欺诈短信识别系统 (SMS-SCAM-DETECTION)

1. 项目概述 (Project Overview)

属性	描述
项目名称	欺诈短信识别系统 (SMS-SCAM-DETECTION)
目标	基于自然语言处理 (NLP) 和机器学习技术，高效、准确地对英文短信进行**二分类（Ham/Spam）**。
技术亮点	包含从零实现核心机器学习模型的能力，以及完整的数据工程和模型部署流程。
核心算法	**逻辑回归 (Logistic Regression)**，同时实现了自研版本和 Sklearn 版本进行对比和验证。
主要工具	Python, NumPy, SciPy (Sparse Matrix), Pandas, Scikit-learn, Matplotlib, Seaborn

2. 技术细节与实现深度 (Technical Details & Implementation Depth)

2.1 数据预处理与特征工程 (data_preprocessing.py)

1.文本清洗与规范化：执行小写化，并对敏感或噪声实体（如 **URL**、**邮箱**、**数字/电话号码**）进行统一替换，以提高特征的泛化性。

2.高效向量化：采用 **TF-IDF (Term Frequency-Inverse Document Frequency)** 将文本转换为**稀疏矩阵**特征，并利用稀疏数据结构提高内存和计算效率。

3.解决类别不平衡：

a.使用 **stratify** 划分训练/测试集以保持标签分布。

b.在训练集上应用 **SMOTE (Synthetic Minority Over-sampling Technique)** 对少数类（Spam）进行过采样，有效解决了垃圾短信数据集固有的数据倾斜问题。

2.2 核心算法实现：从零实现逻辑回归 (model_from_scratch.py)

1.自研算法健壮性： 实现了支持以下关键特性的自定义 **Logistic Regression** 类：

a.稀疏矩阵支持：模型能直接高效处理 TF-IDF 输出的 CSR 稀疏矩阵。

b.Mini-batch 梯度下降：实现了批处理（Mini-batch SGD），优化了大型数据集的训练效率。

c.L2 正则化：集成 L2 惩罚项到损失函数和梯度更新中，用于防止过拟合。

2.类别权重处理：内置对 **class_weight='balanced'** 参数的支持，通过在损失函数中自动计算并应用**样本权重**，无需外部采样即可在训练中动态平衡类别贡献。

3.收敛性监控：实现了 **loss_history** 记录和基于 tol 的**提前停止**机制。

2.3 模型对比与专业评估 (model_sklearn.py, evaluation.py)

1.基线验证：使用 **Sklearn 的 Logistic Regression** 作为基线模型进行训练和对比，以验证自研模型实现的正确性和有效性。

2.全面评估指标：模型评估聚焦于 **Accuracy**、**Precision**、**Recall** 和 **F1-Score**，特别是 Recall，确保诈骗短信被有效捕获。

3.可视化分析：实现了**混淆矩阵 (Confusion Matrix)** 和 **ROC 曲线 (Receiver Operating Characteristic)** 的绘制功能，并通过 **AUC (Area Under the Curve)** 评估模型性能的稳定性。

2.4 工程化与用户部署 (main.py, predict_friendly.py)

1.模型持久化 (MLOps)：使用 **joblib** 序列化并保存了 **TF-IDF 向量器**和**两个模型**（自研和 Sklearn 版本），确保模型可以被快速、准确地加载和部署。

2.用户友好预测接口：提供了灵活的 **predict_friendly.py** 脚本，支持**命令行单条输入**、**代码列表批量输入**和**文件批量读取**，清晰地输出了预测标签和 Spam 概率。

3. 结果与能力总结 (Results & Key Takeaways)

3.1 模型性能对比 (Benchmark)

两个模型在测试集上均取得了优秀的性能，验证了方法的有效性：

预估性能指标 (Expected Performance Metrics)

模型	准确率 (Accuracy)	精确率 (Precision)	召回率 (Recall)	F1-Score	AUC
Logistic Regression (Scratch)	97.58%	90.67%	91.28%	90.97%	0.98
Logistic Regression (Sklearn)	98.21%	94.48%	91.95%	93.20%	0.98

3.2 学习心得与能力提升 (Key Takeaways & Skill Development)

1.算法与代码健壮性提升（自学与调试历程）

a.理解原理与精进实现：项目开始前我已经自学机器学习的部分内容，掌握了 Logistic Regression 的核心数学原理（Sigmoid 函数、对数似然损失、梯度计算）。在**从零实现 (model_from_scratch.py)**过程中，最初的模型仅支持 NumPy 数组，容易因**稀疏矩阵**输入而导致效率低下。

b.代码调试与优化： 通过深入调试和学习 SciPy 的功能，我主动重构了梯度计算部分，使其能高效处理 **CSR 稀疏矩阵**，解决了大规模 TF-IDF 特征下的性能瓶颈，极大地提高了代码的**健壮性**和**通用性**。

c.解决实际问题： 通过最初的测试脚本发现了数据集存在严重**类别不平衡**时，我没有简单地依赖外部库，而是在自研模型的损失计算和梯度更新中**手动引入了样本权重 (class_weight='balanced')**，确保了模型对少数类（Spam）的敏感性，直接提升了**预测召回率**。

2.系统化思维构建（端到端 MLOps 思维）

a.需求分析：确定核心问题（诈骗短信识别）和关键指标（高召回率）

b.数据工程： 独立设计了从**清洗**（URL/数字规范化）、**特征提取**（TF-IDF）到**数据平衡**（SMOTE）的完整数据管道 **(data_preprocessing.py)**。

c.双模型并行：采用**自研模型**和 **Sklearn 基线模型**并行开发和对比，确保结果可靠。

d.专业评估与可视化： 建立了**多维度评估体系**（F1, Recall, AUC, 混淆矩阵），而非仅关注准确率，保证了模型的可解释性。

e.部署与交付：实现了**模型和向量器的持久化**，并提供了**用户友好、多输入模式的预测接口**，完成了从算法到产品的闭环。

3.AI 工具辅助学习与成长

在项目开发过程中，我积极将 **AI 工具**（例如大型语言模型）视为强大的编程助手和知识库：

a.我利用 AI 辅助理解和确认 NumPy/SciPy 在处理**稀疏矩阵点乘**时的最佳实践，加快了代码重构和调试速度。

b.我使用 AI 辅助验证自研模型**梯度公式**的推导和实现细节，确保了算法的**数学正确性**。

c.AI 的使用不仅提高了我的开发效率，更重要的是，我在过程中始终保持**批判性思维**，对 AI 生成的代码进行验证和整合，从而加速了对复杂工程细节的掌握和**代码编写能力**的成长。

4. 后续改进方向 (Future Enhancements)

为将此项目推向更高的工程和算法水平，后续在我对人工智能学科(机器学习、深度学习等内容)深入学习并理解后，希望能进行以下改进：

1.算法升级（深度学习）：引入更先进的 NLP 模型，如 **Word Embeddings (Word2Vec/GloVe)** 结合**循环神经网络 (RNN/LSTM)**或**预训练语言模型 (BERT)**，以捕获更深层次的文本语义信息，进一步提升召回率。

2.模型超参数优化：部署 **Grid Search** 或 **Bayesian Optimization** 对 LR 模型的正则化强度 (λ)、学习率 (lr) 和特征数量 **(max_features)** 等进行系统性调优，以达到最优性能。

3.特征工程扩展：引入 **N-gram 特征**或**词性标注 (POS tagging)** 特征，作为 TF-IDF 的补充，以增强模型对上下文和语法结构的理解。

4.Web 部署 (Full-stack ML)： 利用 **Flask/Streamlit** 等框架，将预测服务部署为 Web API 或小型前端应用，实现模型的在线实时预测。