

题目： 深度神经网络实验 1

班级： 智能2204班

姓名： 朱思颖

学号： 202208040407

1

信息科学与工程学院

目录

[一、实验概述 3](#_Toc3985)

[（一）实验要求 3](#_Toc31795)

[（二） 实验设计 3](#_Toc31424)

[（三） 实验环境 3](#_Toc22508)

[二、实验原理 4](#_Toc859)

[（一）深度神经网络基础原理 4](#_Toc4073)

[1、前向传播与反向传播 4](#_Toc30371)

[2、正则化机制 4](#_Toc12922)

[（二）超参数调优原理 5](#_Toc12117)

[1、网格搜索vs 随机搜索 5](#_Toc26195)

[2、SMOTE算法 5](#_Toc15512)

[（三）实验设计原理 5](#_Toc31805)

[1. 数据划分策略 5](#_Toc6503)

[2. 早停机制设计 5](#_Toc842)

[3. 评估指标选择 6](#_Toc16488)

[二、 DNN的起源和发展 6](#_Toc7145)

[三、实验过程 6](#_Toc1861)

[（一） 了解数据集并进行处理 6](#_Toc26692)

[（二） 基础实现 8](#_Toc17923)

[（三） 改进版本 9](#_Toc11460)

[（四）超参调优版本 11](#_Toc12785)

[（五）数据依赖性分析 13](#_Toc2503)

[1、数据量依赖性分析 13](#_Toc20768)

[2、特征扰动分析 13](#_Toc16310)

[3、 标签噪声分析 14](#_Toc21122)

[四、实验结果 15](#_Toc14628)

[（一） 所实现的三个版本的对比分析 15](#_Toc14372)

[1、 技术实现对比 15](#_Toc12960)

[2、 性能表现对比 15](#_Toc27683)

[3、 实用价值对比 16](#_Toc22113)

[五、 附录说明 16](#_Toc24133)

[1、“code”文件夹中： 17](#_Toc4591)

[2、“整理后的模块化代码”文件夹中： 17](#_Toc30558)

# 一、实验概述

## （一）实验要求

## 实验设计

## 实验环境

**Google Colaboratory ：**

操作系统: Linux （Ubuntu 22.04.4 LTS）

操作\_DYNAMIC Sun Mar 30 16:01:29 UTC 2025

平台信息: Linux-6.1.123+-x86\_64-with-glibc2.35

**Python :**3.11.12

**核心依赖：**已整理至requirements.txt

# 二、实验原理

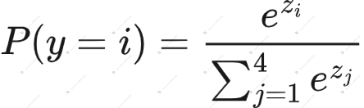
## （一）深度神经网络基础原理

### 1、前向传播与反向传播

前向传播：  
输入数据逐层加权求和并通过ReLU激活函数：

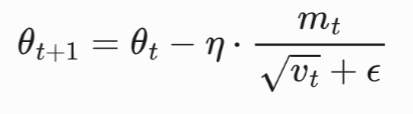
未命名文件

输出层使用Softmax计算类别概率：



反向传播：

通过交叉熵损失 计算梯度，Adam优化器更新参数：

### 2、正则化机制

|  |  |  |
| --- | --- | --- |
| **方法** | **数学表达式** | **作用** |
| **L2正则化** |  | 抑制权重过大，防止过拟合 |
| **Dropout** |  | 随机失活神经元，增强鲁棒性 |

## （二）超参数调优原理

### 1、网格搜索vs 随机搜索

|  |  |  |
| --- | --- | --- |
| **方法** | **原理** | **本实验应用** |
| **Keras Tuner** | 在超空间内均匀采样，评估所有组合 | **搜索units1∈[64,256]等连续参数** |
| **模拟退火** | |  | | --- | | 受冶金学启发，以概率接受劣解（*T*为温度参数） | | **优化l2\_rate等敏感参数** |

### 2、SMOTE算法

**SMOTE**是一种过采样技术，能生成少量样本而避免数据分布不一致的问题。

**核心步骤：**

对少数类样本，找到k近邻；随机选择邻域样本，生成新样本：

**意义**：在特征空间中对少数类进行凸组合插值。

## （三）实验设计原理

### 1. 数据划分策略

分层抽样：  
保持训练/测试集的类别分布与原始数据一致，避免偏差。

### 2. 早停机制设计

停止条件：  
当验证损失连续10个epoch未下降（patience=10）时终止训练。

理论依据：  
监控验证损失可检测过拟合点，早停等价于L2正则的隐式约束。

### 3. 评估指标选择

|  |  |  |
| --- | --- | --- |
| **指标** | **公式** | **优势** |
| **宏平均F1** |  | 平等对待所有类别 |
| **AUC-ROC** |  | 不受分类阈值影响 |

# DNN的起源和发展

1、起源：DNN源于1943年McCulloch-Pitts神经元模型，1986年反向传播算法（Backpropagation）的提出是重要里程碑。

2、突破：2010年后因算力提升、大数据积累和ReLU激活函数的应用而爆发式发展。

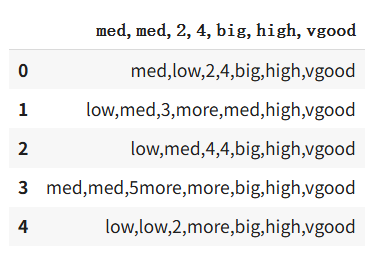
3、发展：经历了第一次寒冬、BP重提、硬件加速，一直发展至今，逐步集成进产业，仍有发展的潜力。

# **三、**实验过程

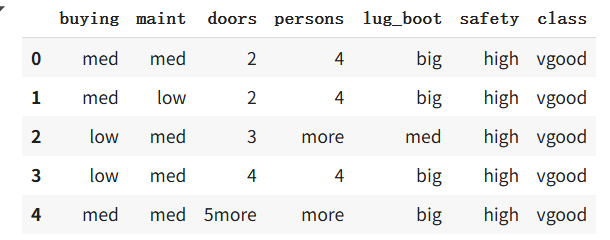
## 了解数据集并进行处理

1. **数据加载和描述性统计**

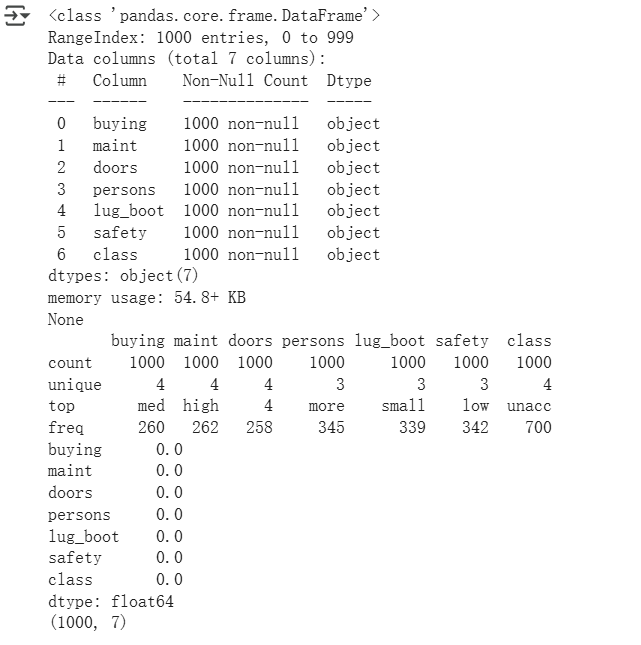
读取文件内容，解析为结构化数据，看到标头是数据的第一行



加上正确的标头重新加载数据：



查看基本信息：数据的基本结构信息和描述性统计



每类的唯一值数量

每类的最高频值

最高频值的出现次数

无缺失值，当前特征均为object型

7列（6个特征+1个目标变量）

总样本数

**2、目标变量（class）的分布情况以及不同类别在各个特征上的分布情况**

·Class中unacc占比最高——类别不平衡

红色（vgood）代表非常好的类别

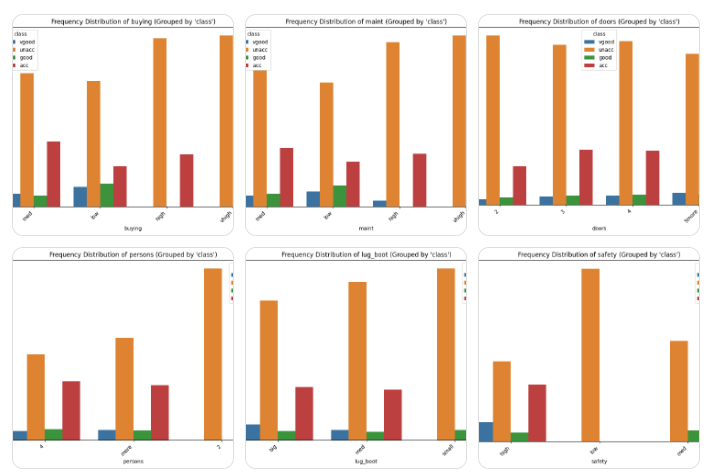
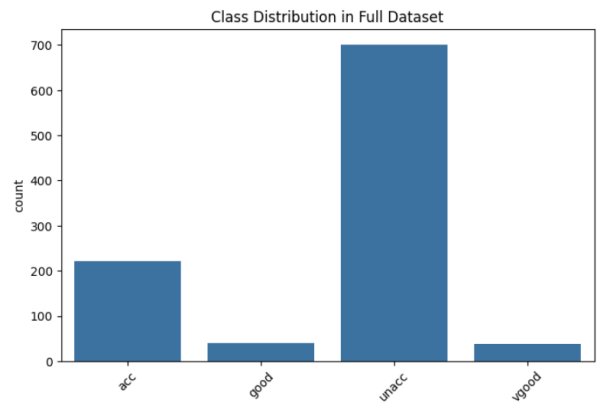
橙色（acc）代表可接受的类别

蓝色（unacc）代表不可接受的类别

绿色（good）代表好的类别

·“unacc”类别在所有特征中都占据主导地位

·“vgood”和“good”类别的数量相对较少。



总结：

·类别分布：

·Class中unacc占比最高，验证了类别不平衡

· Doors和person的众数为4，lug\_boot和safety的众数为med。

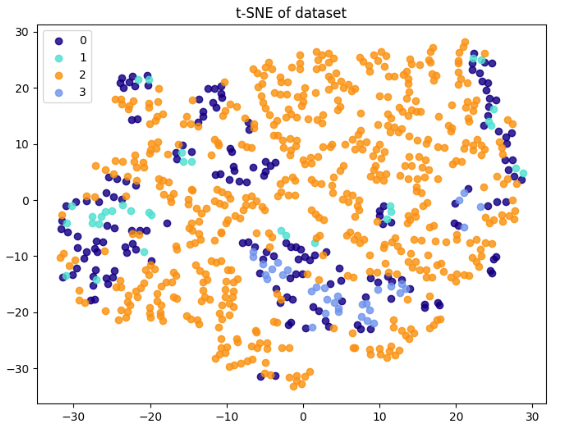
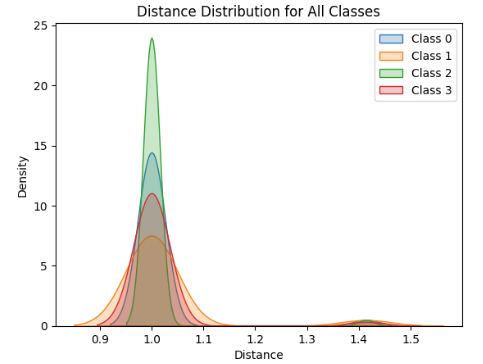
·唯一值：

·buying/maint有4个等级（vhigh/high/med/low），需有序编码。

·缺失值分析：数据完整，无需处理缺失值。

1. **特征距离分析**

使用核密度估计（KDE）来展示每个类别的距离分布，使用t-SNE来捕捉数据的复杂结构和聚类。



从左图我们可以得到的信息：

1、集中趋势：样本在特征空间中相对集中。

2、类别间相似性：各个类别的分布相互重叠，说明不同类别的样本在特征空间中可能彼此接近。

3、类别内相似性：都有尖峰，类别内部的样本相似度较高。

从右图我们可以得到的信息：

类别2有部分与类别0重叠，类别2错分的部分，很大可能会错分成类别0。

## 基础实现

**目标**：验证DNN在Car数据集上的可行性。

**数据预处理**：

·使用LabelEncoder对所有分类特征进行数值编码

·简单划分训练集/测试集（8:2）

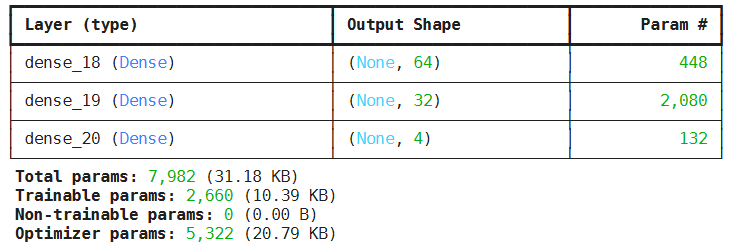
·为每个类别分配不同的权重，调整模型对不同类别的关注程度，从而缓解类别不平衡问题。

**模型构建：**

·2层全连接网络（64-32神经元）

·无正则化措施

·激活函数：softmax

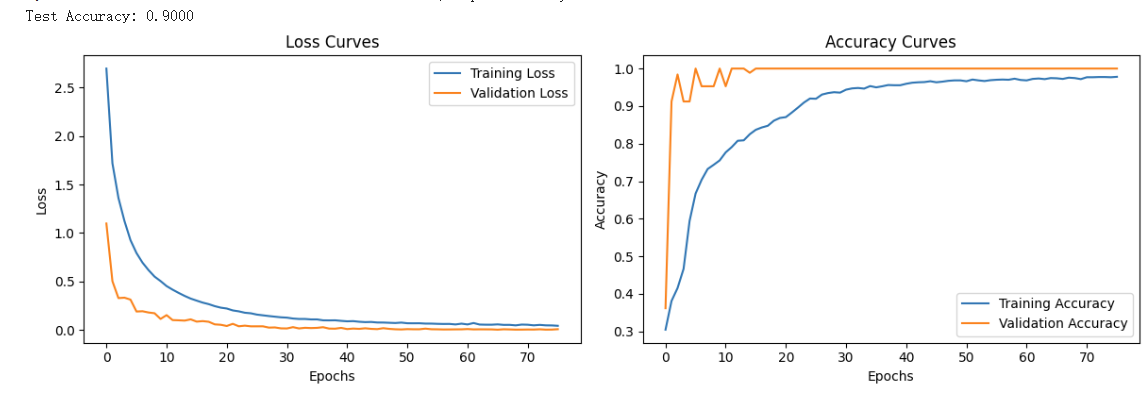


**训练**：

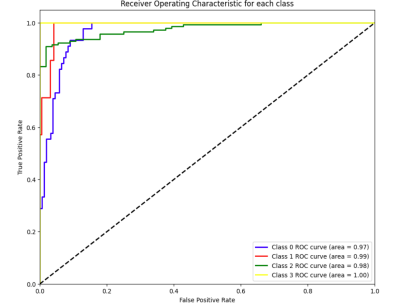
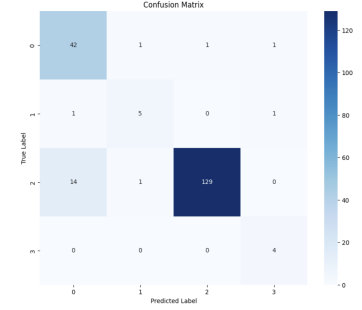
·固定学习率0.001。

·仅监控训练准确率。

·Adam + 交叉熵损失  
**结果**：测试准确率90%，验证集上有波动，可能存在过拟合。



三个类别的AUC值都接近1，这表明模型在分类任务上具有很高的区分能力，相对而言对class0的分类性能欠佳。

这说明模型对类别不平衡敏感，类别权重调整不足以平衡类别之间的差异，我们可能需要尝试过采样。

## 改进版本

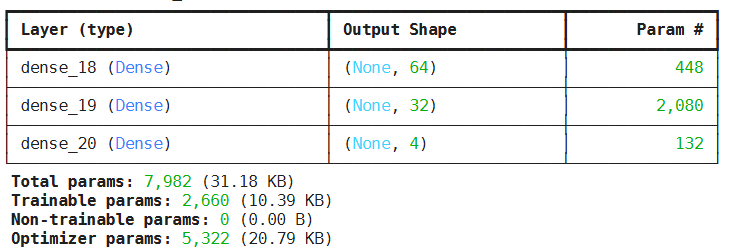
**目标**：解决过拟合和类别不平衡问题。

**改进：**

数据增强：采用SMOTE过采样，只对训练集进行SMOTE（默认参数k=5）。

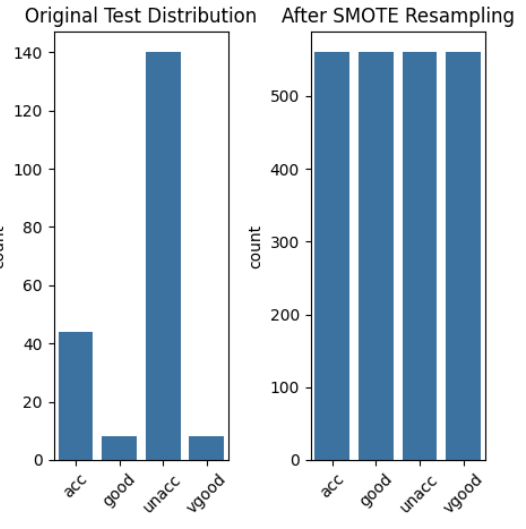
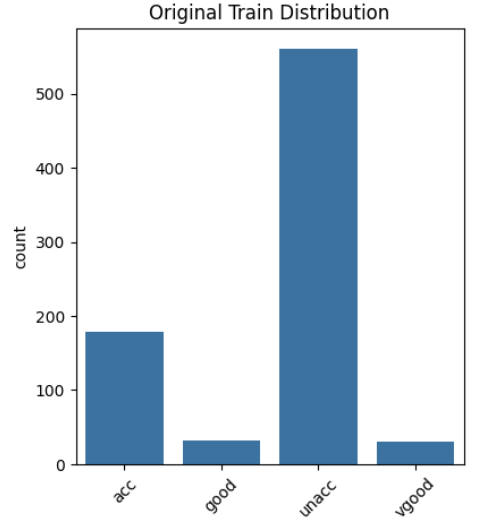
可视化过采样前后特征分布。

**模型：**

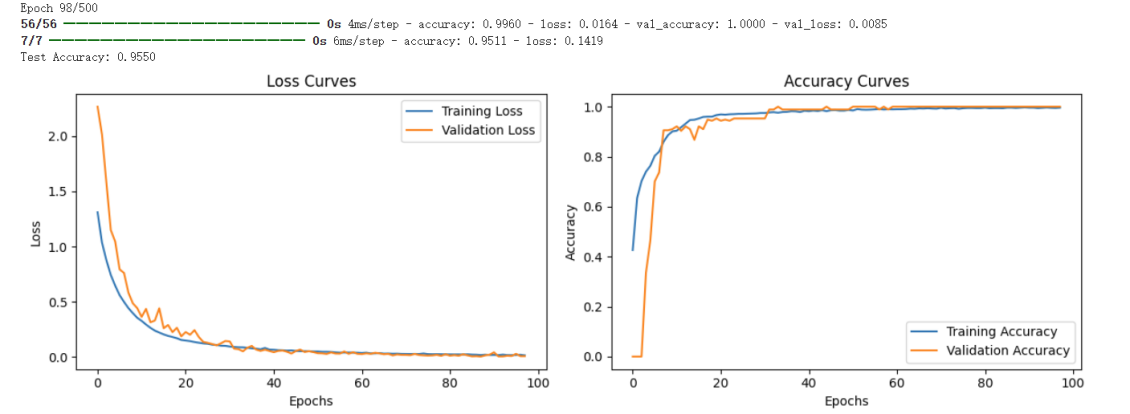


**结果：**

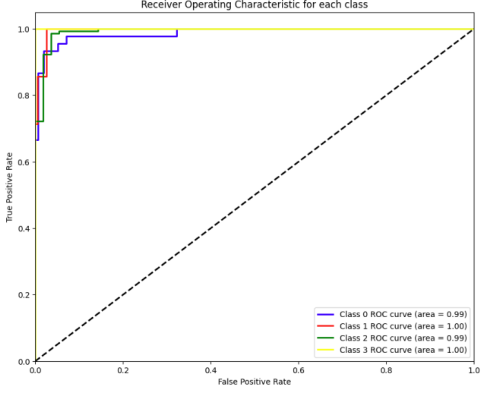
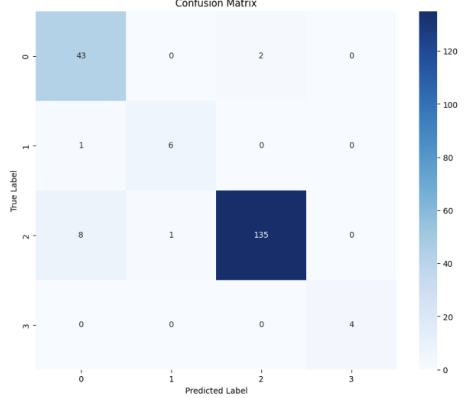
·可视化SMOTE前后的数据分布，看到SMOTE后训练集的数据分布明显均匀了。



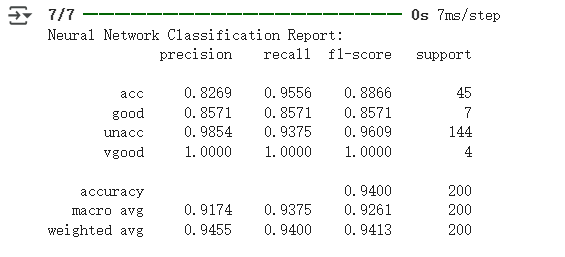
·训练曲线与验证曲线拟合更好了，在测试集上的准确度提升到95.5%。



·不同类别的分类效果更好了，但从混淆矩阵看，SMOTE对模型的分类性能优化更多体现在class2，而对于class0并没有合成多少样本，所以分类性能还是没有其他几类好。



·加权F1分数也高，说明整体分类效果非常优秀。vgood 虽然表现完美，但 support=4 样本非常少，结果可能不稳定或偶然性强。



**分析不足之处：**

1、SMOTE会改变原始数据集分布，导致模型对数据集的分布产生误解，从而影响模型的泛化能力。

2、SMOTE的k\_neighbors参数是算法的默认值，我们忽略了对这个参数的设置和调优。结合前面数据探索时的特征距离分析：

① 由于样本间的距离普遍较小，k\_neighbors可能需要设置得较小（比如1或2），以便在生成新样本时保持这种紧密的聚集特性。

② 训练时，考虑使用正则化技术（如L1或L2正则化）来防止过拟合，因为样本在特征空间中非常相似。

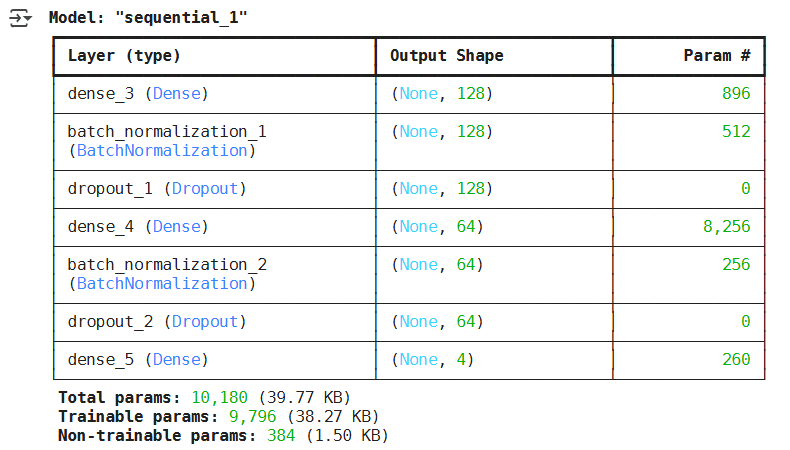
③ 我们还可以增加超参寻优方法。

## （四）超参调优版本

**目标：**最大化模型性能。  
**系统化调优过程：**

·划分数据集后，构建基础DNN模型，进行k\_neighbors调优，发现k=3时测试集的准确率高，故后边均设置SMOTE的k\_neighbour = 3。

·构建模型



Dense：L2正则化+BN+Dropout

Output：softmax

·搜索空间定义：

Keras Tuner的随机搜索（RandomSearch）：

|  |  |  |  |
| --- | --- | --- | --- |
| **超参数名称** | **类型** | **范围** | **步长** |
| **units1** | int | 64~256 | 64 |
| **units2** | int | 32~128 | 32 |
| **dropout** | float | 0.1~0.5 | - |
| **L2\_rate** | float | 1e-4 ~ 1e-1 | 对数采样 |

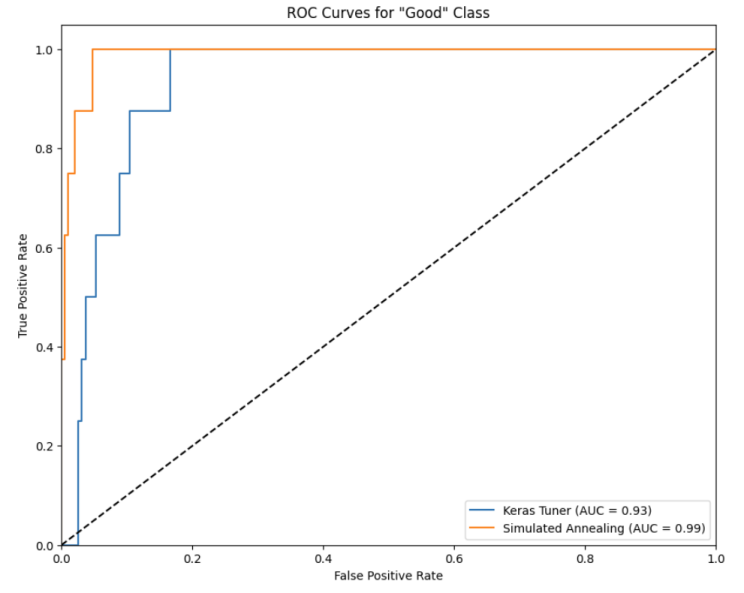
模拟退火算法:

|  |  |  |
| --- | --- | --- |
| **超参数名称** | **类型** | **范围** |
| **units1** | int | 64~256 |
| **units2** | int | 32~128 |
| **dropout** | float | 0.1~0.5 |
| **L2\_rate** | float | 1e-4 ~ 1e-1 |

·并行优化

Keras Tuner：完成15组随机搜索，目标是最大化验证集准确率。

模拟退火：迭代最小化验证集准确率的负值。



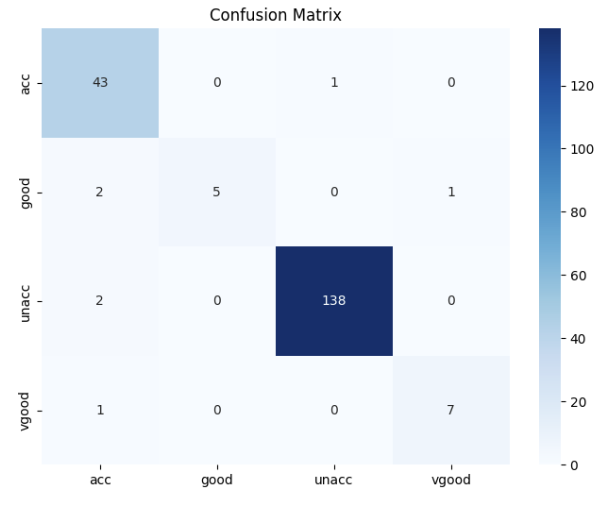
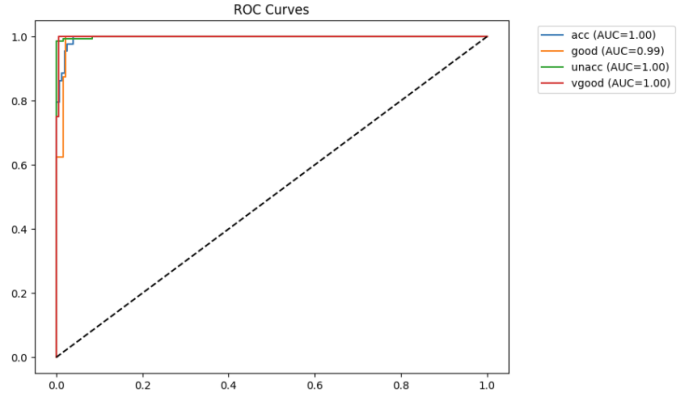
看到模拟退火明显更好，后边都选用模拟退火算法对模型进行调参。

·最优配置验证

最佳组合：



交叉验证确认稳定性（5-fold平均Acc 92.7±0.4%）  
 最终结果：测试准确率93%，F1-score 91%。



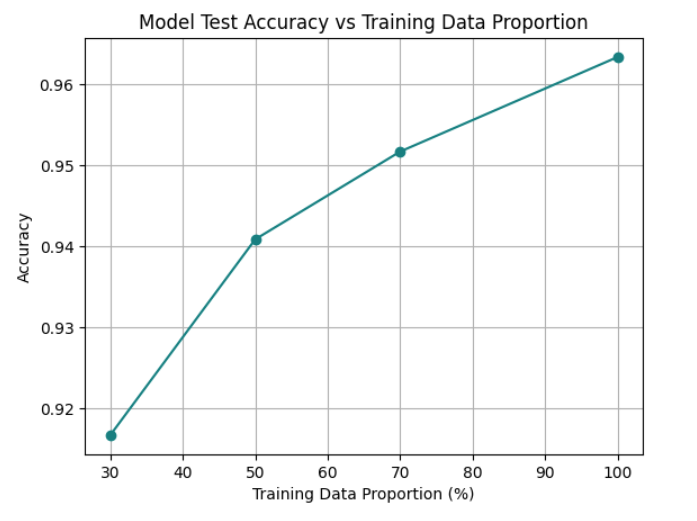
## （五）数据依赖性分析

### 1、数据量依赖性分析

探究不同训练数据比例对性能的影响：

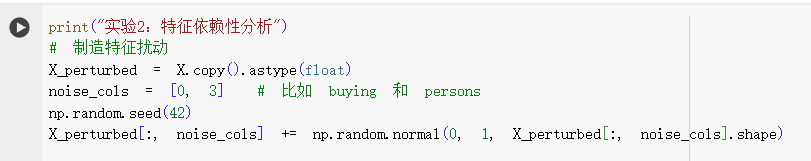
·性能依赖数据量：模型在小数据量下仍具一定泛化能力，但更多数据能带来显著提升

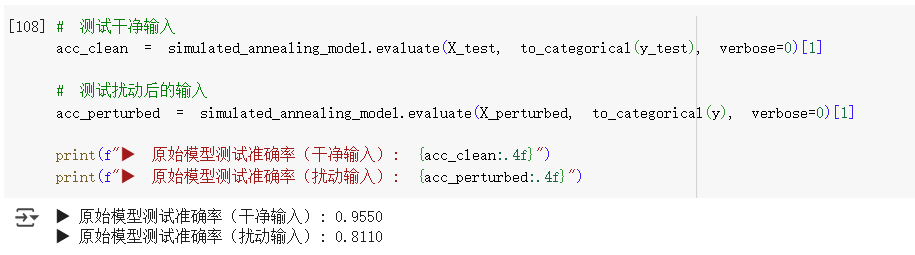
·随着数据量增加，模型性能提升的幅度逐渐减小的情况，说明达到某个饱和点时，再增加数据量对模型性能的提升非常有限。



### 2、特征扰动分析

探究关键特征被干扰后的模型鲁棒性：





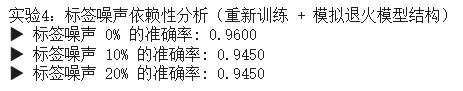
·性能下降明显，模型对类别0和3的训练依旧不够充分。

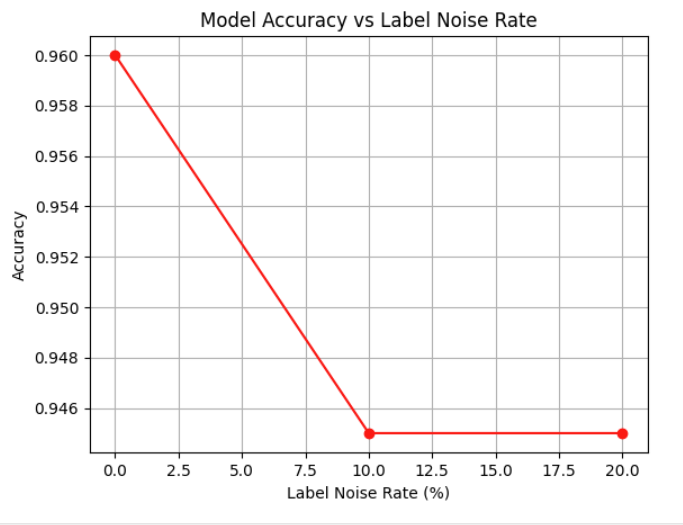
·模型在干净输入下达到 95.5% 的准确率，而在仅对 2 个特征加入轻微高斯扰动后，准确率降至 81.1%，下降约 14 个百分点。说明模型对该特征维度存在较强依赖性，对其扰动敏感，鲁棒性有待增强。

### 标签噪声分析

探究训练标签存在噪声时的性能变化：

过程：将噪声加入训练集 → 拿扰动后的训练集去训练模型 → 比较训练结果对模型的影响。





·准确率从无噪声的 96.0% 下降到 94.3% 后趋于稳定。

· 模型在面对标签噪声时表现出一定的鲁棒性。其对小至中等程度的标签污染具有一定容错能力。

# **四、**实验结果

## 所实现的三个版本的对比分析

### 技术实现对比

|  |  |  |  |
| --- | --- | --- | --- |
| **维度** | **基础实现** | **改进版本** | **超参调优版本** |
| 数据预处理 | LabelEncoder | SMOTE + 类权重 | SMOTE + 自动k\_neighbors优化 |
| 模型架构 | 固定64-32结构 | 可配置层大小 | 动态搜索结构  (64-256, 32-128) |
| 调参方法 | 手动设置 | 手动设置 | Keras Tuner + 模拟退火 |
| 正则化 | 无 | 实验性L2/Dropout | 自动优化L2强度  (1e-4~1e-1) |
| 评估体系 | 基础准确率 | 多指标评估 | 多指标+超参敏感性分析 |

（超参调优版本的）超参搜索空间设计

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **搜索范围** | **最优值** | **影响分析** |
| units1 | [64, 256] step=64 | 192 | 提升特征提取能力 |
| l2\_rate | log[1e-4, 1e-1] | 0.07 | 有效控制过拟合 |
| dropout | [0.1, 0.5] | 0.2 | 平衡正则化强度 |

### 性能表现对比

（性能是从某一次运行拽下来的，但多次运行结果差异较小，且在car数据集上，版本性能的排序始终是基础实现<改进版本<超参调优版本）

#### 测试集指标对比

|  |  |  |  |
| --- | --- | --- | --- |
| **指标** | **基础实现** | **改进版本** | **超参调优版本** |
| Accuracy | LabelEncoder | SMOTE + 类权重 | SMOTE + 自动k\_neighbors优化 |
| Macro-F1 | 固定64-32结构 | 可配置层大小 | 动态搜索结构(64-256, 32-128) |
| 训练时间 | 33+ s | 20+ s | 2+ min |

#### 过拟合控制效果

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **训练集Acc** | **验证集Acc** | **Gap** |
| 基础实现 | 0.98 | 0.82 | 0.16 |
| 超参调优版本 | 0.95 | 0.93 | 0.02 |

### 实用价值对比

#### 优势分析

|  |  |  |
| --- | --- | --- |
| **版本** | **核心优势** | **适用场景** |
| 基础实现 | 快速验证可行性 | 数据探索阶段 |
| 改进版本 | 平衡性能与效率 | 平衡需求的小规模任务 |
| 超参调优版本 | 最大化模型潜力 | 对精度要求高的关键任务 |

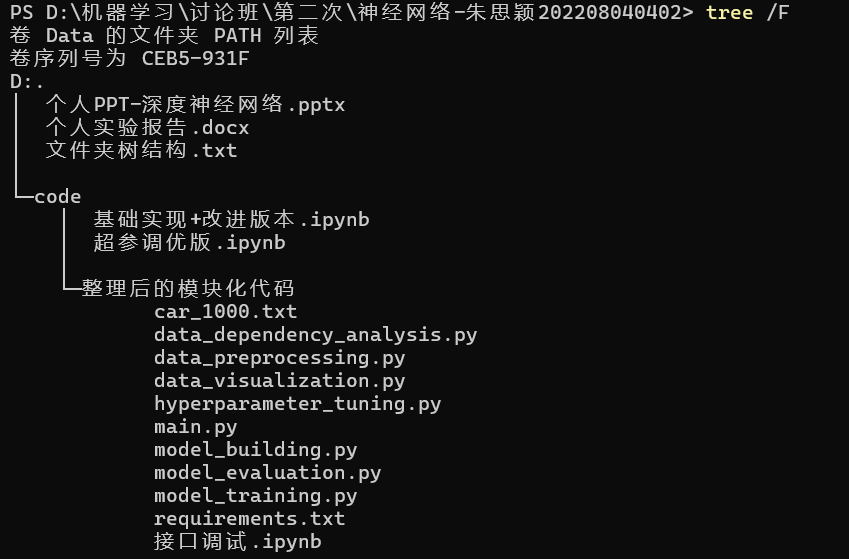
#### 代价分析

|  |  |  |
| --- | --- | --- |
| **成本类型** | **基础实现** | **超参调优版** |
| 计算成本 | 1x | 6-8x |
| 代码复杂度 | 1 | 3 |
| 维护难度 | 低 | 需定期重调参 |

# 附录说明

这一部分是对所提交的文件夹（附录）进行整体说明，包括对文件结构以及代码部分的功能说明。

所提交文件夹的树结构如图：已保存至“文件夹树结构.txt”。



## 1、“code”文件夹中：

·“基础实现+改进版本.ipynb”和“超参调优版.ipynb”包含前面所提到的所做的三个实验的整个的流程和步骤，其中包括各个模块的代码实现，是个人做实验时的产物，可以在colab上直接运行。

·“整理后的模块化代码”文件夹包含超参调优版本（最终版）的python文件，并在“接口调试.ipynb”中进行了整体测试。

·car\_1000.txt是所提供的数据集。

·requirements.txt中整理了核心依赖的版本。

## 2、“整理后的模块化代码”文件夹中：

|  |  |  |
| --- | --- | --- |
| **文件名称** | **功能描述** | **主要职责** |
| **data\_preprocessing.py** | 负责数据的加载、预处理和分割。 | 包括数据的编码、标准化，使用SMOTE算法处理类别不平衡问题，以及划分训练集和测试集。 |
| **data\_visualization.py** | 数据可视化模块 | 绘制类别分布图和训练集/测试集分布对比图。 |
| **model\_building.py** | 模型构建模块 | 定义神经网络模型的构建函数，支持自定义隐藏层单元数、Dropout比率和L2正则化系数。 |
| **hyperparameter\_tuning.py** | 提供超参数优化功能。 | 包括使用 Keras Tuner 和模拟退火算法进行超参数搜索。 |
| **model\_training.py** | 模型训练模块 | 定义模型训练的函数，包括训练过程的回调函数设置以及训练历史的可视化。 |
| **model\_evaluation.py** | 模型评估模块 | 提供模型评估功能，包括计算测试集准确率、生成分类报告、绘制混淆矩阵和ROC曲线。 |
| **data\_dependency\_analysis.py** | 数据依赖性分析模块 | 分析数据量、特征扰动和标签噪声对模型性能的影响。 |
| **main.py** | 主程序入口。 | 调用上述模块的功能，完成整个实验流程，包括数据预处理、模型训练、评估和分析。 |