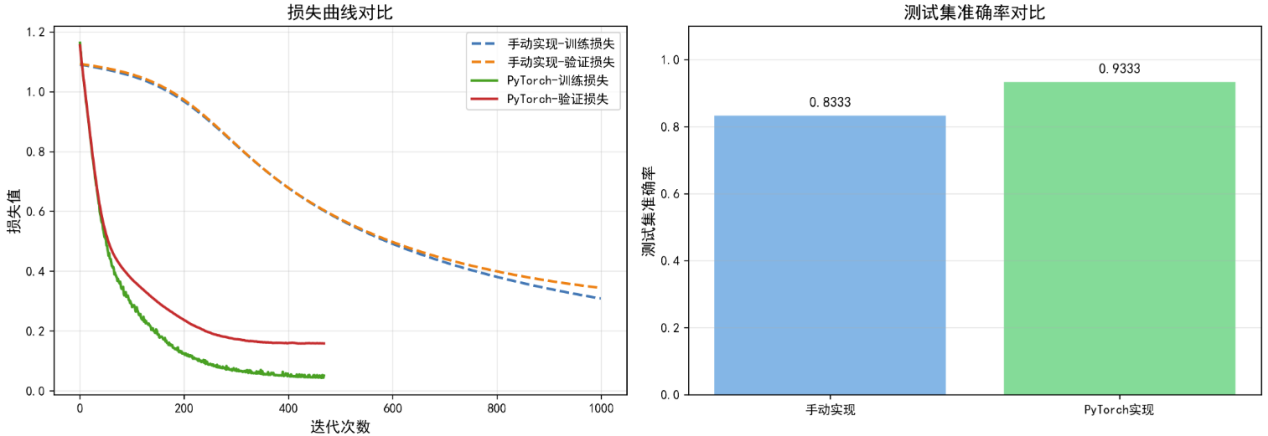
- 使用验证集监控模型性能

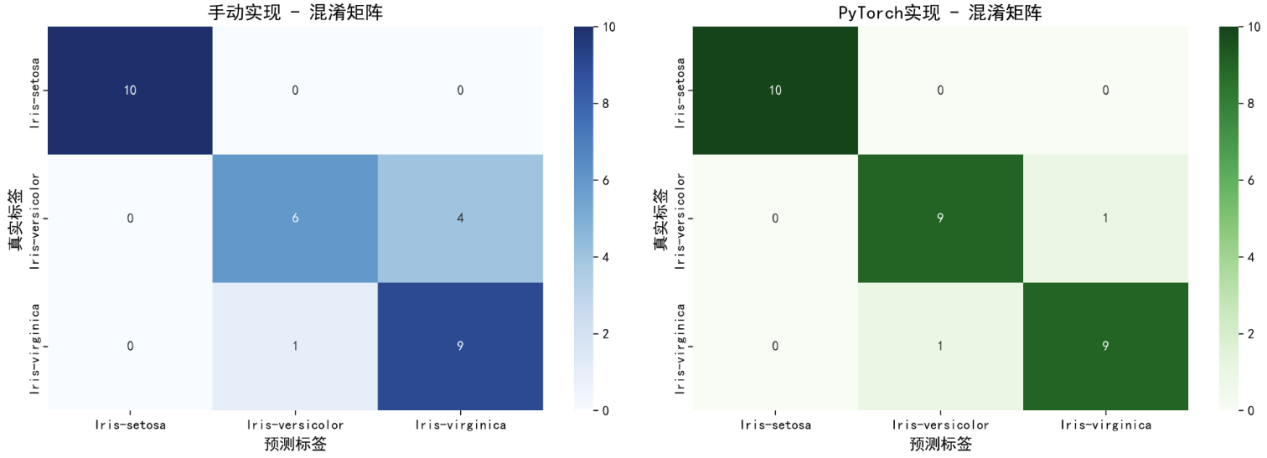




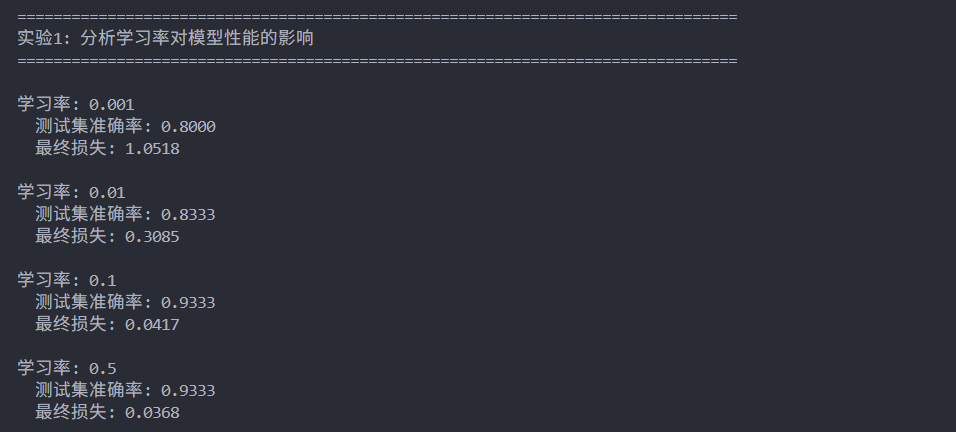
1. 手动实现与框架实现的结果差异

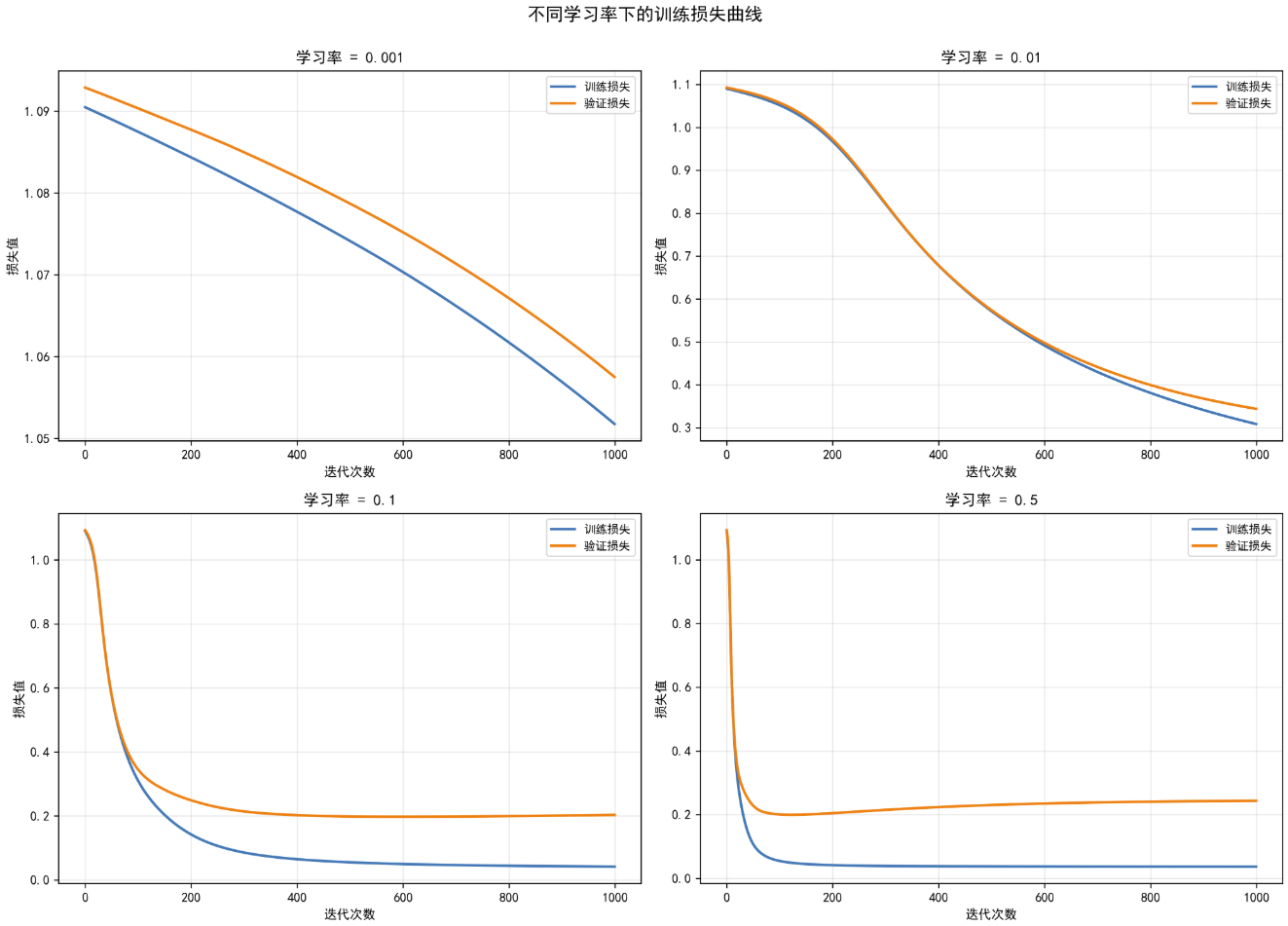


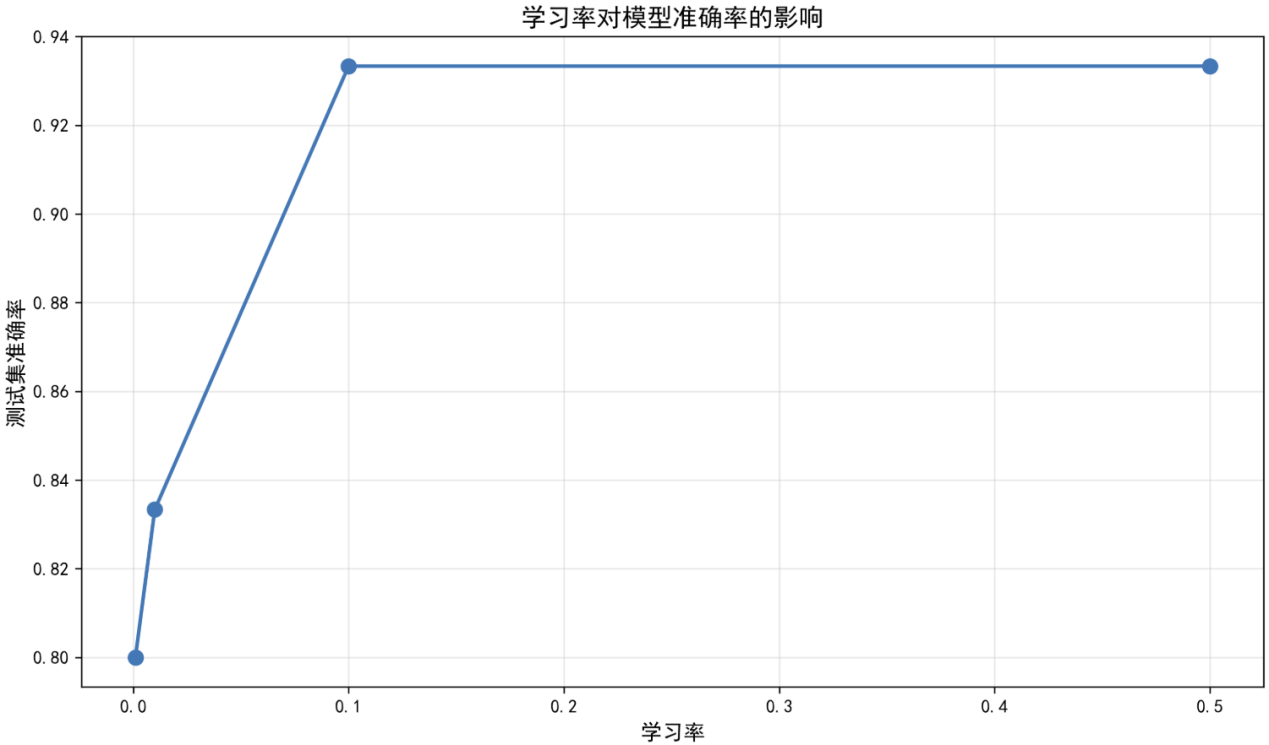


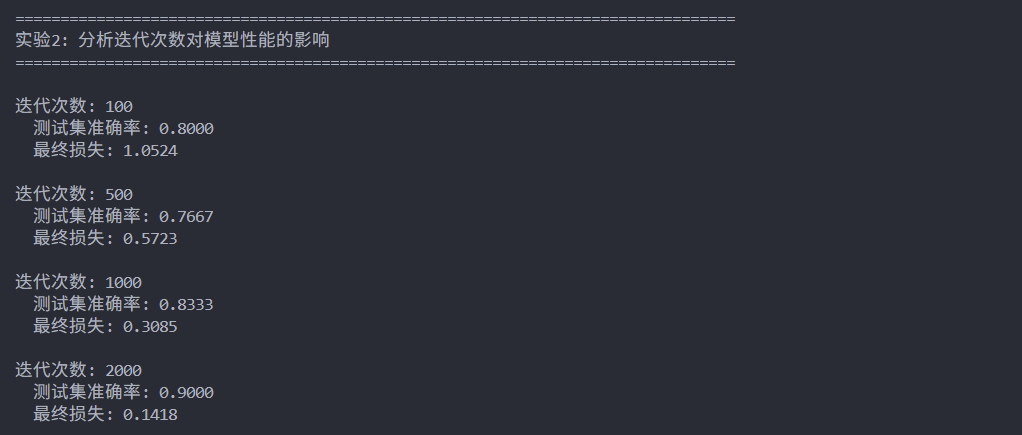


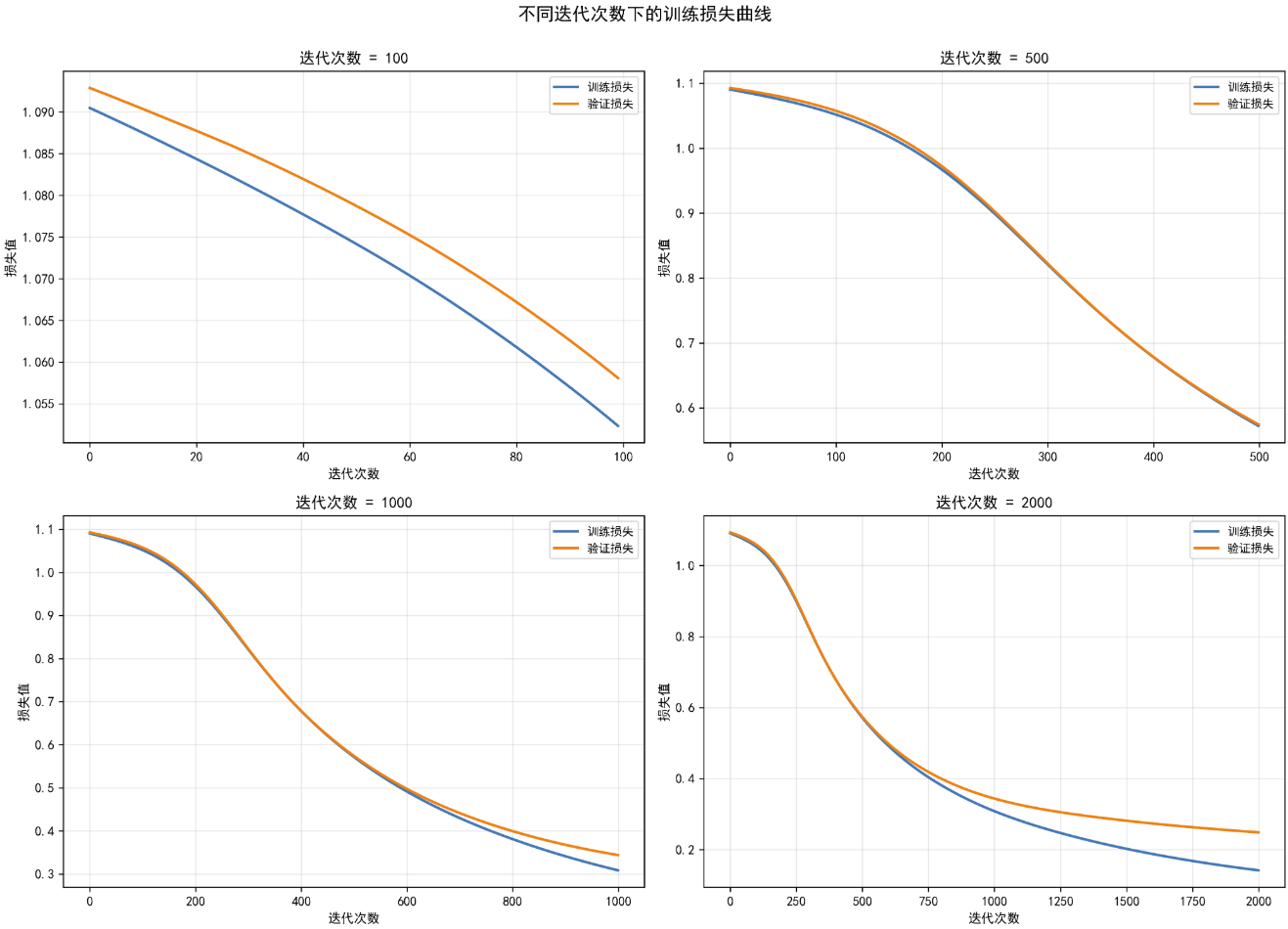
1. 学习率、迭代次数、隐藏层神经元数量的影响

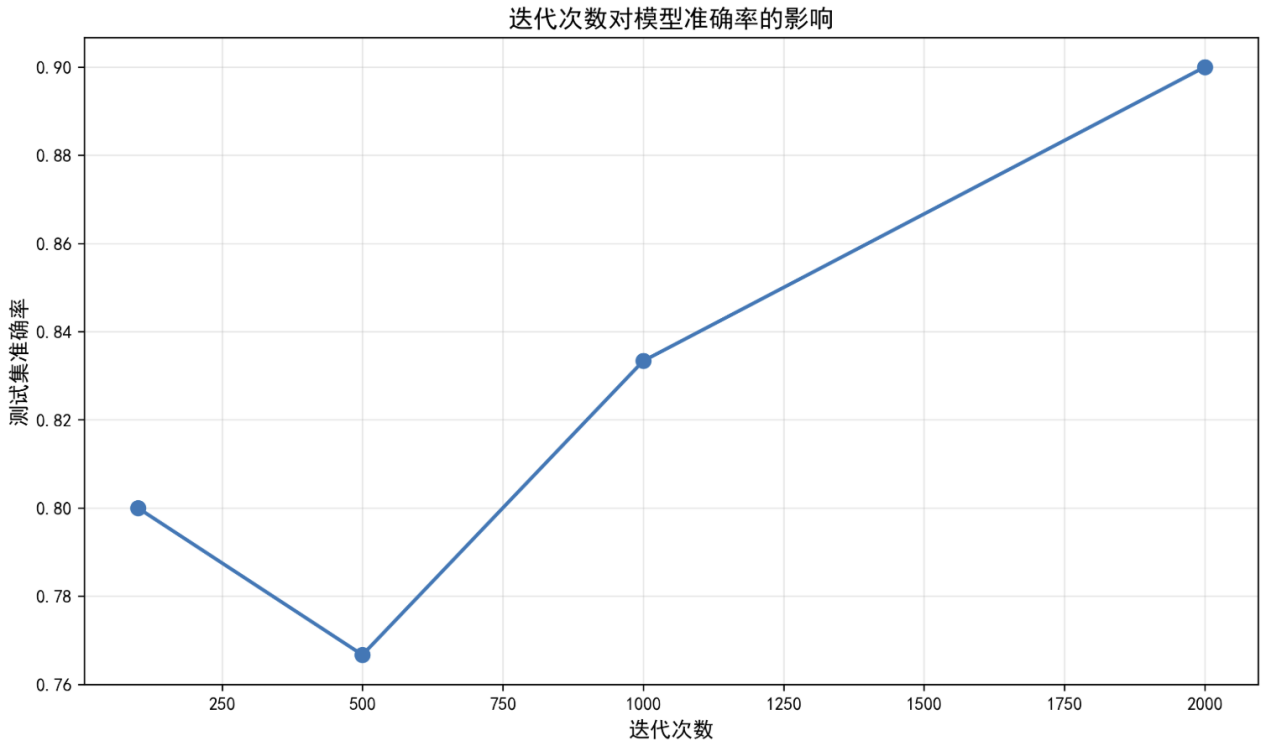


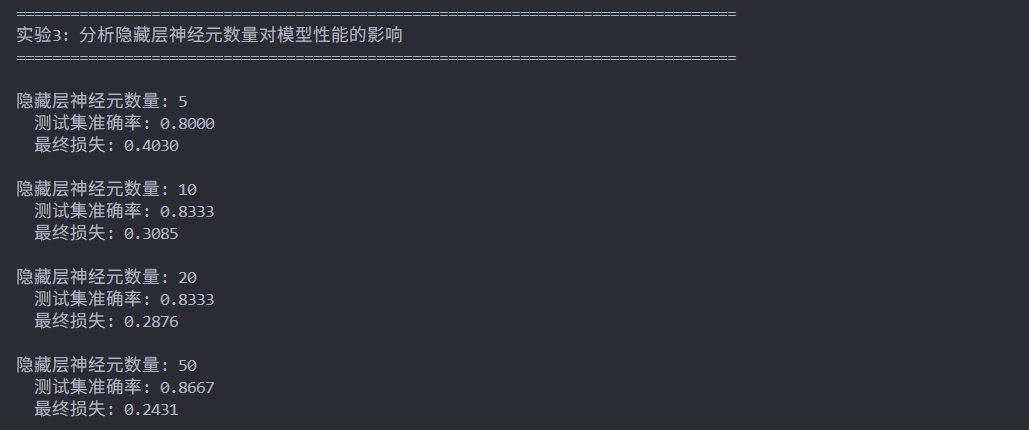


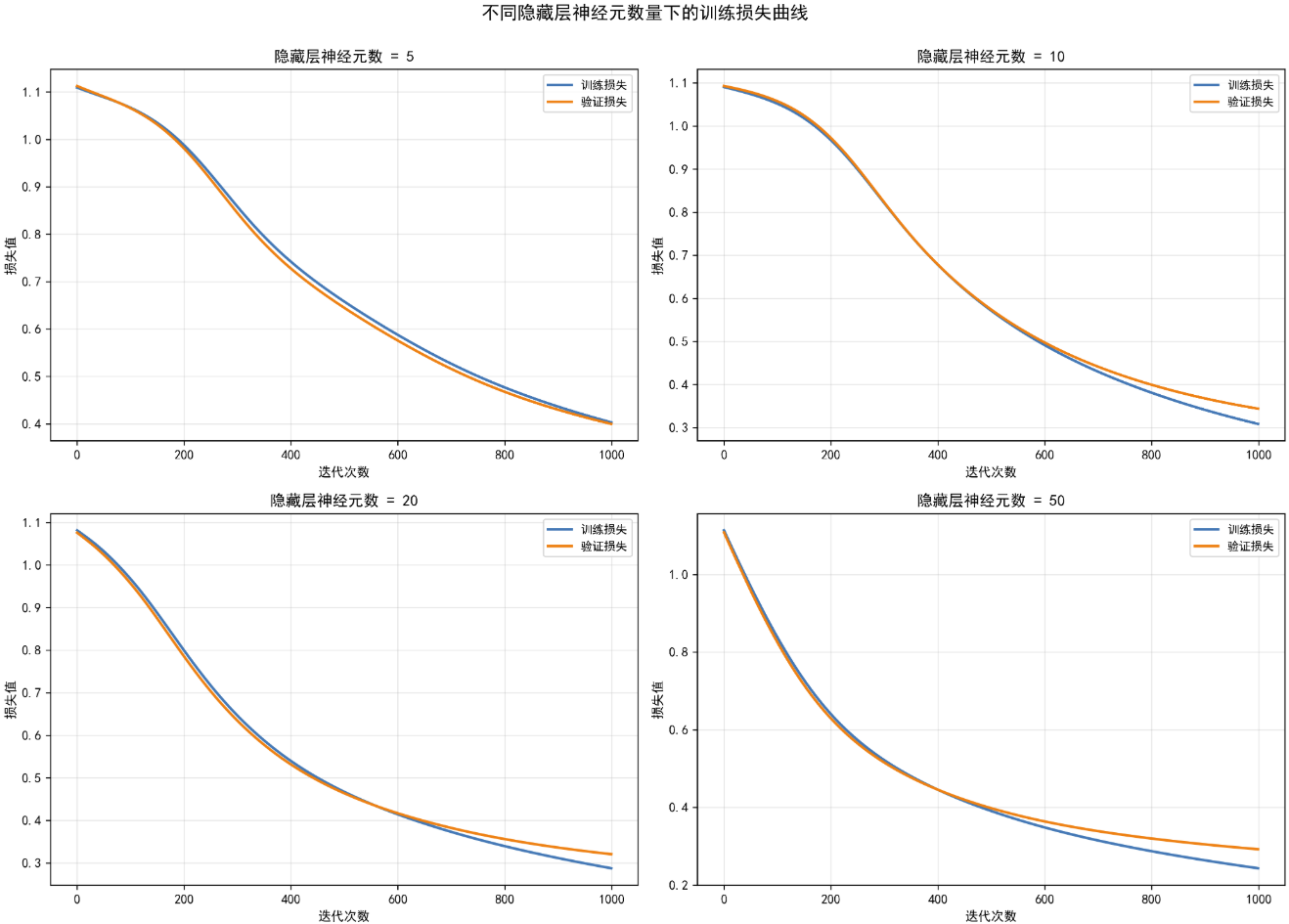


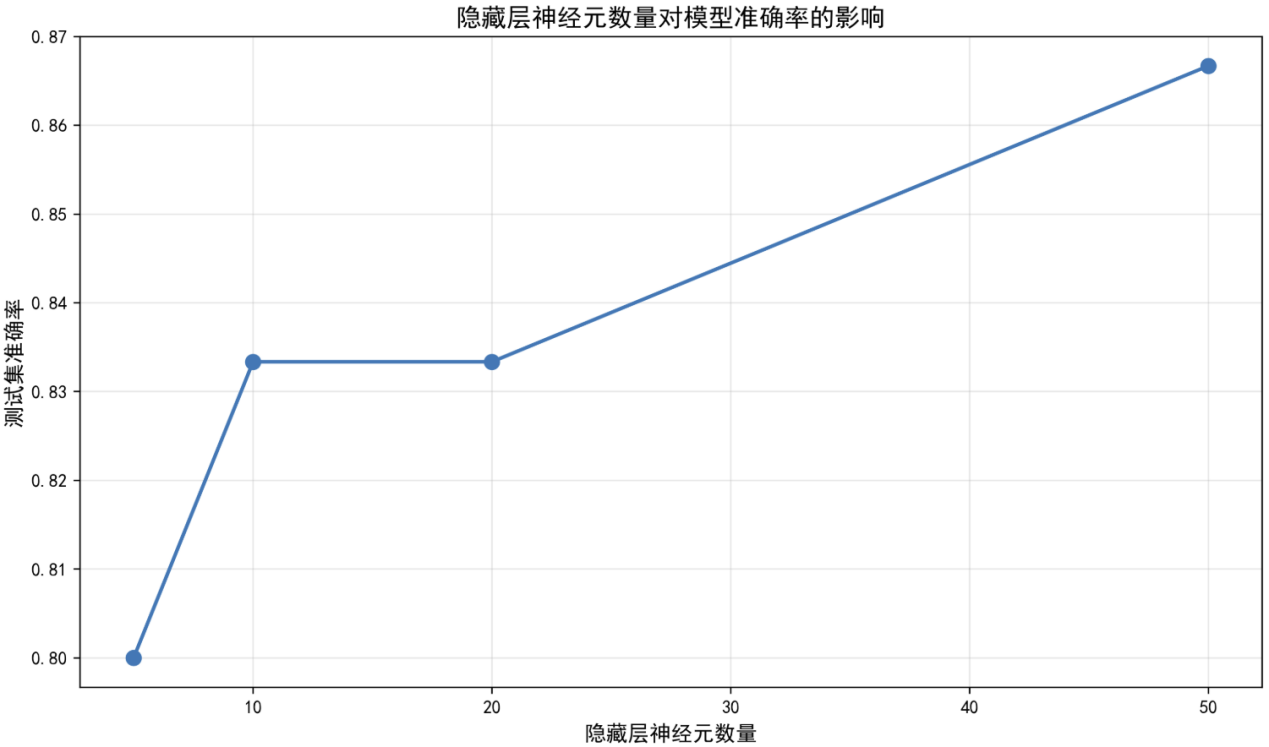


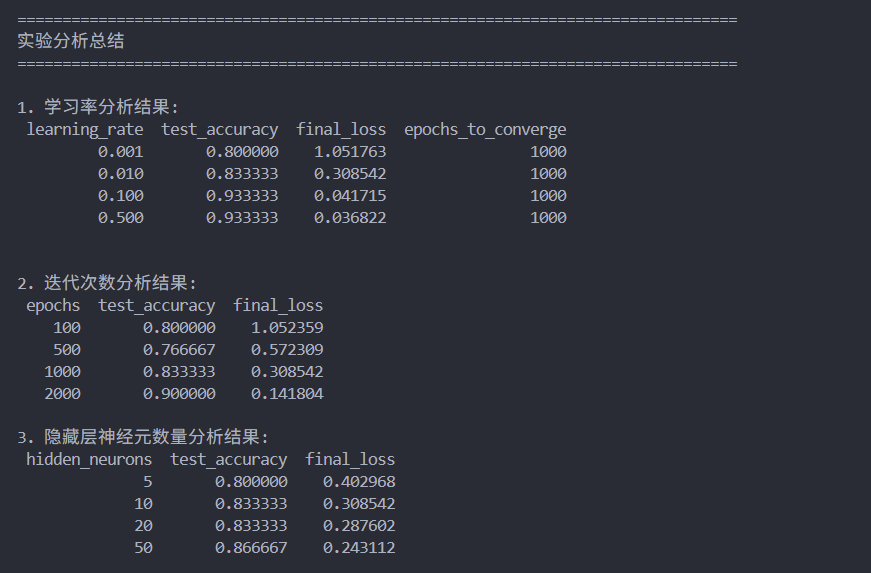






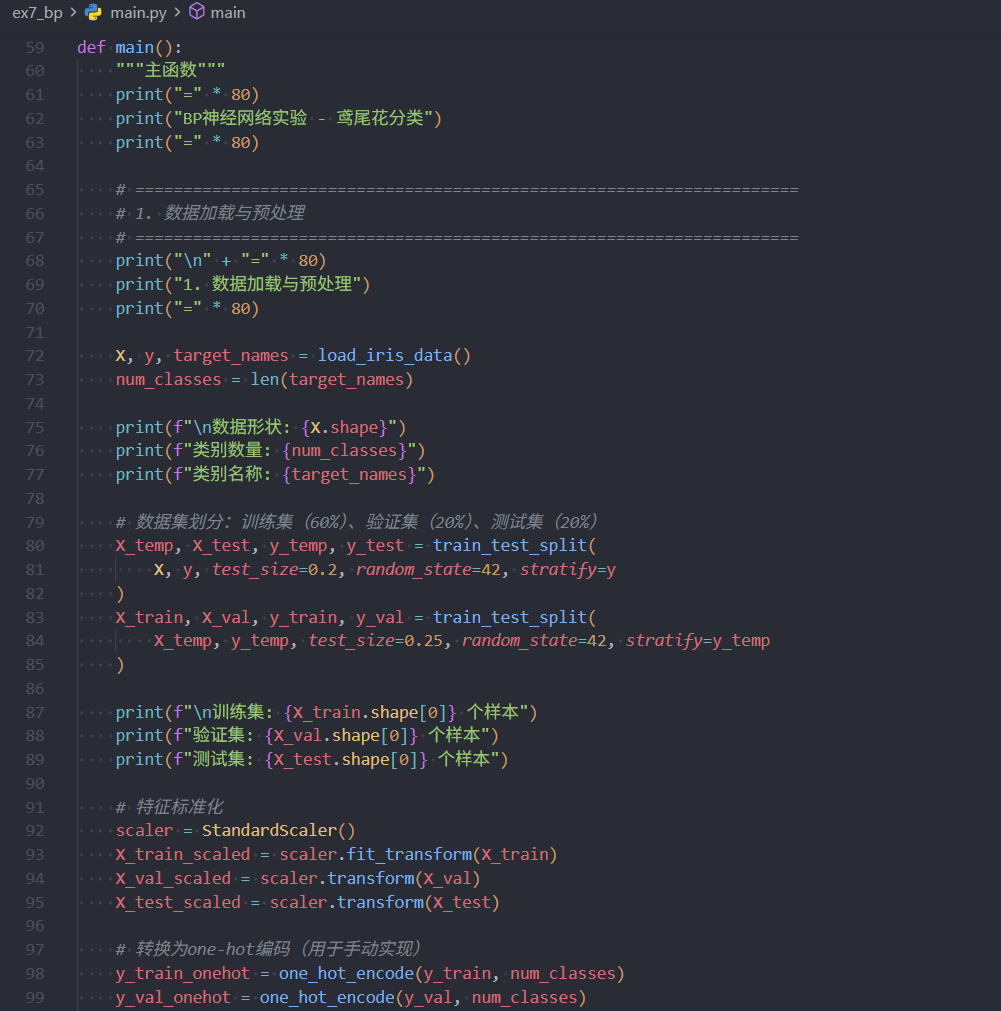






### 程序截图

数据预处理：

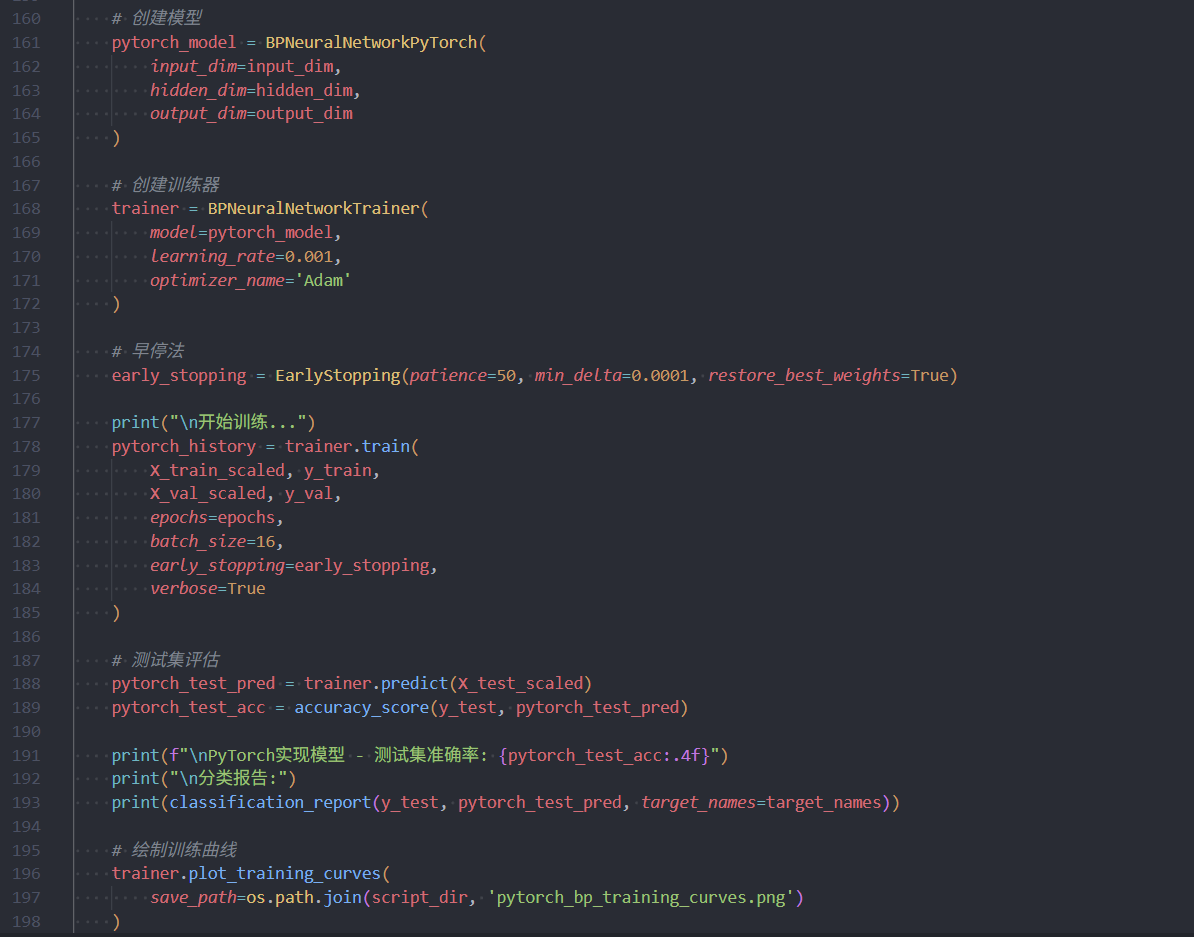


手动实现BP神经网络



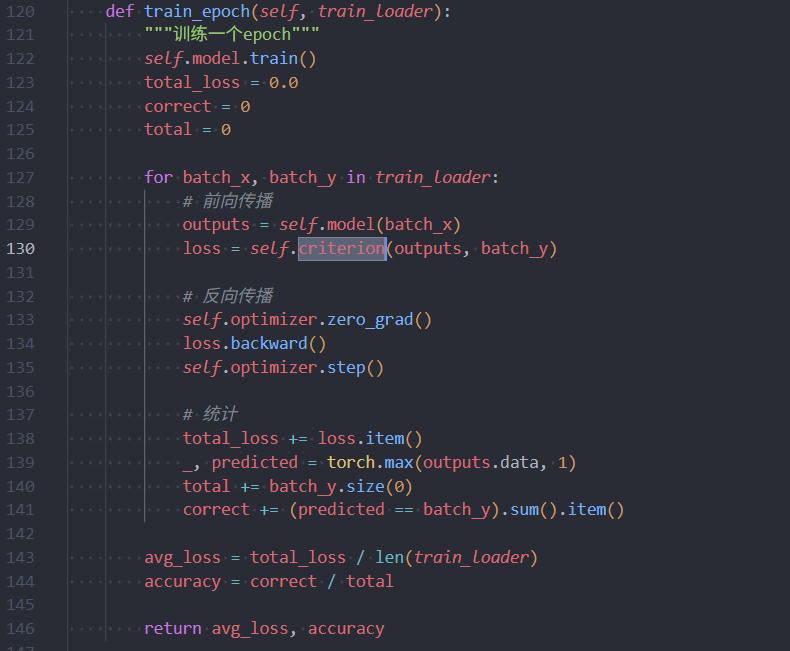
PyTorch框架实现BP神经网络

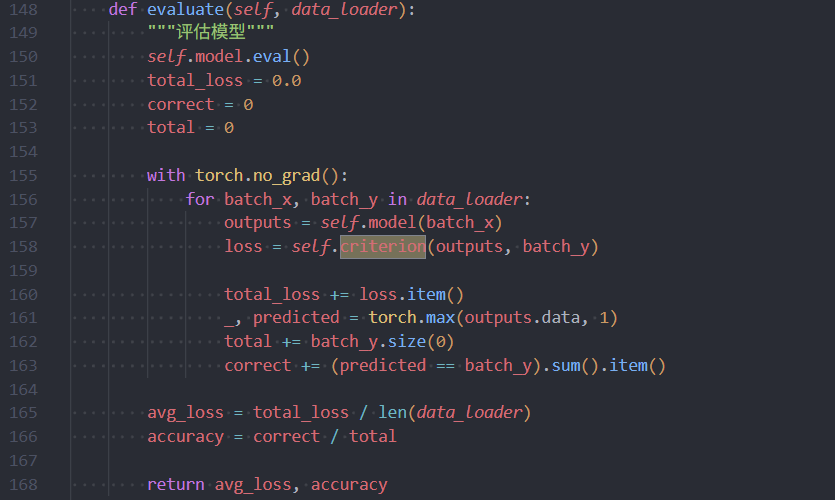
















### 实验名称：深度学习——卷积神经网络（CNN）图像分类

### 实验用时：4小时

### 实验内容

1. 卷积层：通过卷积核（滤波器）对输入图像进行卷积运算，提取图像的局部特征（如边缘、纹理），卷积运算具有局部连接和权值共享的特点，减少模型参数；​
2. 池化层（下采样）：对卷积层输出的特征图进行池化操作（如最大池化、平均池化），降低特征图维度，减少计算量，同时保留关键特征，提高模型的鲁棒性；​
3. 全连接层：将池化层输出的特征图扁平化后，通过全连接层映射到输出层，实现分类或回归任务；​
4. LeNet-5 模型：经典的 CNN 模型，适用于手写数字识别，结构包括：输入层（32×32 图像）→卷积层（6 个 5×5 卷积核）→池化层（2×2 最大池化）→卷积层（16 个 5×5 卷积核）→池化层（2×2 最大池化）→全连接层（120 个神经元）→全连接层（84 个神经元）→输出层（10 个神经元，对应 10 个数字）。

### 实验步骤

1. 数据集准备：使用 MNIST 手写数字数据集（60000 张训练图，10000 张测试图，每张图为 28×28 灰度图，共 10 个类别），通过框架自带接口加载数据集；​
2. 图像数据预处理：​

* 数据格式转换：将图像数据转换为框架支持的张量格式；​
* 归一化：将像素值从 [0,255] 转换为 [0,1]，加速模型训练；​
* 数据增强（可选）：如随机旋转、平移、翻转等，提高模型泛化能力；

1. 构建 CNN 模型（基于 LeNet-5 改进）：​

* 使用 PyTorch 定义模型结构：输入层（28×28×1）→卷积层（32 个 3×3 卷积核，激活函数 ReLU）→池化层（2×2 最大池化）→卷积层（64 个 3×3 卷积核，激活函数 ReLU）→池化层（2×2 最大池化）→扁平化层→全连接层（128 个神经元，激活函数 ReLU）→Dropout 层（ dropout rate=0.5，防止过拟合）→输出层（10 个神经元，激活函数 Softmax）；
* 查看模型结构：打印模型的层信息、参数数量，理解各层的作用；​

1. 模型训练与监控：​

* 配置训练参数：优化器选择 Adam（学习率 0.001），损失函数选择稀疏分类交叉熵（适用于整数标签），评估指标选择准确率；​
* 训练模型：设置 epochs=10、batch\_size=32，使用训练集训练模型，同时用测试集作为验证集监控模型性能，记录训练过程中的训练准确率、验证准确率、训练损失、验证损失；​
* 可视化训练过程：绘制训练准确率与验证准确率曲线、训练损失与验证损失曲线，分析模型是否存在过拟合（如验证准确率上升后下降，验证损失下降后上升）；​

1. 模型评估与预测：​

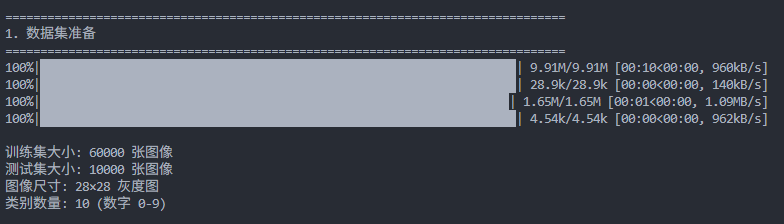
* 在测试集上评估模型性能，计算测试准确率，查看混淆矩阵，分析模型在不同类别数字上的分类效果；​
* 随机选取测试集中的几张图像，使用训练好的模型进行预测，对比预测结果与真实标签，直观观察模型预测效果；​

1. 模型优化（可选）：​

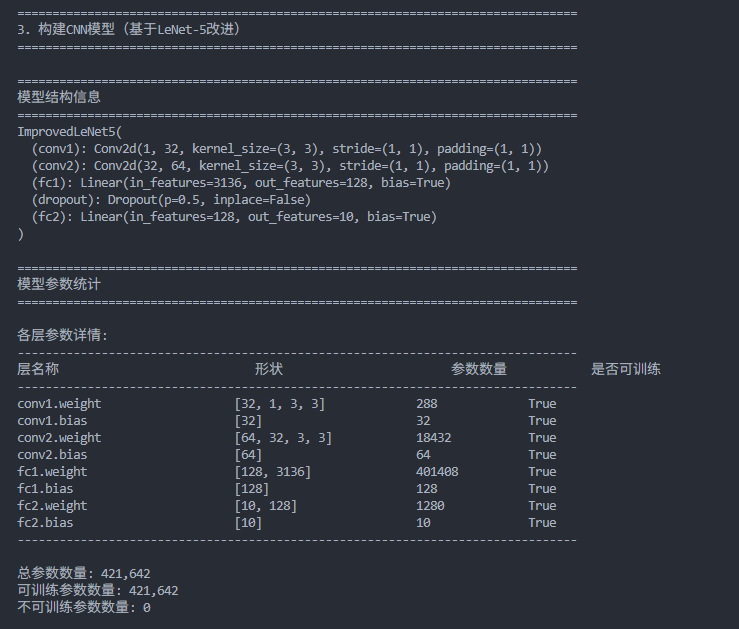
* 调整模型结构（如增加卷积层数量、改变卷积核大小、调整全连接层神经元数量）；​
* 调整训练参数（如学习率、batch\_size、epochs、dropout rate）；​
* 应用数据增强技术，对比优化前后模型的性能变化；​

实验总结：整理实验过程中的关键代码、训练曲线、评估结果，分析 CNN 模型在图像分类任务中的优势，总结模型训练与优化的关键技巧。

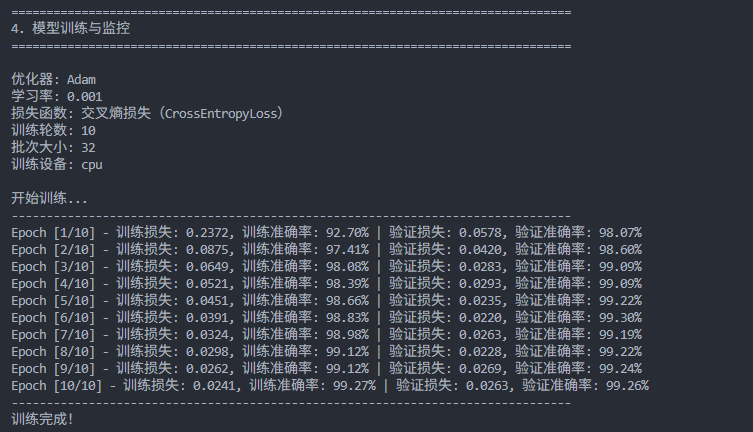
### 实验分析

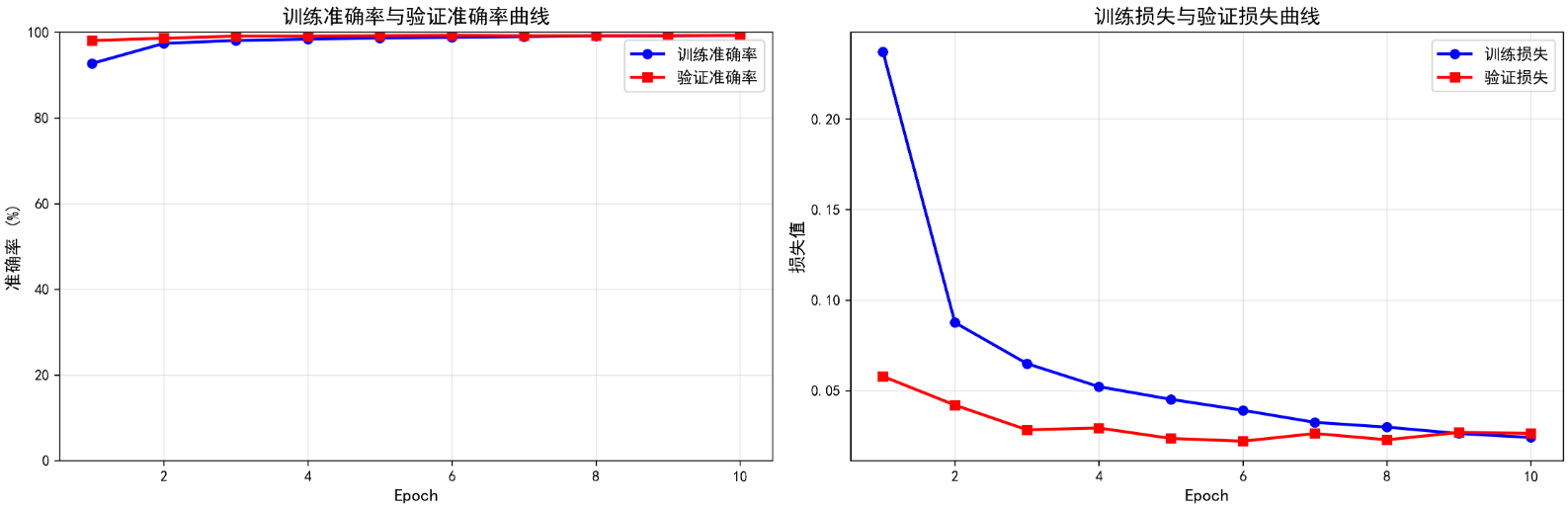


构建CNN模型，打印模型信息如下：

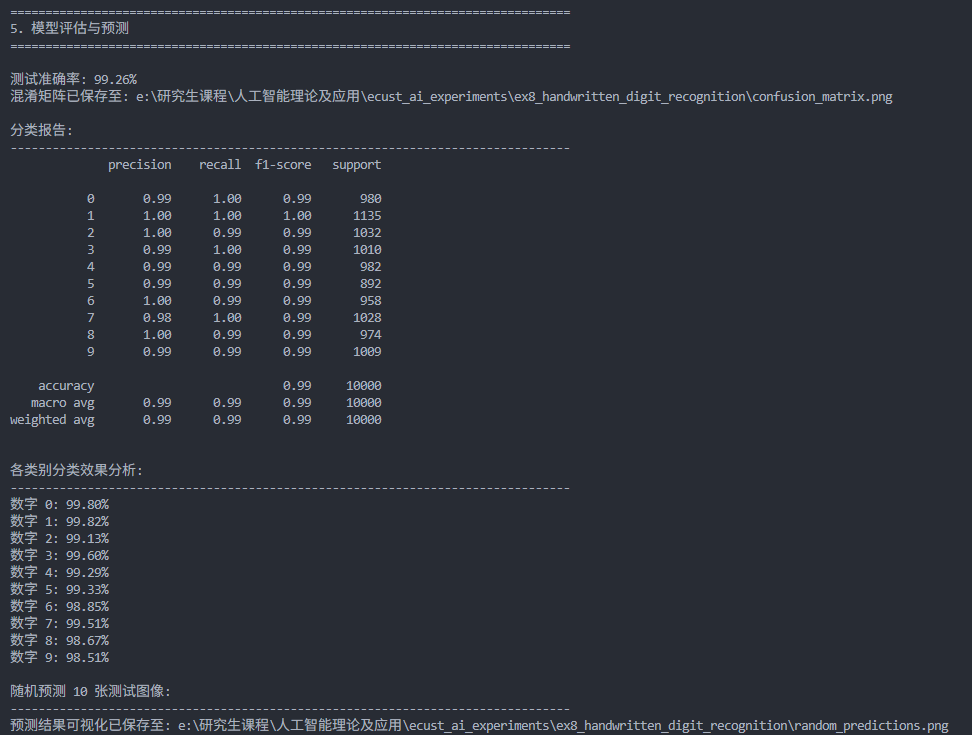


模型训练与监控

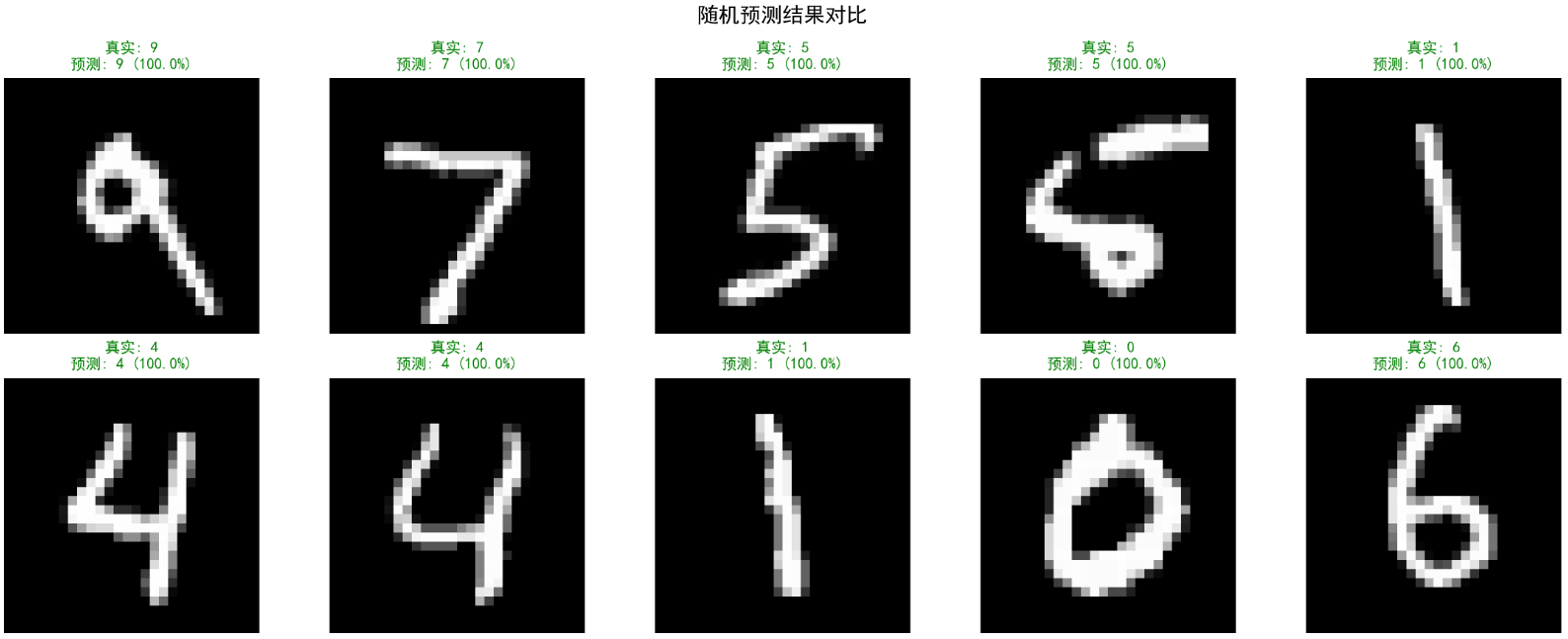


训练效果，准确率达99.26%。

模型评估与预测：

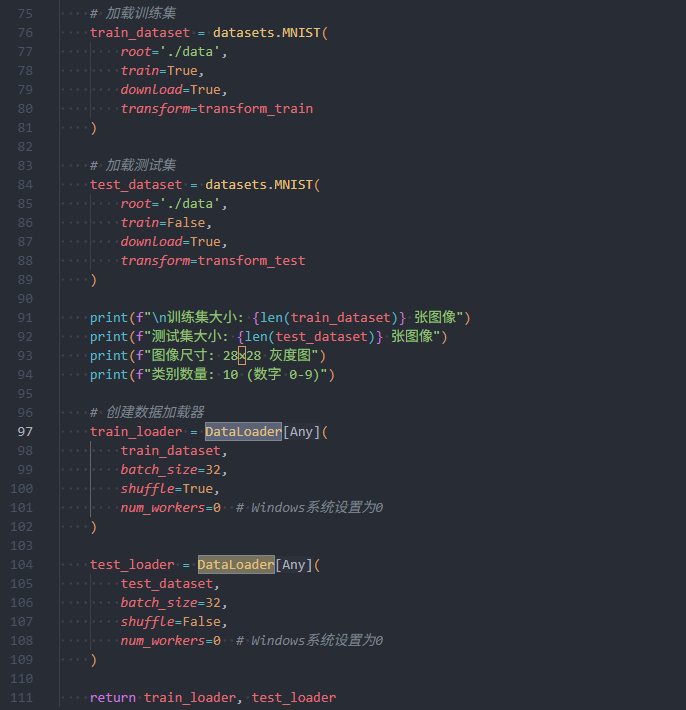




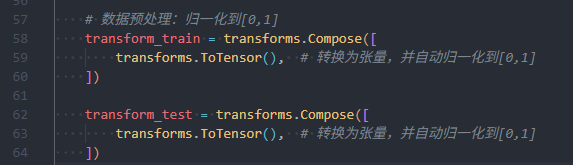


### 程序截图

数据集准备：



图像数据预处理：



构建CNN模型：





模型训练







模型预测





### 实验名称：回归问题建模与求解——线性回归与SVM

### 实验用时：3小时

### 实验内容

1. 线性回归模型：

一元线性回归模型为：*y*=*wx*+*b,*（w 为权重，b 为偏置）

多元线性回归模型为：

通过最小化平方损失函数 ​求解模型参数；​

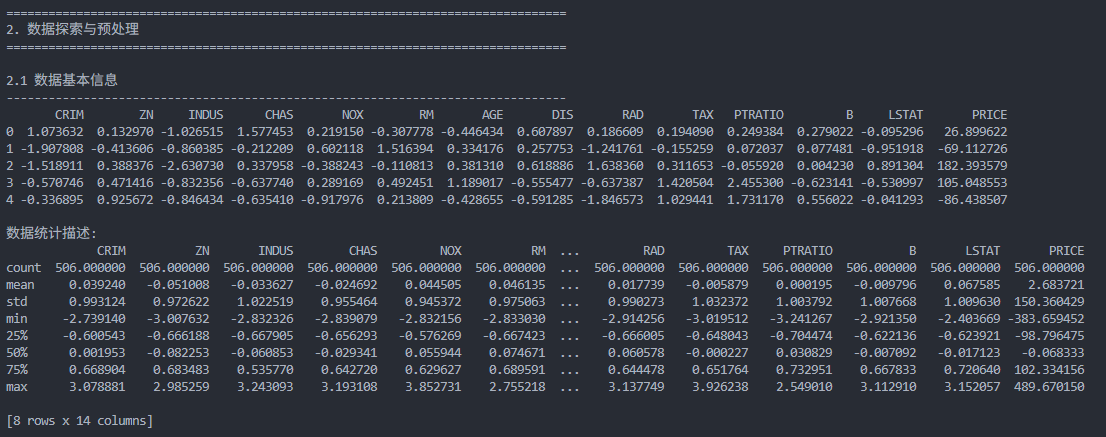
1. 模型评估指标：均方误差（MSE）为 ​，平均绝对误差（MAE）为 ，决定系数 R² 衡量模型对数据的拟合程度，取值范围为 [0,1]，越接近 1 拟合效果越好；​
2. 正则化方法：L1 正则化（Lasso 回归）在损失函数中加入权重的 L1 范数 ​，可实现特征选择；L2 正则化（Ridge 回归）加入权重的 L2 范数 ​，可降低权重绝对值，缓解过拟合。

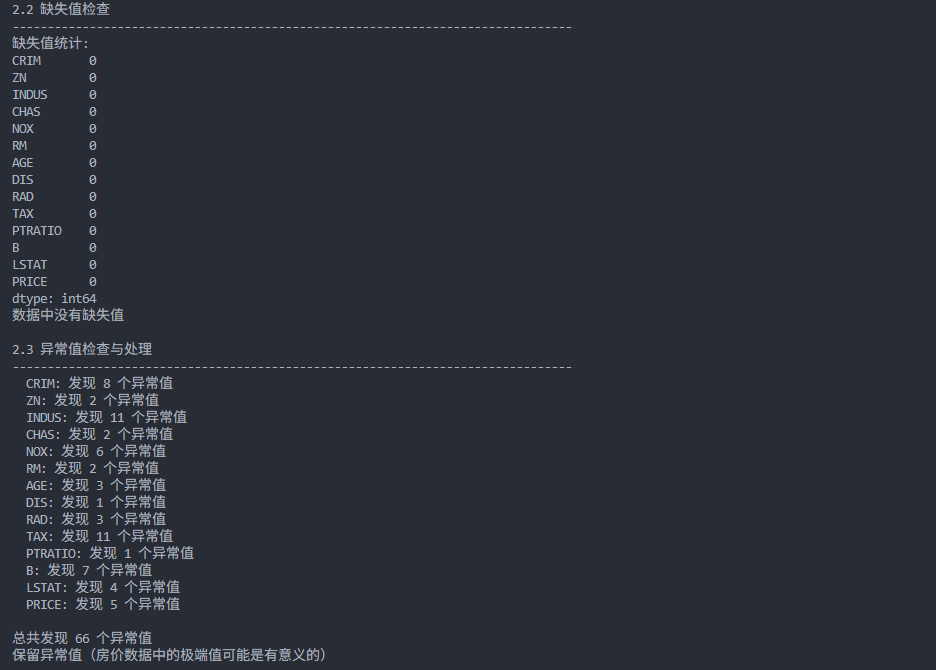
### 实验步骤

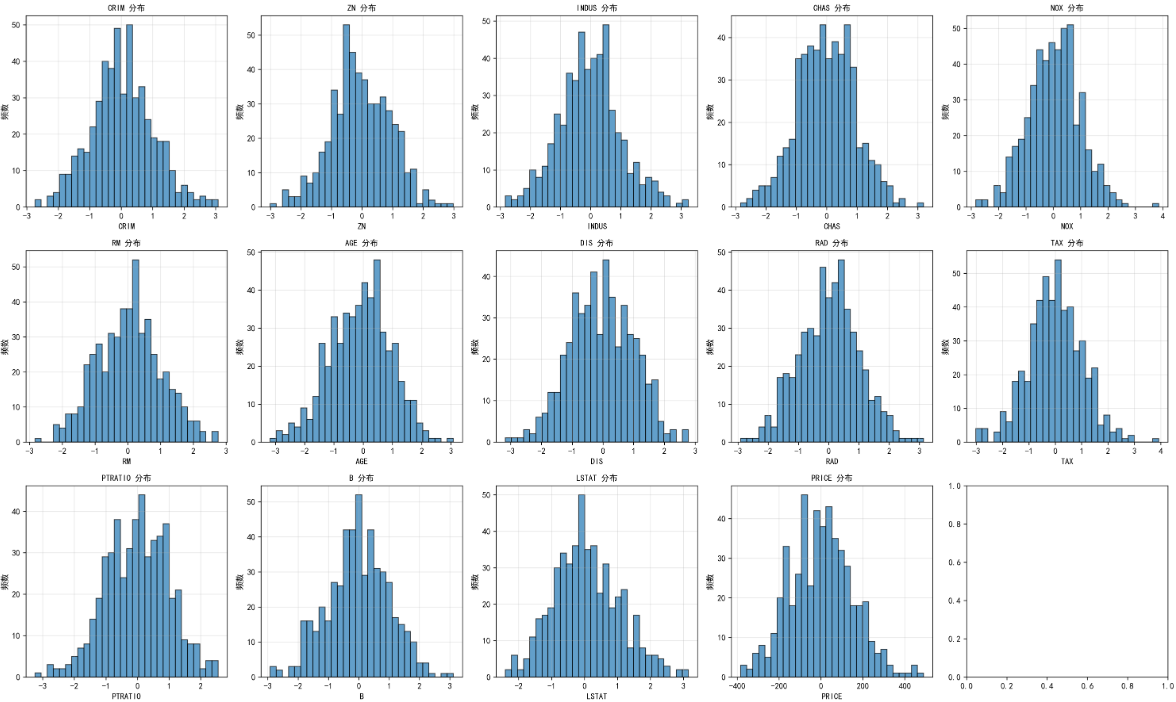
1. 数据准备：使用 Scikit-learn 库的波士顿房价数据集；​
2. 数据探索与预处理：查看数据分布、特征间相关性，处理缺失值、异常值，划分训练集和测试集，对特征进行标准化；​
3. 多元线性回归实现：使用所有特征构建多元线性回归模型，训练模型，分析各特征的权重系数；​
4. 多元非线性回归实现：使用SVM构建回归模型，训练模型，与原始数据对比；​
5. 模型评估：计算训练集和测试集的 MSE、MAE、R²，分析模型拟合效果，判断是否存在过拟合或欠拟合；​
6. 正则化应用：分别构建 Lasso 回归和 Ridge 回归模型，调整正则化参数 λ，对比分析正则化前后模型的性能变化，观察 Lasso 回归对特征的选择效果；​

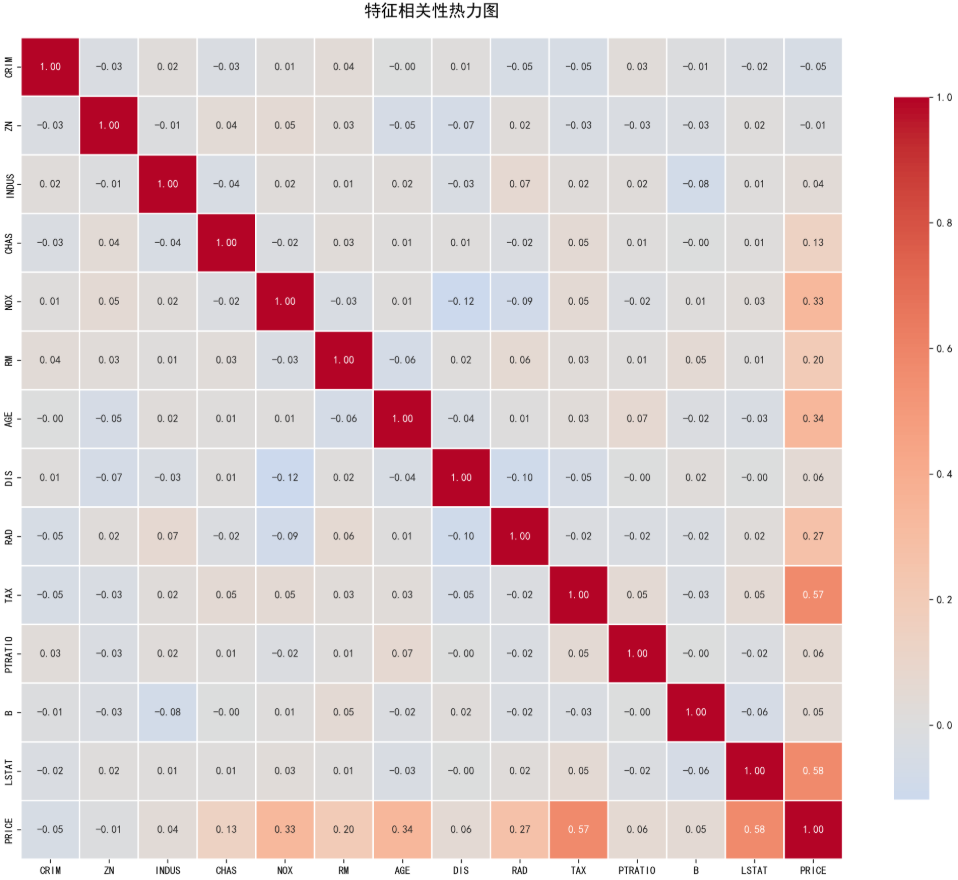
### 实验分析

数据处理



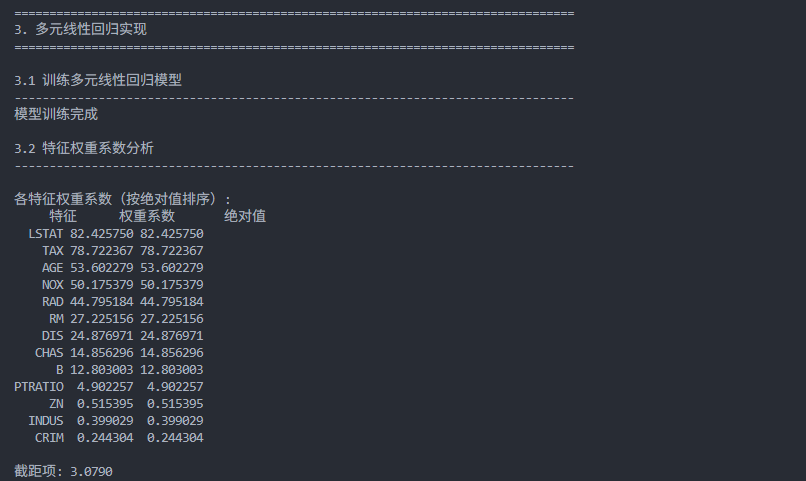


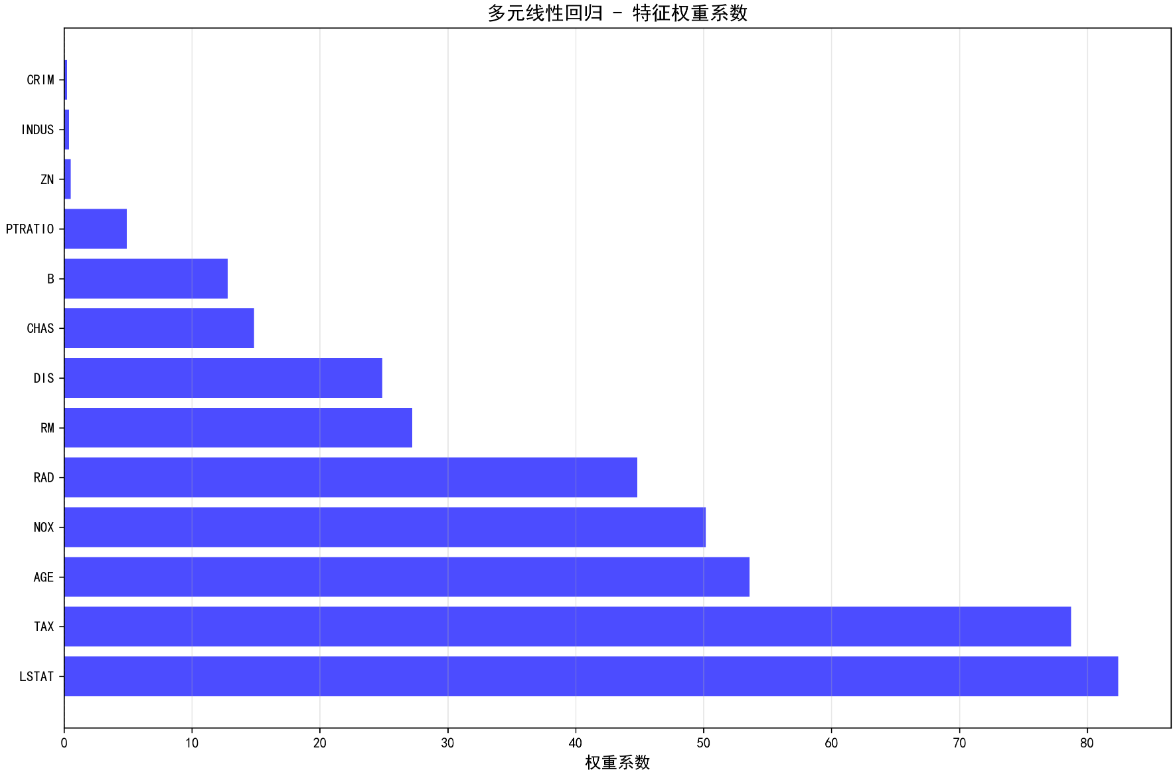




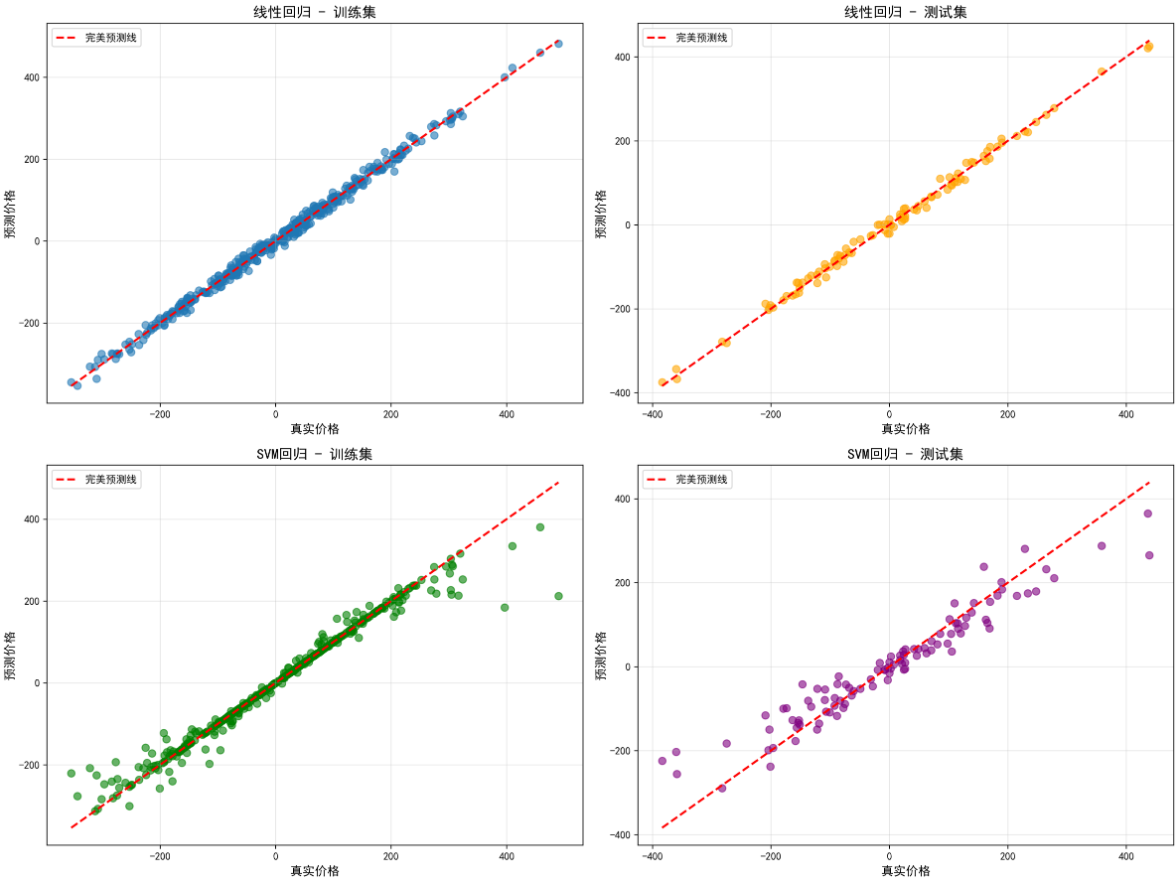
训练集：404个样本，测试集：102个样本

多元线性回归实现：

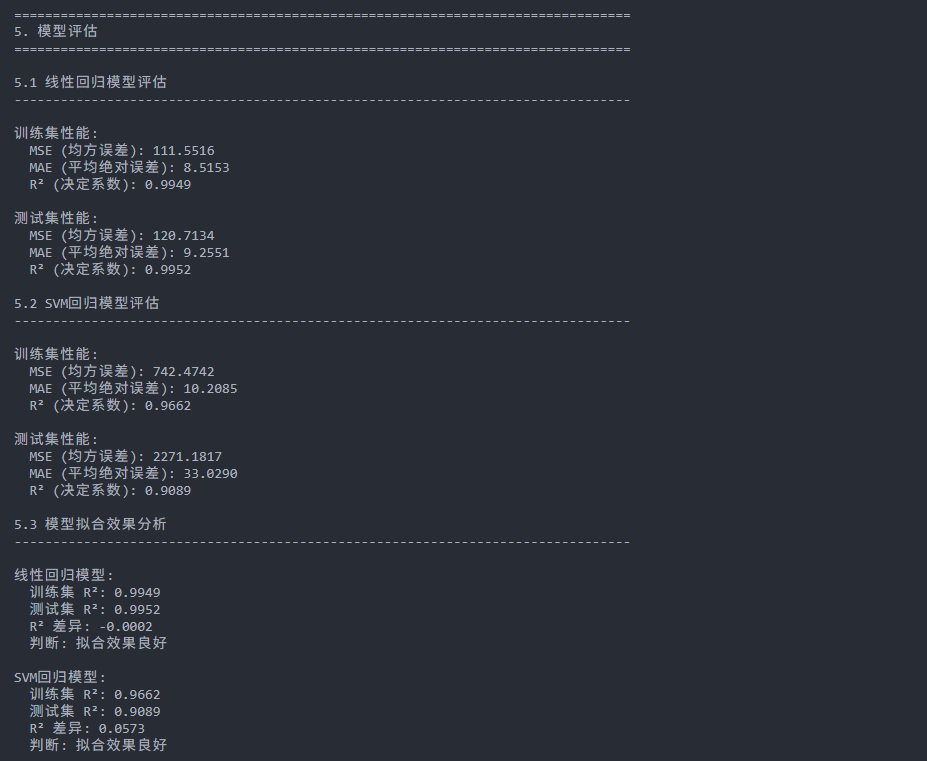


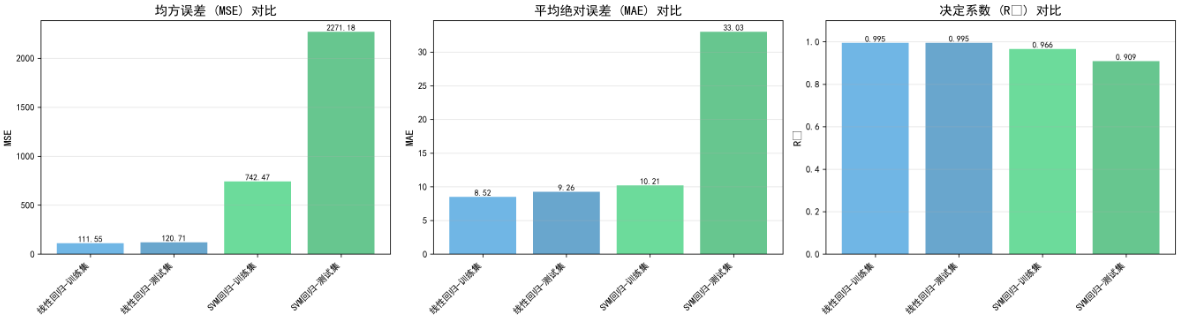


多元非线性回归：

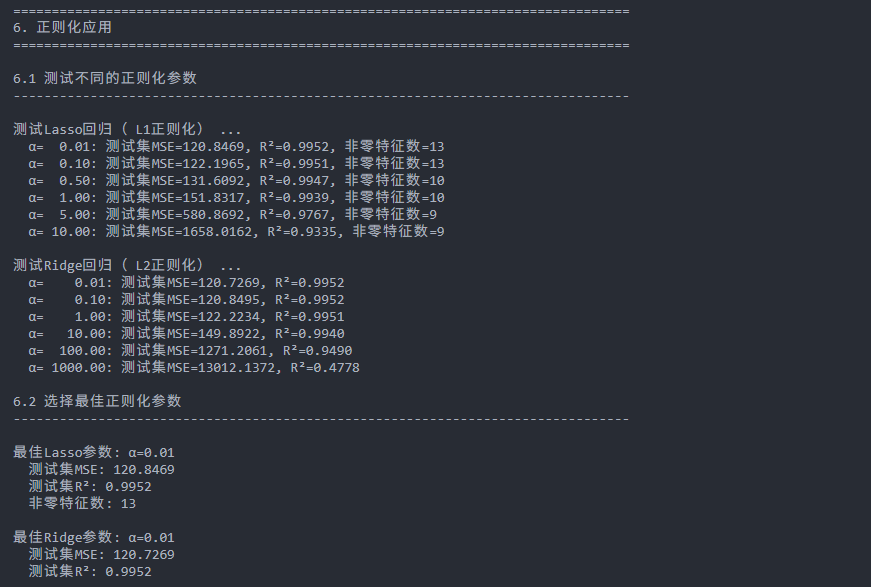


模型评估：

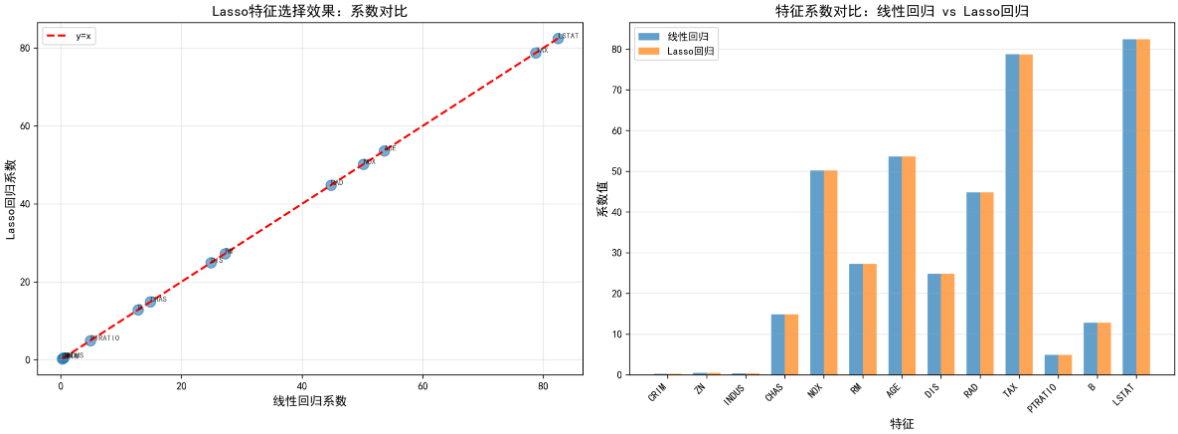


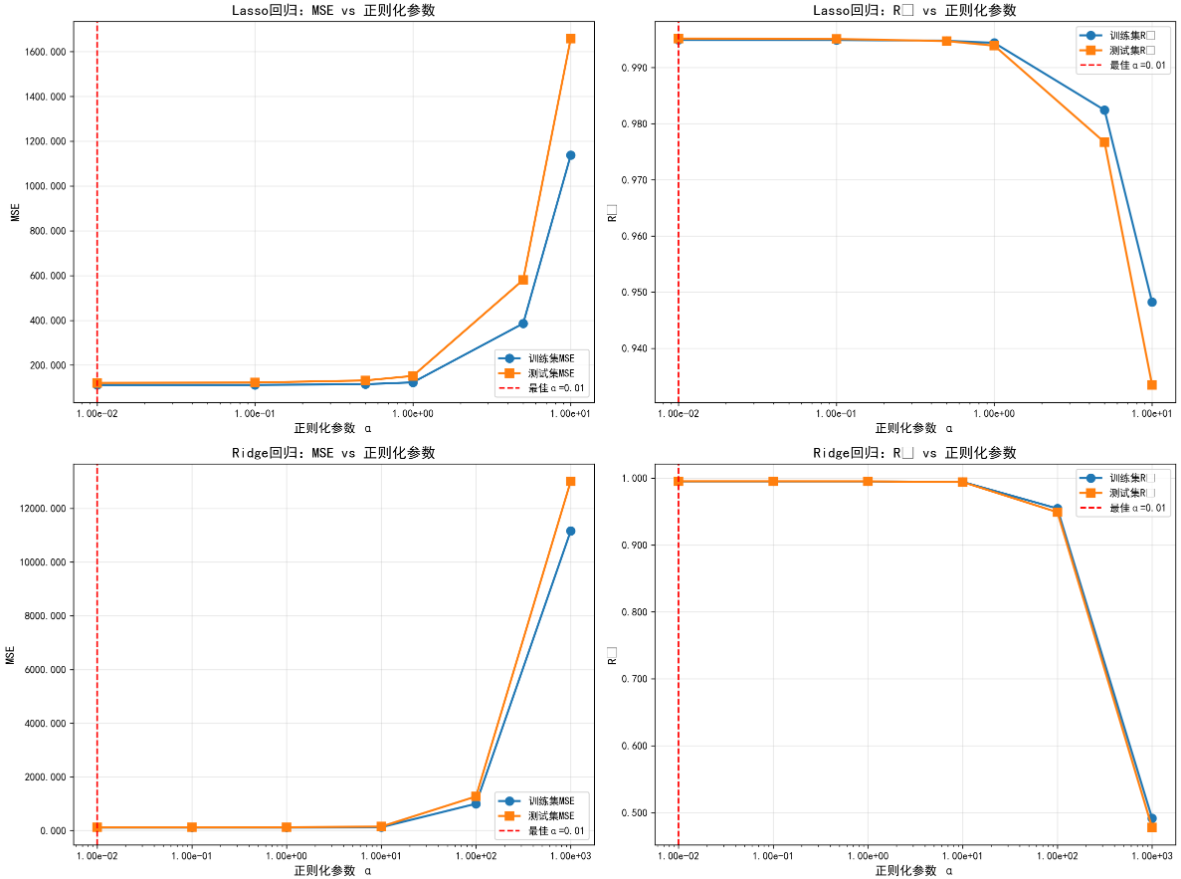


正则化应用：





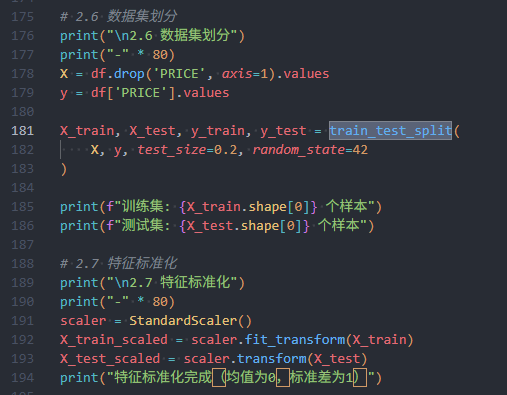






### 程序截图

数据处理——数据集划分、特征标准化



多元线性回归实现：



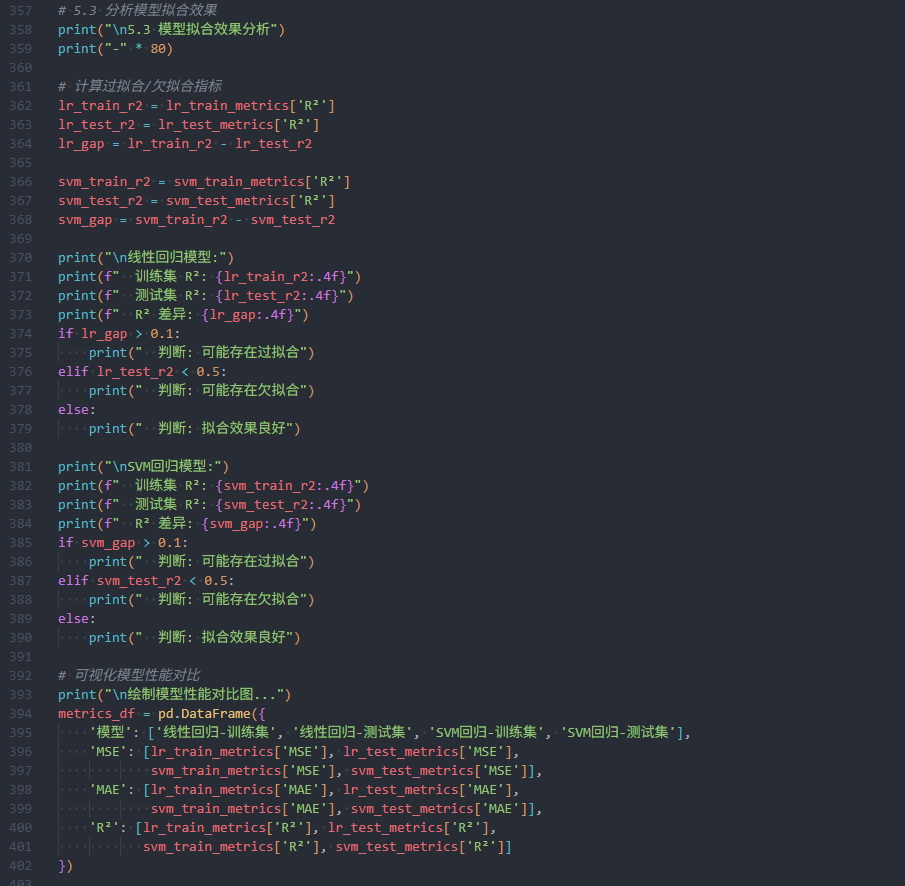
多元非线性回归实现：

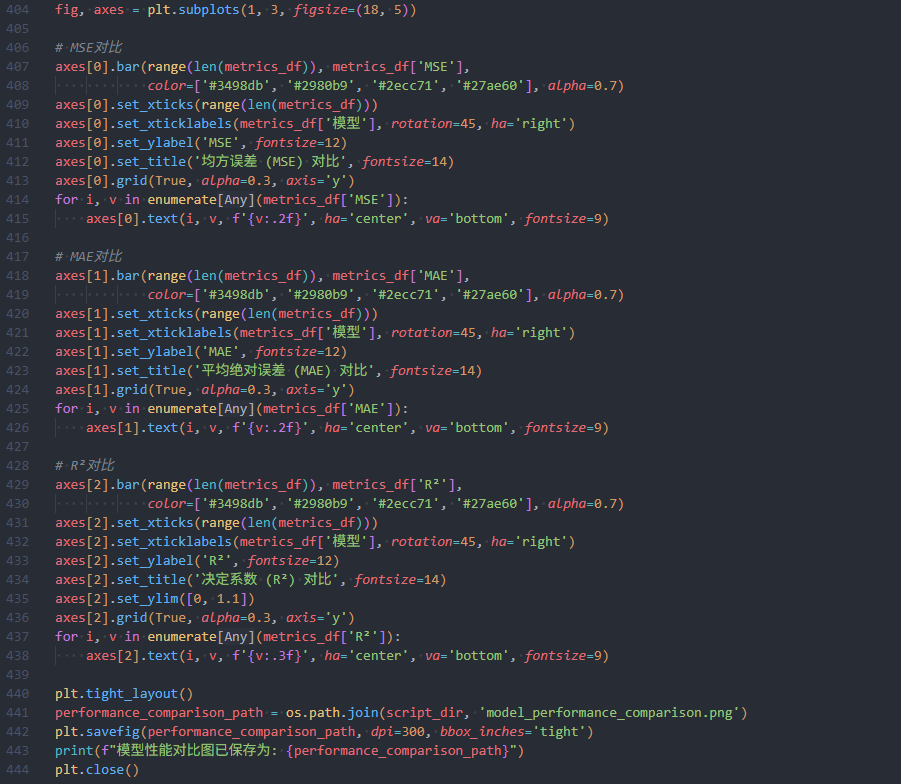




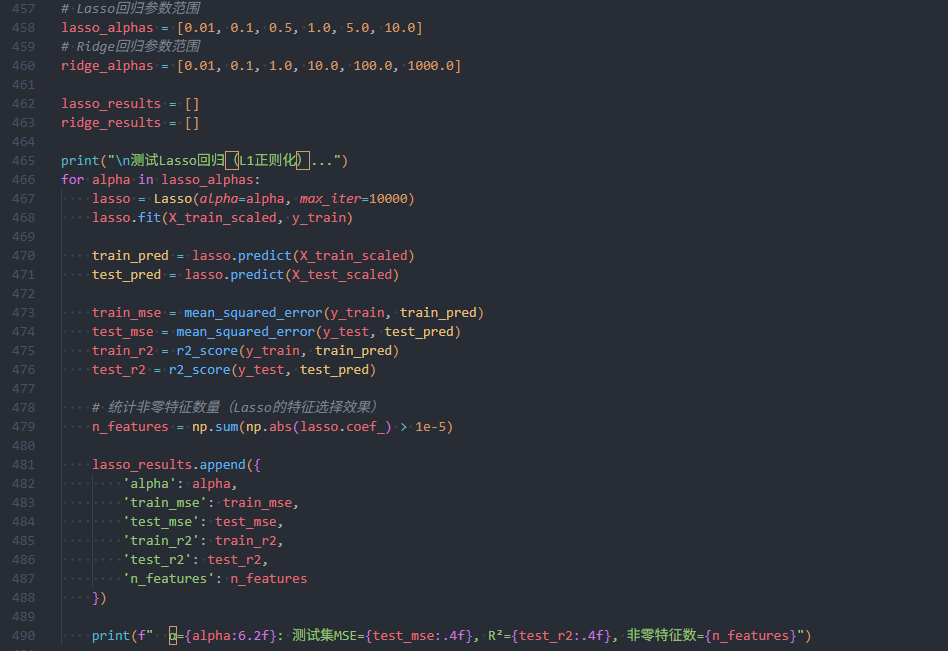
模型评估：

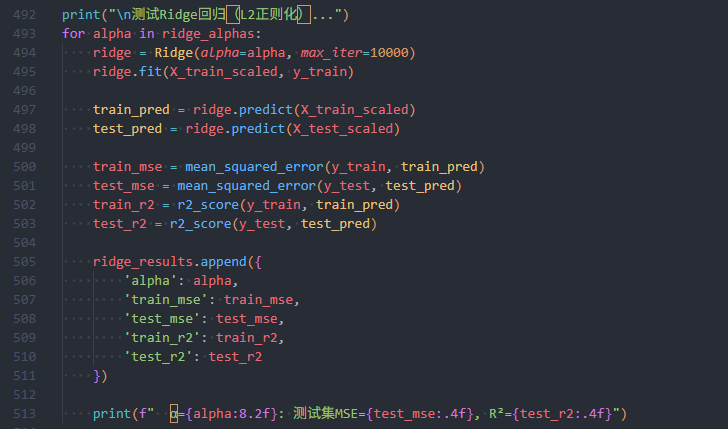






正则化应用：







# 专业实验

### 实验名称：

### 实验用时：

### 实验内容

### 实验步骤

### 实验分析

### 程序截图

### 实验名称：

### 实验用时：

### 实验内容

### 实验步骤

### 实验分析

### 程序截图