**上海电力大学**

**本科毕业设计（论文）**



题 目： 基于集成学习的恶意PE软件特征检测与识别

院 系： 计算机科学与技术学院

专业年级： 信息安全2021级

学生姓名： 郑蜀杭 学号： 20210521

指导教师： 陈思远

2025 年4 月 20日

上海电力大学

本科毕业设计（论文）原创性声明

本人郑重申明：本人所呈交的毕业论文，是在指导老师的指导下独立进行研究所取得的成果。论文中凡引用他人已经发布或未发表的成果、数据、观点等，均已明确注明出处。论文中除已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究成果做出重要贡献的个人和集体，均已在论文中以明确的方式标明。

本声明的法律责任由本人承担。

 论文作者签名： 日期：2025年4月25日

上海电力大学

本科毕业设计（论文）使用授权声明

本人在指导老师的指导下所完成的论文及相关的资料，知识产权归属上海电力大学。本人完全了解上海电力大学有关保存、使用毕业论文的规定，同意学校保存或向国家有关部门或机构送交论文的纸质版或电子版，允许论文被查阅或借阅。本人授权上海电力大学可以将本毕业论文的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存或编汇本毕业论文。如果发表相关成果，一定征得指导教师同意，且第一署名单位为上海电力大学。本人毕业后使用毕业论文或与该论文直接相关的学术论文或成果时，第一署名单位仍然为上海电力大学。



论文作者签名： 日期：2025年4月28日

指导教师签名： 日期：2025年4月28日

摘 要

恶意软件作为一种具有恶意目的的软件程序，对计算机系统和网络安全构成了严重威胁。近年来，恶意软件的攻击频率和规模不断增长，其通过复杂的混淆技术快速衍生变种，使得传统的检测方法难以有效应对。为了应对这一挑战，本文提出了一种基于集成学习的恶意PE软件特征检测与识别方法，旨在提高恶意软件检测的准确性和效率。

本文首先分析了恶意PE软件的常见特征，包括PE数据结构、直方图特征等，并介绍了集成学习技术如Stacking、OOF、加权融合以及特征哈希技术。基于此，构建了多维静态特征检测方案，涵盖PE数据结构特征解析、直方图特征模型检测、字符规则匹配特征等，并通过特征融合方案整合多源特征。进一步，设计了基于多模型融合的恶意PE识别模型，包括基于CNN的直方图特征识别模型、基于堆叠的静态PE特征识别模型以及基于LightGBM的特征工程模型，并通过多层集成学习方案提升检测性能。最终，实现了基于集成学习的恶意PE识别系统，采用React框架、FastAPI框架和SQLite数据库进行开发，实现了样本管理、特征提取、集成学习与检测以及数据可视化等功能。系统测试结果表明，该方法能够高效检测恶意PE软件，具有良好的准确性和泛化能力。

关键词：恶意PE软件；集成学习；特征检测；模型融合；系统

Feature detection and recognition of malicious PE software based on ensemble learning

Abstract

Malicious software, as a type of software program with malicious purposes, poses a serious threat to computer systems and network security. In recent years, the frequency and scale of malicious software attacks have been continuously increasing, and their rapid derivation of variants through complex obfuscation techniques has made it difficult for traditional detection methods to effectively respond. To address this challenge, this paper proposes a feature detection and recognition method for malicious PE software based on ensemble learning, aiming to improve the accuracy and efficiency of malware detection.

This article first analyzes the common characteristics of malicious PE software, including PE data structures, histogram features, etc., and introduces ensemble learning techniques such as Stacking, OOF, weighted fusion, and feature hashing techniques. Based on this, a multidimensional static feature detection scheme was constructed, covering PE data structure feature analysis, histogram feature model detection, character rule matching features, etc., and integrating multi-source features through a feature fusion scheme. Furthermore, a malicious PE recognition model based on multi model fusion was designed, including a CNN based histogram feature recognition model, a stacked static PE feature recognition model, and a LightGBM based feature engineering model. The detection performance was improved through a multi-layer ensemble learning scheme. Finally, a malicious PE recognition system based on ensemble learning was implemented, developed using React framework, FastAPI framework, and SQLite database, and implemented functions such as sample management, feature extraction, ensemble learning and detection, and data visualization. The system test results show that this method can efficiently detect malicious PE software, with good accuracy and generalization ability.

**Key Words**: Malicious PE software; Integrated learning; Feature detection; Model fusion; system

目录

[第一章 绪论 1](#_Toc197529768)

[1.1 研究背景及意义 1](#_Toc197529769)

[1.2 国内外研究现状 3](#_Toc197529770)

[1.2.1 特征匹配技术检测 3](#_Toc197529771)

[1.2.2 机器学习技术检测 4](#_Toc197529772)

[1.2.3 集成学习技术检测 6](#_Toc197529773)

[1.3 常用数据集 10](#_Toc197529774)

[1.4 论文主要工作 11](#_Toc197529775)

[1.5 论文组织与结构 11](#_Toc197529776)

[第二章 技术背景及相关研究 13](#_Toc197529777)

[2.1 恶意PE软件常见特征 13](#_Toc197529778)

[2.1.1 PE数据结构 13](#_Toc197529779)

[2.1.2 直方图特征 13](#_Toc197529780)

[2.2 集成学习技术 15](#_Toc197529781)

[2.2.1 Stacking 15](#_Toc197529782)

[2.2.1 OOF 15](#_Toc197529783)

[2.2.2 加权融合 16](#_Toc197529784)

[2.3 特征哈希技术 16](#_Toc197529785)

[2.4 系统开发所用技术 17](#_Toc197529786)

[2.4.1 React框架 17](#_Toc197529787)

[2.4.2 FastAPI框架 17](#_Toc197529788)

[2.4.3 SQLite数据库 18](#_Toc197529789)

[2.5 本章小结 19](#_Toc197529790)

[第三章 基于多维分析的样本静态特征检测方案 20](#_Toc197529791)

[3.1 特征检测方案概述 20](#_Toc197529792)

[3.2 PE数据结构特征解析 20](#_Toc197529793)

[3.2.1 PE软件结构概述 20](#_Toc197529794)

[3.2.2 各结构特征检测 21](#_Toc197529795)

[3.2.3 特征哈希向量化 23](#_Toc197529796)

[3.3 直方图特征模型检测 23](#_Toc197529797)

[3.3.1 字节直方图特征检测 23](#_Toc197529798)

[3.3.2 字节熵直方图特征检测 24](#_Toc197529799)

[3.4 字符规则匹配特征 26](#_Toc197529800)

[3.4.1 规则匹配方法概述 26](#_Toc197529801)

[3.4.2 Yara规则匹配 26](#_Toc197529802)

[3.4.3 自定义规则方案 27](#_Toc197529803)

[3.5 特征融合方案 29](#_Toc197529804)

[3.6 本章小结 30](#_Toc197529805)

[第四章 基于多模型融合的恶意PE识别模型构建 31](#_Toc197529806)

[4.1 恶意PE识别模型总体设计 31](#_Toc197529807)

[4.2 基于多分支增强训练CNN的恶意PE识别模型构建 32](#_Toc197529808)

[4.2.1 特征引入与网络引入 32](#_Toc197529809)

[4.2.2 网络结构与超参数设计 33](#_Toc197529810)

[4.2.3 训练策略与优化配置 33](#_Toc197529811)

[4.2.4 消融实验和性能对比 34](#_Toc197529812)

[4.3 基于基模型堆叠的静态PE特征识别模型构建 35](#_Toc197529813)

[4.3.1 模型Stacking构建方案 35](#_Toc197529814)

[4.3.2 异构基模型选型与调优 36](#_Toc197529815)

[4.3.3 基模型和特征组消融实验 36](#_Toc197529816)

[4.3.4 实验结果与讨论 37](#_Toc197529817)

[4.4 基于LightGBM特征工程的识别模型构建 38](#_Toc197529818)

[4.4.1 特征选择与模型训练 38](#_Toc197529819)

[4.4.2 特征重要性分析 39](#_Toc197529820)

[4.4.3 主成分分析及降维对比 41](#_Toc197529821)

[4.4.4 LightGBM模型指标和综合对比 44](#_Toc197529822)

[4.5 本章小结 45](#_Toc197529823)

[第五章 基于Stacking和加权融合的多层集成学习方案 46](#_Toc197529824)

[5.1 多层集成学习模型框架 46](#_Toc197529825)

[5.2 子模型并行训练 46](#_Toc197529826)

[5.3 Stacking集成方案 47](#_Toc197529827)

[5.3.1 OOF预测生成和融合策略 47](#_Toc197529828)

[5.3.2 元模型训练与验证 47](#_Toc197529829)

[5.4 加权融合方案 48](#_Toc197529830)

[5.4.1 权重参数优化 48](#_Toc197529831)

[5.4.2 实验结果及误差分析 48](#_Toc197529832)

[5.5 本章小结 50](#_Toc197529833)

[第六章 基于集成学习的恶意PE识别系统的设计与实现 51](#_Toc197529834)

[6.1 系统需求及可行性分析 51](#_Toc197529835)

[6.1.1 系统需求分析 51](#_Toc197529836)

[6.1.2 系统可行性分析 52](#_Toc197529837)

[6.2 系统总体设计 53](#_Toc197529838)

[6.2.1 系统总体架构 53](#_Toc197529839)

[6.2.2 系统数据库设计 54](#_Toc197529840)

[6.2.3 系统API接口设计 56](#_Toc197529841)

[6.3 系统详细设计及功能实现 57](#_Toc197529842)

[6.3.1 系统功能概述 57](#_Toc197529843)

[6.3.2 样本管理模块 58](#_Toc197529844)

[6.3.3 特征提取模块 61](#_Toc197529845)

[6.3.4 集成学习与检测模块 65](#_Toc197529846)

[6.3.5 数据可视化与前端展示模块 65](#_Toc197529847)

[6.4 系统测试 67](#_Toc197529848)

[6.4.1 系统测试环境 67](#_Toc197529849)

[6.4.2 系统测试方案 68](#_Toc197529850)

[6.4.3 改进方法 70](#_Toc197529851)

[6.5 本章小结 70](#_Toc197529852)

[第七章 总结与展望 71](#_Toc197529853)

[7.1 总结 71](#_Toc197529854)

[7.2 展望 71](#_Toc197529855)

[致谢 72](#_Toc197529856)

[参考文献 74](#_Toc197529857)

图目录

[图 2‑1 PE数据结构示意图 13](#_Toc197529203)

[图 2‑2滑动窗口计算字节熵 14](#_Toc197529204)

[图 3‑1 PE样本特征检测总体流程 20](#_Toc197529205)

[图 4‑1恶意PE识别模型总体架构 31](#_Toc197529206)

[图 4‑2消融实验三模型柱状图（左）雷达图（右）对比 35](#_Toc197529207)

[图 4‑3消融实验模型损失函数（左）和模型准确率损失（右）对比 35](#_Toc197529208)

[图 4‑4模型指标热力图（左）模型指标柱状图（右） 36](#_Toc197529209)

[图 4‑5基于基模型的消融实验雷达图（左）柱状图（右） 37](#_Toc197529210)

[图 4‑6基于特征组的消融实验雷达图（左）柱状图（右） 37](#_Toc197529211)

[图 4‑7基模型及Stacking模型性能对比 37](#_Toc197529212)

[图 4‑8Stacking集成对于各基模型的性能提升 38](#_Toc197529213)

[图 4‑9前20个特征重要性分布箱型图 40](#_Toc197529214)

[图 4‑10特征样本分布情况 41](#_Toc197529215)

[图 4‑11各主成分解释方差 42](#_Toc197529216)

[图 4‑12主成分特征权重（TOP 10） 43](#_Toc197529217)

[图 4‑13 PCA降维可视化分析 44](#_Toc197529218)

[图 4‑14 LightGBM等模型指标对比分析 45](#_Toc197529219)

[图 5‑1两类元模型性能评估对比 48](#_Toc197529220)

[图 5‑2最终模型测试集表现情况 49](#_Toc197529221)

[图 5‑3融合性能提升饼图（左）多种融合策略性能提升对比（右） 49](#_Toc197529222)

[图 5‑4最终模型性能表现情况 50](#_Toc197529223)

[图 6‑1系统总体需求架构示意图 51](#_Toc197529224)

[图 6‑1 PE软件样本检测流程 52](#_Toc197529225)

[图 6‑1系统总体架构序列图 54](#_Toc197529226)

[图 6‑4系统入口页面展示 58](#_Toc197529227)

[图 6‑5样本上传页面展示 59](#_Toc197529228)

[图 6‑6样本分析完成页面展示 59](#_Toc197529229)

[图 6‑7样本列表页面展示 60](#_Toc197529230)

[图 6‑8样本列表搜索功能展示 60](#_Toc197529231)

[图 6‑9样本分析结果详情页面展示 61](#_Toc197529232)

[图 6‑10字节直方图特征分析结果 62](#_Toc197529233)

[图 6‑11字节熵直方图特征分析结果 62](#_Toc197529234)

[图 6‑12 PE静态特征分析结果 63](#_Toc197529235)

[图 6‑13 LIEF特征深度分析结果 64](#_Toc197529236)

[图 6‑14特征工程分析结果 65](#_Toc197529237)

[图 6‑15模型特征得分计算 65](#_Toc197529238)

[图 6‑16数据仪表盘页面展示 66](#_Toc197529239)

[图 6‑17节区特征分析图表可视化 67](#_Toc197529240)

[图 6‑18字符串匹配特征分析图表可视化 67](#_Toc197529241)

表目录

[表 3‑1 恶意关键字特征字典 28](#_Toc197529305)

[表 3‑2 PE直方图特征（512维） 29](#_Toc197529306)

[表 3‑3 PE关键静态特征（967维） 29](#_Toc197529307)

[表 3‑4 PE特征工程综合特征（56维） 29](#_Toc197529308)

[表 4‑1消融实验模型指标结果 34](#_Toc197529309)

[表 4‑2 LightGBM特征工程重要特征（TOP 10） 39](#_Toc197529310)

[表 4‑3 LightGBM模型指标和综合对比结果 44](#_Toc197529311)

[表 6‑1样本信息表 54](#_Toc197529312)

[表 6‑2直方图特征表 55](#_Toc197529313)

[表 6‑3 PE静态特征表 55](#_Toc197529314)

[表 6‑4特征工程表 55](#_Toc197529315)

[表 6‑5 LIEF分析表 56](#_Toc197529316)

[表 6‑6主要API接口详情表 56](#_Toc197529317)

[表 6‑7功能点接口总览表 57](#_Toc197529318)

[表 6‑8系统测试用例及其测试结果 68](#_Toc197529319)

# 绪论

## 研究背景及意义

恶意软件（Malware）是指任何蓄意编写以损害计算机系统或其用户的软件代码，包括勒索软件、特洛伊木马、间谍软件、广告软件等，旨在盗取个人隐私信息、破坏计算机系统、窃取敏感数据等【1】。几乎所有现代网络攻击都涉及某种形式的恶意软件，其目的可能是获取数据、劫持设备或扰乱关键系统等。近年来，随着网络应用的普及和计算资源增长，恶意软件样本的数量呈爆炸式增长，成为网络安全的一大挑战。根据 Proofpoint 的威胁情报报告【2】，2021年AV-TEST登记的恶意软件总量已超过 1.3 十亿，相比2013年约 1.82 亿的登记量，十年内增长了超过 600%。这一巨量增长表明，尽管各类安全产品不断升级，攻击者仍通过更频繁、更隐蔽的变种持续扩大样本池。

对 AV-TEST Institute “Malware Statistics & Trends Report”【3】最新统计数据分析，恶意软件的增长逐年递增，每日新增量已远超十万级规模，累计样本量也突破十亿级大关。AV-TEST官方数据显示，截至2025年，每天新登记的恶意程序和潜在有害应用（PUA）超过 450,000。这些数字不仅反映出攻击者投放样本的速度在持续加快，也对安全防护提出了更高的实时检测与响应需求。如此规模的样本增长对传统签名和单一检测机制造成巨大压力，每日数十万的新样本几乎不可能通过人工编写签名及时覆盖。Akwconsultants【4】指出，尽管常规杀软能够识别约 90% 的已知威胁，但面对高频变种仍存在检测盲区。

因此如何快速、准确地检测和识别恶意软件已成为急需解决的问题。目前，针对恶意软件的检测识别方法主要依赖于特征提取和机器学习算法。静态特征提取通过逆向分析可执行文件内容来获取信息，例如提取PE文件的头字段、导入表、字符串、字节/二元码统计、操作码序列或函数调用图等【5】。例如，Saxe等人针对Windows PE文件提取了字节频率、二元字节频率、导入表和元数据等四类静态特征用于分类【5】。动态特征提取则在沙箱或虚拟机环境中运行程序，收集系统调用序列、API调用、网络行为日志等运行时行为特征，以捕获恶意活动的动态模式。基于上述各种特征，研究者通常采用支持向量机、随机森林、决策树、深度神经网络等机器学习算法进行恶意软件分类【6】。相比传统的签名检测，这些方法能够识别未知变种，提高检测泛化能力；例如深度学习模型无需手工制定签名即可自动从数据中学习攻击特征。然而，这些方法也存在不足：静态分析对加壳、代码混淆等对抗技术不敏感，容易被规避；动态分析检测成本高、覆盖不全，且恶意样本可能通过延时、反沙箱检测等手段绕过监控【6】。此外，单一分类模型往往易受训练数据分布变化影响，存在过拟合或在复杂数据下误报漏报率偏高的问题。为此，学者们开始引入集成学习思想，通过组合多个分类器或特征集来提升检测性能。例如，采用Bagging、Boosting或Stacking等策略将多个弱分类器集成为强分类器，可以在一定程度上提高检测准确率和鲁棒性，但相应地也增加了模型复杂度和计算开销。

在上述背景下，集成学习方法在恶意PE软件检测中展现出独特优势。集成学习通过将多个不同模型的预测结果进行融合，能够有效降低单模型的方差和偏差，提高整体泛化能力。例如，对恶意软件样本进行多视角特征提取并分别训练多个基模型，然后采用加权投票或堆叠融合，可以综合各模型的优势，增强对复杂样本的判别力。杨望等人提出了一种多特征集成学习框架，他们分别提取了非PE结构特征、可见字符串与汇编码序列特征、PE结构特征和函数调用关系等五类静态特征，并使用Bagging和Stacking算法构建多个基模型。最终通过加权投票聚合这些模型输出，检测准确率高达96.99%，对加壳和混淆手段的恶意样本也能保持较高识别率【6】。这表明集成学习能够有效提升模型稳定性，降低过拟合风险，对未知变种具有更强的识别能力。总的来看，集成学习在处理大规模、多样化恶意软件样本时具有重要意义：它不仅提高了检测的准确度和鲁棒性，还增强了系统对样本多样性和快速迭代的适应能力，从而为构建更加可靠的恶意软件检测系统提供了有力支持。

【1】IBM中国. 什么是恶意软件 (Malware)?[EB/OL]. <https://www.ibm.com/cn-zh/topics/malware>, [2025-04-07].

【2】Proofpoint. What Is Malware? Proofpoint US[EB/OL]. <https://www.proofpoint.com/us/threat-reference/malware>, [2025-04-07].

【3】AV-TEST Institute. Malware Statistics & Trends Report. AV-TEST[EB/OL]. https://www.av-test.org/en/statistics/malware, [2025-04-07].

【4】Akwconsultants. A Strong Antivirus Software Isn’t Enough Anymore [EB/OL]. <https://akwconsultants.com/antivirus-software-isnt-enough-cybersecurity>, [2025-04-07].

【5】张玉清, 董颖, 柳彩云, 等. 深度学习应用于网络空间安全的现状、趋势与展望[J]. 计算机研究与发展, 2018, 55(6): 1117-1142.

【6】 杨望, 高明哲, 蒋婷. 一种基于多特征集成学习的恶意代码静态检测框架[J]. 计算机研究与发展, 2021, 58(5): 1021-1034.

## 国内外研究现状

目前针对恶意软件分析的方法有基于特征匹配技术检测方法、基于机器学习技术检测方法、基于集成学习技术检测方法等。

### 特征匹配技术检测

特征匹配技术检测是一种快速识别恶意PE显著特征的方式，依赖于从可执行文件中提取预设好的特征检测逻辑，主要包括静态特征、动态特征、图结构特征等。

基于静态分析技术检测恶意软件是网络安全领域的重要研究方向。静态分析不执行程序，而是直接对程序代码、可执行文件等进行分析，以发现其中可能存在的恶意模式、语义特征等。通常会提取文件属性及元数据等，包括PE头信息（如文件大小、时间戳、机器类型等）、节区特征（节区数量、大小、熵值等）、导入表中特定的DLL和API函数列表，以及文件中提取到的可疑字符串等（如URL、路径、注册表键等）【1】。例如，Yousuf等【1】通过静态提取PE头属性、节区属性和导入函数列表等四类特征，构建了分类器。Bensaoud等【2】通过特征匹配（签名）方法，利用已知样本特征库，通过哈希、模式匹配等方式检测已知恶意代码。

此外，Mihai Christodorescu和S. Jha【201】提出了一种用于检测可执行文件中恶意模式的架构，该架构能够抵御常见的混淆变换。他们开发的原型工具SAFE（a static analyzer for executables）通过实验结果证明了其有效性。这种方法为可执行文件的静态分析检测提供了一种可行的技术路径，从模式识别的角度来发现恶意软件。然而，该方法可能面临随着恶意软件混淆技术不断发展，原有的模式检测能力可能下降的局限，需要不断更新和优化检测模式库。

随后，Mihai Christodorescu等人【202】提出了语义感知的恶意软件检测算法。实验评估表明，该算法能够以相对较低的运行时开销检测出恶意软件的变体，并且能够抵御黑客常用的混淆手段。该研究从语义层面出发，突破了传统基于特征模式检测的局限，通过分析程序的语义信息来识别恶意软件，具有更强的适应性和鲁棒性。但该算法可能在处理复杂语义和大规模程序时面临性能挑战，需要进一步优化算法的效率和准确性。

随着恶意软件的更新与迭代，变形恶意软件通过不断改变自身代码来逃避检测，给恶意软件检测带来了巨大挑战。Qinghua Zhang和D. Reeves 【203】提出了MetaAware方法，利用对可执行文件的全自动静态分析来总结和比较程序语义，以识别变形恶意软件。他们认为这种检测方法对于恶意软件编写者来说难以绕过。该方法从程序语义的角度出发，为变形恶意软件的识别提供了新的思路。然而，该方法可能在面对一些高级变形技术时，语义分析的准确性会受到影响，并且对于大规模样本的处理效率也需要进一步提高。

动态特征检测主要通过在受控沙箱中执行样本，记录程序的实际行为，包括系统调用序列、文件/注册表操作、网络活动等。其中，Windows API调用序列是典型的动态行为特征，反映了恶意软件与操作系统的交互模式【3】。近年来，研究者常将API调用序列视作序列数据，使用RNN/LSTM等模型进行分析。Owoh【3】等提出结合GRU和GAN的混合深度模型，通过学习API调用序列检测恶意代码。

【1】Yousuf M I, Anwer I, Riasat A, et al. Windows malware detection based on static analysis with multiple features[J]. PeerJ Computer Science, 2023, 9: e1319.

【2】Bensaoud A, Kalita J, Bensaoud M. A survey of malware detection using deep learning[J]. Machine Learning With Applications, 2024, 16: 100546.

【3】Owoh N, Adejoh J, Hosseinzadeh S, et al. Malware detection based on API call sequence analysis: a gated recurrent unit–generative adversarial network model approach[J]. Future Internet, 2024, 16(10): 369.

【201】Mihai CHRISTODORESCU, S. JHA. Static Analysis of Executables to Detect Malicious Patterns[J]. , 2003. DOI:10.21236/ADA449067.

【202】M. SCHULTZ, E. ESKIN, E. ZADOK, et al. Data mining methods for detection of new malicious executables[J]. Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, 2001. DOI:10.1109/SECPRI.2001.924286.

【203】Corey MALONE, M. ZAHRAN, R. KARRI. Are hardware performance counters a cost effective way for integrity checking of programs[J]. 2011. DOI:10.1145/2046582.2046596.

### 机器学习技术检测

Schultz 等人【202】首次利用机器学习技术检测恶意软件，他们提出了一种数据挖掘框架，基于朴素贝叶斯等多个分类器进行对比实验验证。该框架能够准确、自动地检测新的、此前未见过的恶意可执行文件，并且使新恶意可执行文件的当前检测率提高了一倍多。此研究为利用数据挖掘技术进行恶意软件检测奠定了基础。

基于手工特征的传统机器学习算法在恶意PE检测中应用广泛。常见方法包括支持向量机（SVM）、随机森林（Random Forest）、K最近邻（KNN）、朴素贝叶斯、逻辑回归等。例如，Yousuf等【1】对静态提取的PE头特征、节区特征和导入函数特征分别训练了决策树、随机森林、SVM、KNN等模型，并将结果进行集成，最终获得了99.5%的检测率。Shokouhinejad等【4】基于PE的图结构特征，通过构建程序的控制流图（CFG）、函数调用图（FCG）或系统调用图（SCG）等，反映代码的执行逻辑，将CFG描述的程序执行路径作为图数据输入GNN模型，以捕获恶意行为中的结构性模式。Transformer等新型深度模型开始应用于恶意软件分析领域。Kunwar等【15】指出，Transformer模型由于能够捕捉高维数据中的时空与结构特征，在处理原始二进制、反汇编代码、图形和图像等多种表示形式时显示出潜力。

在微结构特征级别，Corey Malone 等人（2011）【203】探讨了硬件性能计数器（HPC）在程序完整性检查中的成本效益。他们的初步研究结果证实，HPC 能够以极低的成本高效地检测程序的修改情况。Yubin Xia 等人（2012）【208】提出了 CFIMon 系统，这是首个非侵入式的系统，可在不修改应用程序或使用专用硬件的情况下，检测并分析各种违反控制流完整性的攻击。该系统利用HPC来实现控制流完整性检测，为控制流完整性检测技术提供了新的思路和方法。J. Demme 等人（2013）【209】研究了利用现有HPC在硬件中构建恶意软件检测器的可行性。他们发现，HPC的数据可用于识别恶意软件，并且检测技术对恶意软件程序的微小变化具有较强的鲁棒性。这为在线恶意软件检测提供了一种潜在的有效方法，有助于及时发现和防范恶意软件的攻击。Sayadi 等人（2018）【206】提出了一种轻量级的定制化硬件辅助恶意软件检测（HMD）方法，该方法利用HPC的特性，通过定制化的机器学习算法结合硬件辅助，能够在运行时有效地检测并进一步分类各种恶意软件类别，使得检测过程更加高效和精准，从而提高了恶意软件检测的准确性。这种结合硬件特性的数据挖掘检测方法为恶意软件检测提供了新的思路。

机器学习的优点是训练速度快、解释性较好，但对特征工程依赖较大，且面对大规模数据和复杂模式时可能力不从心。

【4】Shokouhinejad H, Razavi-Far R, Mohammadian H, et al. Recent advances in malware detection: Graph learning and explainability[J]. arXiv preprint arXiv:2502.10556, 2025.

【15】Kunwar P, Aryal K, Gupta M, et al. SoK: Leveraging Transformers for Malware Analysis[J]. arXiv preprint arXiv:2405.17190, 2024.

【203】 M. SCHULTZ, E. ESKIN, E. ZADOK, et al. Data mining methods for detection of new malicious executables[J]. Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, 2001. DOI:10.1109/SECPRI.2001.924286.

【204】Corey MALONE, M. ZAHRAN, R. KARRI. Are hardware performance counters a cost effective way for integrity checking of programs[J]. 2011. DOI:10.1145/2046582.2046596.

【206】H. SAYADI, Hosein Mohammadi MAKRANI, Onkar RANDIVE, et al. Customized Machine Learning-Based Hardware-Assisted Malware Detection in Embedded Devices[J]. 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018. DOI:10.1109/TrustCom/BigDataSE.2018.00251.

【208】 Yubin XIA, Yutao LIU, Haibo CHEN, et al. CFIMon: Detecting violation of control flow integrity using performance counters[J]. IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), 2012. DOI:10.1109/DSN.2012.6263958.

【209】J. DEMME, Matthew MAYCOCK, J. SCHMITZ, et al. On the feasibility of online malware detection with performance counters[J]. Proceedings of the 40th Annual International Symposium on Computer Architecture, 2013. DOI:10.1145/2485922.2485970.

### 集成学习技术检测

集成学习在恶意软件检测中展现出了显著的优势，它能够结合多个弱学习器的优势，提高检测的准确性和效率，为网络安全提供更可靠的保障。Pengbin Feng等人（2018）【101】提出了一种有效的动态分析框架EnDroid，旨在基于多种类型的动态行为特征实现高精度的恶意软件检测。该研究发现Stacking集成学习方法在Android恶意软件检测中实现了最佳的分类性能，具有很大的应用潜力。其研究方法是通过构建基于多种动态行为特征的检测框架，对比不同集成学习方法的分类效果。该研究的理论贡献在于为动态Android恶意软件检测提供了一种新的有效框架和方法，证明了Stacking在该领域的优势。

T. To等人（2023）【102】引入了MalDEX，这是一种新颖的可解释黑盒恶意软件检测系统。该系统利用从Windows可移植可执行（PE）文件中提取属性，并采用包括单算法、同质集成算法和异质集成算法等一系列分类器来确定恶意软件的存在。这种方法的优势在于能够提供对检测结果的解释，增强了检测系统的可信度和可理解性。其研究方法是通过属性提取和多种分类器的组合来构建检测系统。理论贡献在于为可解释的恶意软件检测提供了一个新的系统架构和方法。但该研究可能面临在不同类型恶意软件上解释效果不一致的问题，对于一些复杂恶意软件的解释能力可能有限。

Azeez等人（2021）【103】提出了一种基于集成学习的Windows PE恶意软件检测方法。该方法通过七个神经网络的集成，并使用ExtraTrees分类器作为最终阶段的分类器来实现恶意软件检测。这种集成多种分类器的方法能够综合不同分类器的优势，提高检测的准确性和可靠性。该研究展示了集成学习在Windows PE恶意检测中的具体应用方式，为实际检测工作提供了一种可行的解决方案。

Wei Li等人（2021）【104】提出了一种基于静态特征和集成学习的恶意代码检测方法。该方法提取恶意代码样本PE头文件中的信息作为静态特征，并使用堆叠集成学习构建恶意代码检测模型。这种方法利用静态特征的稳定性，结合集成学习的优势，提高了恶意代码检测的准确性。其研究方法是通过提取特定的静态特征和采用堆叠集成学习的方式构建模型。理论贡献在于为恶意代码静态特征检测提供了一种新的有效方法。然而，该方法可能存在对动态变化的恶意代码适应性不足的问题，对于一些通过动态加载方式隐藏自身的恶意代码检测效果可能不佳。

T. To等人（2023）【105】提出了一种突变系统，通过结合生成对抗网络（GANs）和强化学习（RL）模型来对抗基于集成学习的检测器，克服了MalGAN模型的局限性，并解决了对Windows可移植可执行（PE）恶意软件进行对抗攻击的三个关键挑战。该研究关注到了集成学习恶意软件检测器面临的对抗样本威胁，并提出了相应的应对策略。其研究方法是通过结合GANs和RL模型构建突变系统。理论贡献在于为解决集成学习恶意软件检测器的对抗攻击问题提供了新的思路和方法。但该研究可能在实际应用中面临计算资源消耗大、对抗样本生成的泛化性不足等问题。

To等人（2023）【105】的研究聚焦于对抗样本对基于集成学习的Windows PE恶意检测效果的影响。该研究提出了一个结合生成对抗网络（GANs）和强化学习（RL）模型的突变系统，以对抗基于集成学习的检测器。此研究克服了MalGAN模型的局限性，并解决了对Windows可移植可执行（PE）恶意软件进行对抗攻击的三个关键挑战。这表明对抗样本可能会对基于集成学习的Windows PE恶意检测产生显著干扰，而有效的对抗样本生成系统能够针对检测模型的弱点进行攻击，从而降低检测效果。

在基于树的集成机器学习方法应用于PE文件恶意分类的研究中，不同学者进行了深入探索。V. Atluri (2019) 【106】研究了六种不同的基于树的集成机器学习技术，这些技术能够在Windows可移植可执行（PE）文件中识别恶意软件，并分析了在识别恶意软件中重要的特征。

N. Azeez等人 (2021) 【103】提出了一种基于集成学习的恶意软件检测方法，该方法通过七个神经网络的集成以及ExtraTrees分类器作为最终阶段分类器来完成检测，虽然这里并非单纯基于树的集成，但体现了集成学习在PE文件恶意检测中的应用思路。综合来看，基于树的集成机器学习方法在PE文件恶意分类中具有一定的潜力，然而，当前的研究仍存在一些局限性。例如，对于不同树集成方法在复杂PE文件环境下的性能稳定性还缺乏更深入的研究，且尚未形成统一的标准来评估这些方法在实际应用中的效果。

此外，集成学习在网络攻击中的特征检测也提供了多种方法。在VPN加密流量识别方面，Jie Cao等人（2022）【107】提出了一种基于集成学习的VPN加密流量识别方法。该方法基于特征冗余和数据类不平衡的最优贝叶斯，研究揭示此方法在识别加密流量时表现出更高的准确性。这一成果为VPN加密流量的准确识别提供了新的途径，有助于更好地监控和管理网络流量，保障网络安全。在DDoS攻击检测中，A. Alashhab等人（2024）【108】设计了一种集成在线机器学习模型，旨在增强SDN环境中的DDoS检测和缓解能力。该模型为SDN环境中的主动DDoS威胁识别和缓解奠定了坚实的基础。通过在线学习的方式，能够实时适应网络环境的变化，及时发现并应对DDoS攻击，提高了网络的抗攻击能力和稳定性。在恶意URL检测方面，N. P. Mankar等人（2024）【109】进行了不同机器学习模型的比较评估。他们的研究通过比较评估，证明了使用集成机器学习进行自动恶意URL检测的可行性。研究得出结论，基于树的模型能够有效地识别威胁，从而加强网络防御。这一研究为恶意URL检测提供了新的思路和方法，有助于提高网络安全防护能力。

[101] Pengbin FENG, Jianfeng MA, Cong SUN, et al. A Novel Dynamic Android Malware Detection System With Ensemble Learning[J]. IEEE Access, 2018. DOI:10.1109/ACCESS.2018.2844349.

[102] T. TO, Hien Do HOANG, Phan The DUY, et al. MalDEX: An Explainable Malware Detection System Based on Ensemble Learning[J]. 2023 International Conference on Multimedia Analysis and Pattern Recognition (MAPR), 2023. DOI:10.1109/MAPR59823.2023.10288922.

[103] N. AZEEZ, Oluwanifise Ebunoluwa ODUFUWA, S. MISRA, et al. Windows PE Malware Detection Using Ensemble Learning[J]. Informatics, 2021. DOI:10.3390/informatics8010010.

[104] Wei LI, Chenyi ZHANG, Jie-Gui ZHOU, et al. Malicious Code Detection Method Based on Static Features and Ensemble Learning[J]. Journal of Physics: Conference Series, 2021. DOI:10.1088/1742-6596/2010/1/012165.

[105] T. TO, Danhee KIM, Do Thi Thu HIEN, et al. On the Effectiveness of Adversarial Samples against Ensemble Learning-based Windows PE Malware Detectors[J]. Int. J. Inf. Sec., 2023. DOI:10.48550/arXiv.2309.13841.

[106] V. ATLURI. Malware Classification of Portable Executables using Tree-Based Ensemble Machine Learning[J]. 2019 SoutheastCon, 2019. DOI:10.1109/SoutheastCon42311.2019.9020524.

[107] Jie CAO, Xing-Liang YUAN, Ying CUI, et al. A VPN-Encrypted Traffic Identification Method Based on Ensemble Learning[J]. Applied Sciences, 2022. DOI:10.3390/app12136434.

[108] A. ALASHHAB, Mohd Soperi Mohd ZAHID, Babangida ISYAKU, et al. Enhancing DDoS Attack Detection and Mitigation in SDN Using an Ensemble Online Machine Learning Model[J]. IEEE Access, 2024. DOI:10.1109/ACCESS.2024.3384398.

[109] N. P. MANKAR, Prashant E SAKUNDE, Sachin Y. ZURANGE, et al. Comparative Evaluation of Machine Learning Models for Malicious URL Detection[J]. 2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon), 2024. DOI:10.1109/MITADTSoCiCon60330.2024.10575452.

## 相关数据集

研究者使用多种公开数据集来训练和评估恶意PE检测模型。VirusShare是一个恶意软件样本存储库，截至2025年已有超过9,800万条恶意样本。

【300】Virusshare. System currently contains 98,931,046 malware samples. <https://virusshare.com[DB/OL>], [2025-04-07].

EMBER数据集（Elastic Malware Benchmark）提供了大规模的PE特征集合，其中EMBER2017包含约110万份样本，EMBER2018包含约100万份样本【301】。该数据集不仅包含静态特征，还包括恶意/良性标注，用于基准测试机器学习模型【302】。

【301】Elastic. Elastic Malware Benchmark for Empowering Researchers. [https://github.com/elastic/ember[DB/OL](https://github.com/elastic/ember%5bDB/OL)], [2025-04-07].

【302】Anderson H S, Roth P. Ember: an open dataset for training static pe malware machine learning models[J]. arXiv preprint arXiv:1804.04637, 2018.

微软恶意软件分类数据集（Microsoft Malware Classification Challenge，BIG 2015）包含约0.5 TB的恶意样本，以字节码和反汇编形式提供，约有2万多个样本，用于家族分类任务【304】。

【304】Microsoft. Microsoft Malware Classification Challenge <https://paperswithcode.com/dataset/microsoft-malware-classification-challenge>[DB/OL], [2025-04-07].

此外，还有经典图像化数据集Malimg，包含9,339个样本，25个恶意家族【305】。

【305】L. et al. Malware Images:Visualization and Automatic Classification. [https://paperswithcode.com/dataset/malimg[DB/OL](https://paperswithcode.com/dataset/malimg%5bDB/OL)], [2025-04-07].

这些数据集为研究者提供了丰富的训练和测试平台。

## 论文主要工作

本研究围绕基于集成学习的恶意PE软件特征检测与识别展开，主要工作如下：

1) 深入分析恶意PE软件的常见特征，包括PE数据结构、直方图特征等，并系统研究集成学习技术（如Stacking、OOF、加权融合）及特征哈希技术在恶意软件检测中的应用，为后续研究奠定理论基础。

2) 提出一种基于多维分析的样本静态特征检测方案，涵盖PE数据结构特征解析、直方图特征模型检测、字符规则匹配特征等，并通过特征哈希向量化和特征融合技术整合多源特征，生成统一的高维特征向量，为恶意软件检测提供丰富的特征输入。

3) 设计并实现基于多模型融合的恶意PE识别模型，包括基于CNN的直方图特征识别模型、基于堆叠集成的静态PE特征识别模型以及基于LightGBM的特征工程模型。通过多层集成学习方案优化模型性能，提升检测精度和泛化能力。

4) 开发基于集成学习的恶意PE识别系统，采用React框架构建前端界面，FastAPI框架搭建后端服务，SQLite数据库存储数据。系统实现样本管理、特征提取、模型推理和数据可视化等功能，通过系统测试验证了其高效性和准确性，为恶意软件检测提供了一种实用化的解决方案。

## 论文组织与结构

本文的组织结构如下：

第一章为绪论，介绍研究背景、意义及国内外研究现状，并阐述论文的主要工作和组织结构。

第二章为技术背景及相关研究，详细介绍了恶意PE软件的常见特征、集成学习技术、特征哈希技术以及系统开发所用技术。

第三章提出基于多维分析的样本静态特征检测方案，涵盖PE数据结构特征解析、直方图特征模型检测和字符规则匹配特征等内容。

第四章构建基于多模型融合的恶意PE识别模型，包括CNN模型、堆叠集成模型和LightGBM模型，并介绍多层集成学习方案。

第五章设计基于Stacking和加权融合的多层集成学习方案，优化模型性能。

第六章实现基于集成学习的恶意PE识别系统，介绍系统需求分析、总体设计、功能实现和系统测试。第七章为总结与展望，总结研究成果并提出未来研究方向。

# 技术背景及相关研究

## 恶意PE软件常见特征

### PE数据结构

PE（Portable Executable）数据结构是Windows操作系统下可执行文件（如EXE、DLL等）的标准格式，其设计用于支持多种硬件平台和操作系统版本。PE文件结构主要由DOS头、PE头、节表、节数据等部分组成。

PE文件的起始部分是DOS头其包含一些基本的MS-DOS可执行文件信息，最重要的是e\_lfanew字段，该字段指向PE头的偏移量。PE头是PE文件的核心部分，其以4字节的“PE\0\0”签名开始，随后是文件头和可选头。文件头包含文件的物理层信息及属性，如运行平台、区块数目、创建时间等。可选头则提供了关于程序运行时的详细信息，如入口点地址、映像基址、堆栈和堆的大小等。在PE头之后是节表，它定义了文件中各个节的属性和位置，这些节分别存放程序的代码、初始化数据、资源等，如图 2‑1所示。

PE数据结构的这种设计使得Windows系统能够高效地加载和执行可执行文件，同时也为程序的调试、资源管理和动态链接提供了支持[13]。

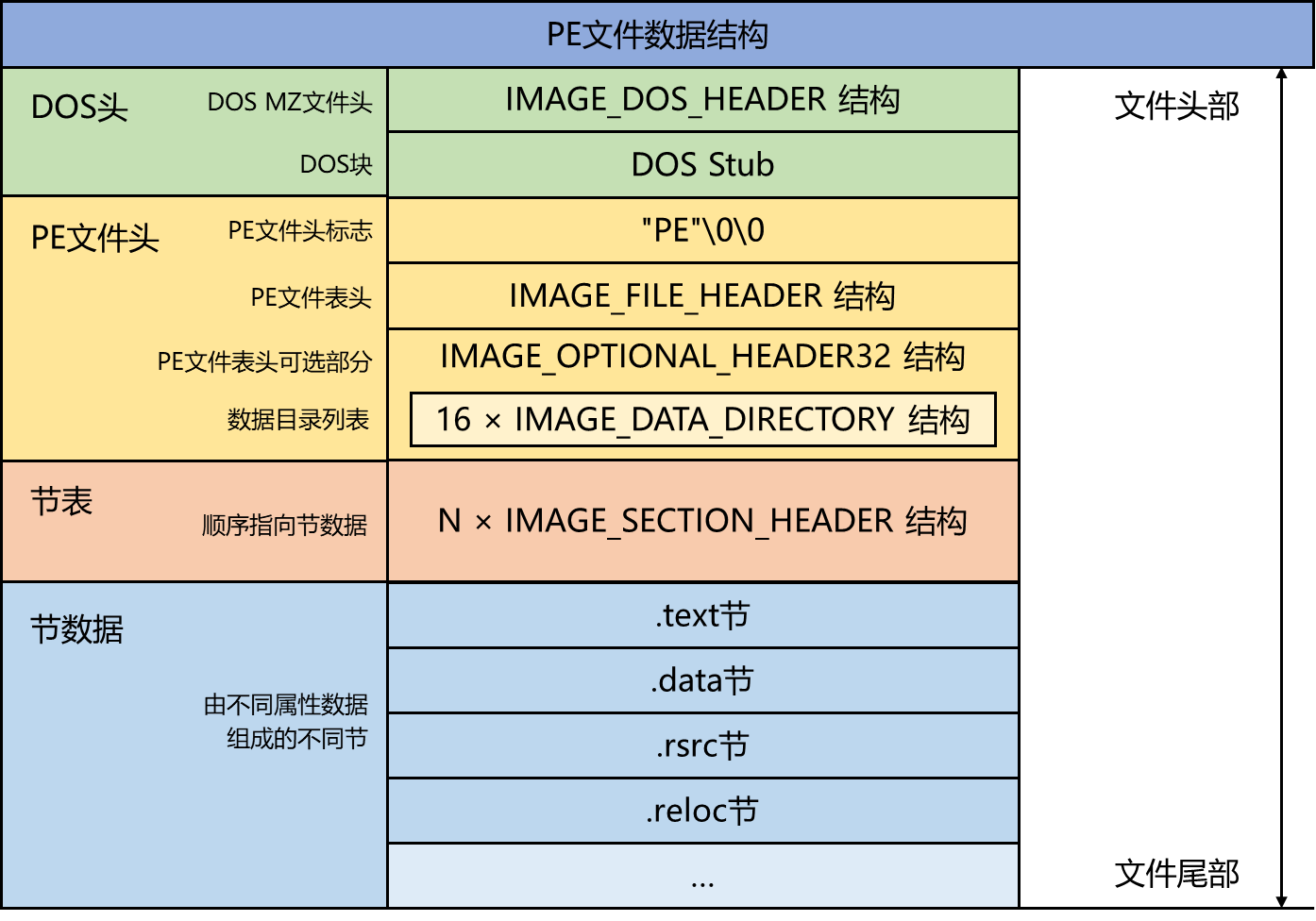


图 2‑1 PE数据结构示意图

### 直方图特征

本文中，采用了直方图特征来对恶意软件进行检测，这一方法受到了文献《Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features》[14]的启发。该文献提出了一种基于深度神经网络的恶意软件检测方法，通过融合字节直方图和字节熵直方图来提取二进制程序的特征，并取得了显著的检测效果。本文借鉴了这一思路，设计并实现了类似的特征提取方法，以提升模型的检测能力。

字节直方图（Byte Histogram）和字节熵直方图（Byte Entropy Histogram）是两种互补的静态特征。字节直方图通过统计每个字节值（范围为0到255）在二进制文件中出现的次数，生成一个256维的向量，反映了文件中字节值的整体分布情况。这种特征能够有效检测文件中是否存在异常的字节分布模式，例如加密或填充数据。

字节熵直方图通过滑动窗口的方式计算得到。例如在二进制文件上滑动一个大小为2048字节的窗口，步长为1024字节，对于每个窗口，计算其熵值，熵值的计算公式如式(2-1)所示，共需计算 次，如图 2‑2所示。

|  |  |  |
| --- | --- | --- |
|  |  | (2-1) |

其中，表示字节值 在当前窗口中出现的概率。通过统计滑动窗口的字节值、熵值对数据，最终将这些对转换成一个256维的特征向量。字节熵直方图能够捕捉到文件中局部区域的熵值变化情况，这对于识别代码段、混淆区域或加密数据具有重要作用。

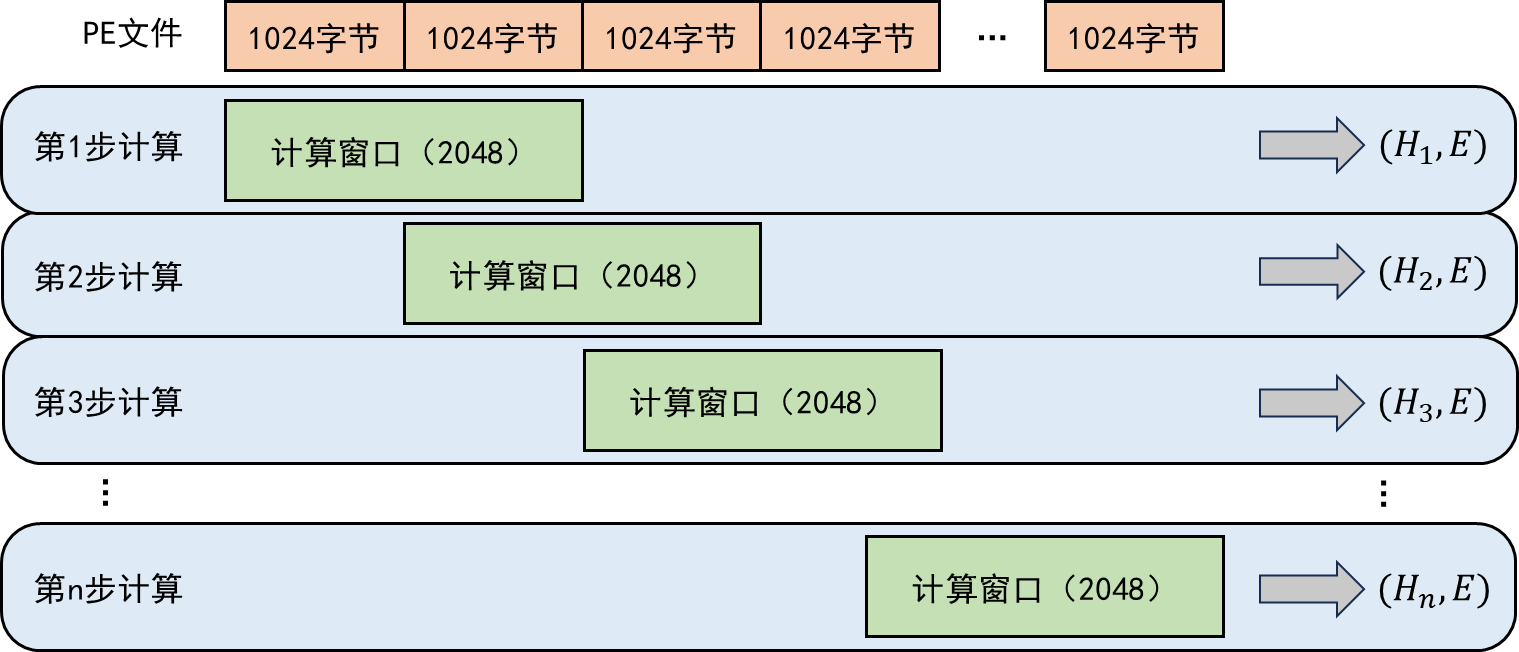


图 2‑2滑动窗口计算字节熵

## 集成学习技术

### Stacking

Stacking，即堆叠技术，是一种强大的集成学习方法，它通过将多个基学习器的预测结果作为输入，训练一个高级的元学习器来优化最终的预测结果[15]。其核心在于利用不同基学习器的自身优势，通过元学习器来校正基模型的系统性偏差，以最小化最终的泛化误差，从而提高整体模型的性能【3】。

在Stacking中，首先需要选择多个不同的基学习器，这些学习器可以是不同类型或参数的模型，例如决策树、支持向量机、神经网络等。每个基学习器独立地对训练数据进行学习，并生成对训练数据的预测结果。这些预测结果被用作新的特征，与原始训练数据的标签一起，构成了一个新的“元数据集”。在这个元数据集上，训练一个元学习器，通常是简单的线性模型，如逻辑回归或线性回归。元学习器的任务是学习如何最优地组合基学习器的预测结果，以生成最终的预测。

Stacking的一个关键优势在于其能够充分利用不同基学习器的预测结果，从而捕捉到数据中更复杂的关系。通过这种方式，Stacking可以有效提高模型的泛化能力和预测精度。此外，由于元学习器通常是简单的线性模型，因此Stacking在模型解释性方面也具有一定的优势。

Stacked Generalization（或称 Stacking）是一种二级学习框架，其核心在于通过“元学习器”来校正基模型的系统性偏差，以最小化最终的泛化误差【3】。该方法首先将训练集分为若干折，对每个基模型分别在折外数据上生成预测，再将所有基模型的折外预测结果合并为新的训练特征，最后由元学习器（通常为线性回归、逻辑回归或更复杂的模型）进行二次学习【4】。Wolpert 最初提出该框架时，即证明了通过该两阶段结构可以减少集成中因各基模型偏差不同所带来的误差累积，从而获得更稳健的预测效果【3】。随后的大量实证研究表明，Stacking 在分类和回归任务中均能超越简单投票或平均的集成方法，尤其在基模型多样性较高时优势更为显著【4】。

### OOF

OOF（Out-Of-Fold）技术，即“折外预测”技术，是一种在机器学习中用于评估模型泛化性能和构建集成模型的重要方法[16]。它通常与交叉验证结合使用，通过在交叉验证的每一折中保留一部分数据作为测试集，从而生成模型的“折外预测”结果。

在OOF技术中，数据集首先被划分为多个子集（通常称为“折”）。对于每个折，模型在训练集上进行训练，并在对应的测试集（即该折的数据）上进行预测。这些预测结果被称为“折外预测”，因为它们是在模型未参与训练的数据上生成的。通过这种方式，每个数据点最终都会有一个对应的预测值，这些预测值可以用来评估模型的性能。

OOF技术的一个重要应用是构建集成模型，特别是Stacking（堆叠）模型。在Stacking中，多个基学习器的预测结果被用作新的特征，而元学习器则基于这些特征进行训练。OOF预测结果为元学习器提供了丰富的输入，使得元学习器能够学习如何最佳地组合基学习器的预测结果，从而提高整体模型的性能。

OOF技术的优势在于它能够充分利用交叉验证的思想，确保每个数据点都参与到模型的训练和测试过程中，同时避免了过拟合的问题。通过这种方式，OOF不仅能够提供对模型泛化能力的准确估计，还能够为构建更强大的集成模型提供支持。

### 加权融合

加权融合是一种简单而有效的集成学习方法，其核心思想是将多个模型的预测结果进行加权平均，以形成一个新的预测结果。在实际应用中，每个模型的权重可以通过交叉验证等方法来确定，从而确保权重分配的合理性。这种方法简单直观，易于实现，且能够有效提高预测的准确率。

加权融合的一个关键点在于模型的独立性。为了获得更好的融合效果，参与融合的各个模型应具有一定的独立性，这样可以减少模型之间的相关性，从而降低整体模型的方差。此外，加权融合对异常值较为敏感，因此在处理包含异常值的数据时需要格外小心。

加权融合适用于各种机器学习任务，尤其是在数据集较小或特征维度较高时，通过结合多个模型的预测结果，可以显著提高预测的准确率。这种方法在实际应用中非常灵活，可以根据具体任务的需求调整模型的权重，以达到最佳的预测效果。

## 机器学习模型介绍

### 卷积神经网络（CNN）

卷积神经网络（Convolutional Neural Network, CNN）是一种具有局部连接和权值共享特性的深度学习模型，可自动学习空间或时序数据的多层次特征表示【1】。该网络通过卷积层（convolutional layer）提取局部特征、池化层（pooling layer）进行下采样、全连接层（fully connected layer）整合全局信息，并利用反向传播算法优化网络参数【2】。CNN自LeNet-5以来，在ImageNet等大规模图像分类竞赛中表现卓越，成为图像识别、目标检测、语义分割等任务的基石算法【3】。此外，深度学习权威教材《Deep Learning》指出，CNN的层次化结构使其能够在参数可控的前提下捕获输入的局部不变性与平移不变性，显著提升模型的泛化能力【4】。

### 随机森林（Random Forest）

随机森林是一种基于Bagging（Bootstrap Aggregation）思想的集成学习方法，由Leo Breiman于2001年提出，其核心是在不同的样本子集和特征子集上构建多棵决策树，然后通过多数投票（分类）或平均（回归）汇总各树输出，以降低单棵决策树过拟合带来的方差【5】。在每次节点分裂时，随机森林不仅对样本进行有放回抽样，还从所有特征中随机选取一部分用于候选分裂，从而进一步减少树间相关性并增强模型稳定性。理论分析和实验结果均表明，随着树的数量增多，森林的泛化误差将收敛至一个常数，且通常优于单棵决策树的性能。然而，随机森林的集成结构降低了模型的可解释性，并且在特征高度相关或类别分布严重不平衡的场景中，性能提升可能受到限制【6】。

### LightGBM

LightGBM（Light Gradient Boosting Machine）是一种高效的GBDT实现框架，由Ke等人于2017年提出，旨在在保持与传统GBDT相近准确度的同时大幅加速训练过程【7】。针对大规模数据和高维特征的挑战，LightGBM引入了梯度单边采样（Gradient-Based One-Side Sampling, GOSS）策略，通过保留梯度较大的样本并随机舍弃小梯度样本来减少训练数据量，同时利用独占特征捆绑（Exclusive Feature Bundling, EFB）技术将稀疏且互斥的特征合并以降低特征维度。该框架采用基于直方图的决策树学习算法，并支持叶子优先（leaf-wise）的树结构生长策略，显著提升了时间和内存效率【8】。实验证明，LightGBM在多种公共数据集上训练速度可较传统GBDT实现提升超过20倍，同时在分类和回归任务中维持相当或更优的准确率【7】。

【1】LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

【2】Yamashita R, Nishio M, Do R K G, et al. Convolutional neural networks: an overview and application in radiology[J]. Insights into imaging, 2018, 9: 611-629.

【3】Zhao X, Wang L, Zhang Y, et al. A review of convolutional neural networks in computer vision[J]. Artificial Intelligence Review, 2024, 57(4): 99.

【4】Goodfellow I, Bengio Y, Courville A, et al. Deep learning[M]. Cambridge: MIT press, 2016.

【5】Breiman L. Random forests[J]. Machine learning, 2001, 45: 5-32.

【6】Random Forest. Wikipedia[EB/OL], last modified 2025-03.

【7】Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[J]. Advances in neural information processing systems, 2017, 30.

【8】LightGBM. Wikipedia[EB/OL], last modified 2025-01.

## 特征哈希技术

特征哈希（Feature Hashing）是一种强大的技术，能够将非结构化数据高效地转化为固定维度的数值向量[17]。它特别适合处理大规模数据集中的文本、类别型数据等，这些数据通常难以直接用于机器学习模型。通过特征哈希，我们可以将这些复杂的数据类型转化为模型能够理解的数值形式，同时避免了传统方法中需要预先构建词表的繁琐步骤。

特征哈希的核心在于它能够动态处理新出现的特征。例如，在处理程序的节区信息时，可能会遇到新的节区名称或属性。传统的特征处理方法可能需要重新构建词表或调整模型结构，但特征哈希技术可以即时处理这些新特征，无需额外的预处理步骤。这种灵活性使得特征哈希非常适合在线学习和动态数据环境。

此外，特征哈希技术还具有内存友好的特性。它不需要存储庞大的词表，从而大大减少了内存的使用量。这对于处理大规模数据集尤为重要，因为传统的词表可能会占用大量的内存资源。通过特征哈希，我们可以将不定长的特征信息压缩为固定维度的向量，这不仅提高了数据处理的效率，还使得数据能够更好地适配机器学习模型的输入要求。

特征哈希技术的这些特性使其成为处理非结构化数据的理想选择。它不仅能够高效地将数据转化为数值形式，还能够动态处理新特征，减少内存使用，并支持在线学习。这些优点使得特征哈希在实际应用中表现出色，特别是在处理大规模数据集和动态数据环境时。

【1】Weinberger K, Dasgupta A, Langford J, et al. Feature hashing for large scale multitask learning[C]//Proceedings of the 26th annual international conference on machine learning. 2009: 1113-1120.

## 余弦退火技术

余弦退火学习率调度（Cosine Annealing）最早在“Stochastic Gradient Descent with Warm Restarts”（SGDR）一文中由Loshchilov与Hutter于2016年提出【4】。该方法通过将训练轮次 映射到学习率 ，如式所示。

在每个周期内呈现余弦曲线下降，当达到周期末端时可选择“重启”（warm restart），将学习率恢复到 以探索新的优化区域【4】。这种调度方式兼顾了初始的高探索能力与后期的稳定收敛，有效减少了过拟合并提升了收敛速度。实验证明，余弦退火不仅在图像分类任务（如CIFAR-10/100）上取得了新的最优结果，而且结合模型快照可用于深度集成以进一步提高泛化性能【5】【6】

【4】Loshchilov I , Hutter F .SGDR: Stochastic Gradient Descent with Warm Restarts[J]. 2016.DOI:10.48550/arXiv.1608.03983.

【5】Huang G, Li Y, Pleiss G, et al. Snapshot ensembles: Train 1, get m for free[J]. arXiv preprint arXiv:1704.00109, 2017.

【6】AWS Labs. Advanced Learning Rate Schedules: Cosine Annealing. GitHub Tutorial[EB/OL], [2025-04-07].

## Fast Gradient Sign Method (FGSM)

Fast Gradient Sign Method（FGSM）是一种基于模型梯度的对抗样本生成方法，最早由Goodfellow等人在2015年提出。该方法在保持原始输入 𝑥 与其真实标签 𝑦 不变的前提下，通过一次前向和反向传播计算损失函数xx关于输入的梯度xx并按照梯度符号方向对输入施加微小扰动，如式所示。

其中 为扰动幅度， 取梯度的符号【1】。该方法的核心思想是利用神经网络对输入线性响应的脆弱性，通过最快速的单步扰动即可大幅降低模型准确率。实验表明，FGSM生成的对抗样本在多种模型和数据集（如ImageNet）上均能有效欺骗目标模型，且计算开销仅为一次梯度反向传播，因此在对抗研究和鲁棒性评估中广泛应用【2】【3】。

【1】Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.

【2】Papernot N, McDaniel P, Goodfellow I, et al. Practical black-box attacks against machine learning[C]//Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017: 506-519.

【3】TensorFlow. Adversarial example using FGSM. TensorFlow Tutorial[EB/OL], [2025-04-07].

## 特征降维技术

### 卡方关联性分析

### 主成分分析（Principal Component Analysis, PCA）

主成分分析（PCA）由Pearson于1901年首次提出，并由Hotelling在1933年进一步形式化【7】【8】。PCA的目标是在保证数据总方差损失最小的条件下，将 𝑝 维原始变量线性映射到少数 维正交空间。具体而言，PCA通过对数据协方差矩阵进行特征分解，取前 个最大特征值对应的特征向量作为主成分基，变换后样本在这些基上的投影即为主成分。PCA既可用于去除变量间冗余相关性，也能保留数据的主要差异信息，是经典且高效的多元统计分析方法【9】。

【7】Pearson K. LIII. On lines and planes of closest fit to systems of points in space[J]. The London, Edinburgh, and Dublin philosophical magazine and journal of science, 1901, 2(11): 559-572.

【8】Hotelling H. Analysis of a complex of statistical variables into principal components[J]. Journal of educational psychology, 1933, 24(6): 417.

【9】Abdi H, Williams L J. Principal component analysis[J]. Wiley interdisciplinary reviews: computational statistics, 2010, 2(4): 433-459.

### PCA降维技术

在实际应用中，PCA经常被用作降维技术，以缓解“维度灾难”和特征冗余问题。其核心思路是保留累计方差贡献度达到一定阈值（如95%）的前 个主成分，并舍弃剩余分量，从而将数据维度从 降至 。这种降维方式不仅可极大降低后续模型的计算复杂度，还能抑制噪声和过拟合。Jolliffe的专著对PCA及其在信号处理、图像识别等领域的扩展应用进行了系统论述，指出PCA降维在大规模数据预处理和特征工程中仍具不可替代的价值【10】【11】。

【10】Shlens J. A tutorial on principal component analysis[J]. arXiv preprint arXiv:1404.1100, 2014.

【11】IBM. What Is Principal Component Analysis (PCA)? IBM Think[EB/OL], [2025-04-07].

## 系统开发所用技术

### React框架

React 是一个用于构建用户界面的开源 JavaScript 库，由 Facebook 维护[18]。它专注于构建单页应用中的视图层，允许开发者使用声明式编程范式来创建交互式的、可复用的 UI 组件。React 的设计理念是让构建复杂的用户界面变得简单而高效，通过将界面拆分为独立的组件，每个组件负责渲染一部分用户界面，从而提高开发效率和代码的可维护性。

React 的核心特性之一是虚拟 DOM。当组件的状态发生变化时，React 会重新渲染组件的虚拟 DOM，然后通过一个高效的算法比较新旧虚拟 DOM 的差异，并将这些差异应用到真实的 DOM 上，从而实现界面的更新。这种机制不仅提高了性能，还减少了直接操作 DOM 所带来的复杂性。开发者无需手动操作 DOM，只需关注组件的状态和渲染逻辑，React 会自动处理界面的更新。

React 的另一个重要特性是组件化。组件是 React 应用的基本构建块，可以是类组件或函数组件。类组件通过继承 React.Component 类来定义，而函数组件则是简单的 JavaScript 函数，通过返回 JSX 来描述组件的 UI。组件可以包含自己的状态和生命周期方法，这些方法允许开发者在组件的不同阶段执行特定的操作，例如在组件加载时获取数据或在组件卸载时清理资源。通过组合不同的组件，可以构建出复杂的用户界面。

React 的声明式编程范式和高效的 DOM 更新机制使其成为构建现代 Web 应用的理想选择。它不仅提高了开发效率，还通过组件化和状态管理机制增强了代码的可维护性和可扩展性。React 的强大生态系统也为开发者提供了丰富的资源和工具，从而能够满足各种开发需求。无论是小型项目还是大型企业级应用，React 都能够提供灵活且高效的解决方案。

### FastAPI框架

FastAPI 是一个现代、快速（高性能）的 Python Web 框架，专为构建 API 而设计，基于标准的 Python 类型提示，使用 Python 3.6 及以上版本构建。它于 2018 年 12 月首次发布，迅速在开发者社区中获得了广泛关注。

FastAPI 的核心特性之一是其高性能。它建立在 Starlette 和 Pydantic 之上，利用异步编程（async/await）提供了出色的性能，与 NodeJS 和 Go 相当，是目前最快的 Python Web 框架之一。此外，FastAPI 自动生成交互式 API 文档，支持 Swagger UI 和 ReDoc，这些文档基于 OpenAPI 标准，使得 API 的理解和测试更加直观。

在数据验证方面，FastAPI 利用 Python 的类型注解和 Pydantic 模型，自动对请求和响应数据进行验证和序列化，确保数据的完整性和一致性。这种基于类型提示的开发方式不仅提高了代码的可读性和可维护性，还减少了人为错误。

FastAPI 还支持依赖注入系统，允许开发者将依赖项注入到请求处理函数中，从而简化代码复用和模块化。此外，它原生支持 WebSocket，适用于实时通信场景。

FastAPI 的异步支持是其另一个重要特性。通过 async/await 语法，FastAPI 能够高效处理大量并发请求，特别适合处理 I/O 密集型任务。这种异步机制使得 FastAPI 在处理高并发请求时表现出色，提升了 API 的响应速度和效率。

### SQLite数据库

SQLite 是一种轻量级的嵌入式关系数据库，以其简单性、高效性和易用性而被广泛应用于各种开发场景。SQLite 的核心设计理念是“零配置”，用户无需进行复杂的安装或管理配置即可在应用程序中直接使用。整个数据库存储在一个单一的跨平台磁盘文件中，这种设计不仅简化了数据管理，还便于数据的传输和备份。

SQLite能够支持标准的 SQL 语言，包括创建表、插入数据、查询、更新和删除等操作。它还支持 ACID（原子性、一致性、隔离性、持久性）事务，确保数据的完整性和一致性，即使在系统崩溃或断电的情况下也能保持数据的可靠性。此外，SQLite 提供了多种高级功能，如触发器、视图、索引以及对 JSON 数据的支持。

在性能方面，它在许多普通数据库操作中比一些流行的客户-服务器模式数据库更快。它的代码占用空间小，完全配置后小于 750 KiB，省略可选功能后可以更小。SQLite 使用 ANSI-C 编写，源代码清晰且注释良好，测试覆盖率高，易于嵌入到各种项目中。

此外，SQLite 的开源性和无服务器架构使其成为一种灵活且经济高效的数据库解决方案。它支持多种操作系统和编程语言，包括 Linux、Windows、Mac OS X、Android、iOS 等。SQLite 的这些特性使其成为开发人员在多种应用场景中的首选数据库之一。

## 本章小结

本章主要对本研究中涉及的相关技术背景及相关研究进行了介绍。主要包括恶意PE软件常见特征、集成学习技术、特征哈希技术以及在系统开发过程中涉及的相关技术。

# 基于多维分析的样本静态特征检测方案

## 特征检测方案概述

本章针对PE文件的多维结构与行为模式，提出了一种系统化的静态特征检测方案，该方案提取的特征包括基于字节分布的直方图特征、基于局部熵分布的熵直方图特征、基于PE数据结构的静态特征、基于规则与正则的字符串匹配特征、基于Opcode片段的函数行为特征等。所有特征均可直接从PE软件样本中直接获取，并以浮点向量形式输出，以供后续分类模型使用。

特征检测整体流程如图3-1所示：首先从原始二进制中并行提取多源特征，然后对字符型与类别型信息进行特征哈希与归一化，最终输出统一的高维特征向量，用于后续的集成学习中。

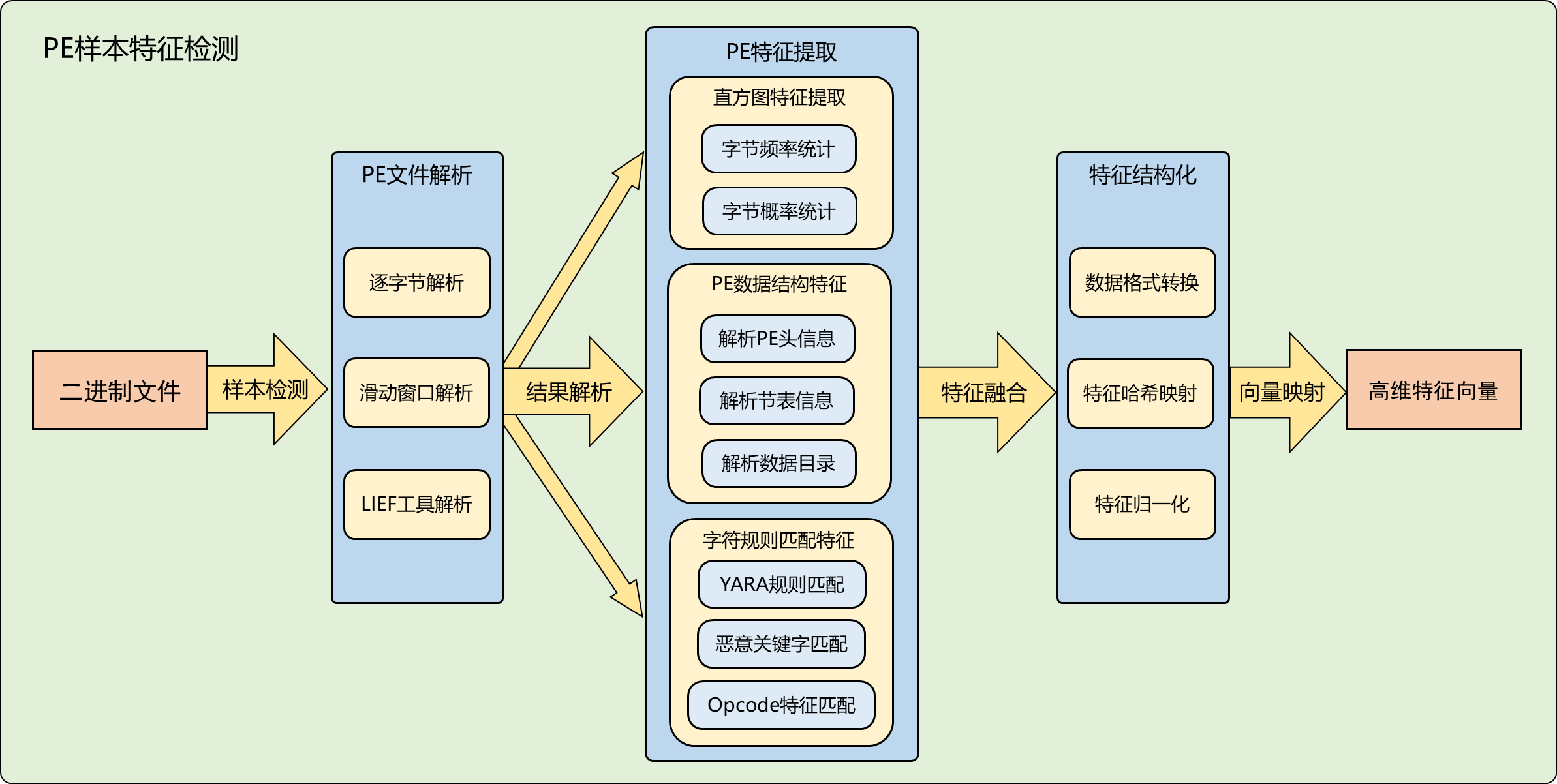


图 3‑1 PE样本特征检测总体流程

## PE数据结构特征解析

在PE格式中，DOS头、PE文件头、数据目录、节表、导入/导出表、资源表、重定位表等数据结构蕴含了丰富的元信息。本节将逐项说明如何从这些结构中提取特征，并分析各结构特征的特点和应用方式。

### PE软件结构概述

PE格式是Windows可执行文件的二进制封装标准，其数据结构主要包括：DOS Header、PE Header、Optional Header、Section Table、Data Directories等部分。其中PE Header中包含文件类型（Machine）、入口点地址、节区对齐方式等信息，Optional Header中包含子系统类型、数据目录偏移等。本文针对其中的关键字段，如TimeDateStamp、Subsystem、SizeOfImage、起始入口点、入口节名等进行数值化统计，以捕捉编译器版本、打包方式、时间戳异动等特征，用于后续的模型训练。

### 各结构特征检测

PE各结构中都或多或少存在一些可检测的特征用于恶意软件的检测，以下是本文主要使用的一些特征，包括其检测方法的具体实现。

#### DOS头及PE文件头

从文件起始位置开始，PE 文件以DOS Header作为引导，其固定长度为 64 字节，在Magic 字段（即文件偏移 0x00）存储着标志性值0x4D5A（即 "MZ"），该字段用于标识文件类型。e\_lfanew 字段（偏移 0x3C）记录 PE Header 在文件中的偏移地址，这是后续结构解析的入口点。

基于 e\_lfanew 指向的位置，可定位到PE Header，该部分长度为 224 字节，包含 Signature（偏移 0x00）、Machine（偏移 0x04）、NumberOfSections（偏移 0x06）等重要字段。Signature 固定为0x50450000（即 "PE\0\0"），用于验证 PE 文件格式；Machine 字段占 2 字节，以特定编码标识目标 CPU 架构（如 0x8664 代表 AMD64）；NumberOfSections 记录节区数量，可用于判断文件是否存在异常添加或删除节区的行为。这些字段均以固定偏移量存储，可通过二进制文件读取操作直接获取检测。

在 PE Header 后即为Optional Header字段，其长度可变，默认大小为 224 字节。该部分包含入口点地址（AddressOfEntryPoint，偏移 0x74）、节区对齐方式（SectionAlignment，偏移 0x28）、子系统类型（Subsystem，偏移 0x50）等重要信息。例如，入口点地址总是指向程序执行的起始位置，若该地址指向异常区域，可能暗示代码注入；子系统类型用于标识程序运行环境（如 0x0003 表示 Windows 图形界面程序），异常的子系统类型设置可能暴露其恶意伪装方法。

此外，从 DOS Header 中的 Magic、e\_lfanew、UsedBytesInLastPage 等字段提取数值特征，结合 PE Header 中的 Signature、Machine、NumberOfSections 等，能够用于识别非标准编译器或异常捆绑器。例如，标准的 DOS Header Magic 值为0x4D5A（即 "MZ"），若该值异常，则可能表明文件被篡改或格式错误。PE Header 中的 Machine 字段用于标识目标 CPU 架构，当 Machine 字段值为 0x8664（AMD64），但实际代码中存在 x86 指令时，可能表明存在异常编译或逆向工程痕迹。TimeDateStamp 字段记录文件的编译时间戳，若该时间戳与文件声称的发布时间存在较大偏差，或多个文件具有相同的时间戳，都可能是恶意软件批量生成的特征[19]。

#### 节表及节区数据

节表（Section Table）位于 PE Header 和 Optional Header 之后的位置，其结构由一系列节区描述符组成，每个描述符固定长度为 40 字节。节表通过 PE Header 中的 NumberOfSections 字段确定条目数量，每个描述符包含节区名称（Name，偏移 0x00）、虚拟地址（VirtualAddress，偏移 0x0C）、大小（SizeOfRawData，偏移 0x10）、属性（Characteristics，偏移 0x28）等信息。通过遍历节表，可获取每个节区的详细属性，如判断 ".text" 节是否具备可执行权限，".data" 节是否包含异常大的数据量[20]。

对每个节区（section）可以检测到其 Name、VirtualSize、RawDataSize、Entropy、Characteristics 等属性。在正常的 PE 文件中，代码节（".text"）通常设置为可读和可执行（MEM\_READ | MEM\_EXECUTE），数据节（".data"）设置为可读和可写（MEM\_READ | MEM\_WRITE）。然而恶意代码常通过将代码节标记为可读、可写、可执行以实现动态代码注入。例如，某些勒索软件会在运行时向代码节写入加密密钥和勒索信息。通过统计可读、可写、可执行节区的数量与平均大小，以及其资源节区（.rsrc）数量，能够有效反映壳保护与代码混淆强度[21]。

此外，节区的 Entropy（熵值）可以衡量数据的随机性，加密或压缩后的恶意代码节区熵值通常很高，而正常节区熵值相对较低。

#### 关键数据结构

数据目录（Data Directories）位于 Optional Header 之后，包含 16 个固定长度 8 字节的目录项，分别指向导入表、导出表、资源表等关键数据结构的位置。例如，导入表（Import Table）地址由数据目录的第 1 项指定，导出表（Export Table）地址由第 2 项指定。这些数据结构记录了程序调用外部函数、暴露自身功能等信息，通过解析对应地址存储的内容，可提取函数数量、DLL 名称等特征，用于检测恶意代码的异常行为。

导入表与导出表分别统计函数数量、DLL 名称、函数名哈希分布等。在正常程序中，导入表包含的函数调用通常符合其功能需求，如办公软件会调用与文档处理相关的系统函数。而恶意软件可能会调用一些可疑的 API，如用于进程注入的CreateRemoteThread、用于网络通信的WSASocket等。通过分析函数名哈希分布，可以发现异常的函数调用模式。

资源表（Resource Table）中资源类型、资源大小、嵌入脚本等信息也能够用于恶意检测。例如，恶意软件可能会在资源表中嵌入恶意脚本或图标，用于伪装成正常程序。重定位表（Relocation Table）中可以统计条目个数和分布差异，能够用于检测恶意软件的自修改代码、动态加载行为。当重定位表条目过多时，可能表明代码经过混淆或加密处理，需要在运行时进行动态重定位[20]。

#### 其他重要数据结构

调试信息（Debug Directory）字段包含 PDB 路径，反映开发环境。正常的开发过程中，PDB 路径指向合法的调试信息文件，而恶意软件可能会故意保留错误的 PDB 路径，或者不包含 PDB 路径以隐藏其真实来源。

TLS（Thread Local Storage）回调函数指针可指示恶意持久化，一些恶意软件会利用 TLS 回调函数在进程启动时自动执行恶意代码，实现自启动。

WIN数字证书结构（WIN\_CERTIFICATE）可用于判断签名状态及证书异常，未经验证的签名或过期证书都可能存在恶意行为[22]。

### 特征哈希向量化

对于字符串与类别型数据（如节区名称、导出函数名、DLL 名、Section 特性列表等），本文采用特征哈希技术对非结构化数据进行哈希编码。记输入键值对集合，哈希后生成固定维度 的向量，计算公式如式(3-1)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-1) |

其中 基于二次哈希决定正负号，以减轻冲突影响。通过该方式，可将任意新出现的键映射至相同维度，不依赖词典，同时保证在线与离线一致性。例如，对于节区名称 ".text"，哈希后可能映射至第 128 维，并根据其出现次数累加权重。

## 直方图特征模型检测

### 字节直方图特征检测

字节直方图特征提取旨在捕捉 PE 文件中各字节的全局分布模式，恶意PE软件在进行混淆和隐藏时，往往会添加过多的单一混淆代码，具体表现为某字节的分布频率异常，例如，某些恶意软件通过大量插入 0x90（NOP 指令）来混淆代码，在字节直方图中表现为该字节的频率显著高于正常文件，从而可作为检测特征。

字节直方图特征检测的过程主要包括数据统计、归一化与特征融合三个步骤。具体而言：首先，将 PE 文件视为字节序列 ，通过构建 256 维计数器数组 ，逐字节扫描文件并统计每个字节值（0-255）的出现次数，如式(3-2)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-2) |

其中 为指示函数。

当条件成立时返回 1，否则为 0。该过程可在 时间复杂度内完成，有效保留文件的字节频率分布特征。

其次，为消除文件大小对特征的影响，需对计数向量进行归一化处理。本文采用线性归一化方法将计数值映射至 [0, 1] 区间，计算公式如(3-2)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-3) |

其中 表示字节 的归一化频率，该操作使得特征具有尺度不变性，便于后续模型处理。

最后将归一化后的 256 维概率向量与 256 维字节熵直方图特征拼接，形成 512 维输入向量 ，为后续集成学习模型提供全局字节分布信息。

### 字节熵直方图特征检测

字节熵直方图侧重于检测文件局部区域的信息熵分布，通过滑动窗口计算熵值并量化，能够捕捉数据的随机性和复杂性，对恶意软件的加密、压缩等混淆手段较为敏感。

如图 3.2 所示，正常 PE 文件的熵值分布相对均匀，而加密后的恶意代码区域熵值接近最大值 8，在熵直方图中呈现明显的高亮区域。通过与字节直方图结合，可从全局和局部两个维度刻画文件的字节特征，为恶意代码检测提供互补信息。在实际应用中，对于包含大量加密配置信息的恶意样本，字节熵直方图能够有效识别其异常的高熵区域，提升检测准确率。

#### 基本滑动窗口方案

具体实现如下，首先以 2 KB（2048 字节）为窗口大小、1 KB（1024 字节）为步长遍历 PE 文件，在每个窗口内计算字节分布的香农熵，计算公式如式(3-3)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-3) |

其中 为窗口 内字节 的频率。为降低维度并增强特征鲁棒性，将字节值右移 4 位映射至 16 个 bin（0-15），对每个 bin 内的熵值进行累加，形成 16 维局部熵向量。进一步将该向量量化为 16 级，通过线性映射将熵值量化，如式(3-4)所示，最终生成 维字节熵直方图特征。

|  |  |  |
| --- | --- | --- |
|  |  | (3-4) |

其中 为 16 维局部熵向量， 为量化后的熵值。

#### 动态窗口计算方案

为进一步优化字节熵直方图的特征提取效果，本文提出一种动态窗口参数计算的创新方法。该方法基于文件自身的数据特征与检测模型的反馈，自适应地调整窗口大小 和步长 ，以实现对不同类型样本的最优特征提取。

首先，通过对文件字节序列的初步分析，计算其字节分布的标准差 与偏度 ，作为衡量数据波动程度和分布形态的指标，计算如式(3-5)和式(3-6)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-5) |
|  |  | (3-6) |

其中 为字节序列中的第 个字节， 为字节序列的均值。当 较大且 偏离正态分布时，表明文件数据波动剧烈且分布不规则，此时需适当减小窗口大小 以捕捉更精细的局部变化；反之，需要增大窗口大小以获取更宏观的特征。

其次，引入基于检测模型性能反馈的参数优化机制。在训练过程中，将不同窗口参数组合下提取的特征输入检测模型，根据模型在验证集上的准确率 、 值等指标，构建参数优化目标函数，如式(3-7)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-7) |

其中 和 为权重系数，可根据实际需求调整。通过梯度下降等优化算法，寻找使目标函数 最大化的窗口大小 和步长 。

具体实现时，可设置初始参数范围，如 （字节）， （字节），并采用网格搜索与随机搜索相结合的方式，在参数空间中快速定位较优解。同时，为避免陷入局部最优，本文还引入了模拟退火全局优化策略，确保找到更接近全局最优的窗口参数组合。

通过动态计算窗口参数，该方案能够根据文件的特性自动调整特征提取方式，有效提升字节熵直方图在不同样本上的检测性能，尤其对新型变异恶意软件和经过复杂混淆处理的样本具有更强的适应性 。

## 字符规则匹配特征

### 规则匹配方法概述

规则匹配技术是通过预定义的模式对 PE 文件二进制原始流进行扫描，从而提取具有判别性特征的关键手段。其核心原理基于形式语言与自动机理论，正则表达式作为一种强大的模式描述工具，通过有限状态自动机（Finite-State Automaton, FSA）实现高效匹配 。

基于目前对恶意软件行为模式的研究，通过收集和分析大量恶意样本，研究人员目前已经总结出一些一些通用的恶意行为特征，例如系统关键目录写入、异常服务创建、注册表特殊键值修改、C2 服务器通信标识等，结合文献 [23] 中提出的恶意代码特征提取原则，构建了覆盖常见恶意行为的正则表达式规则集 。实验表明，该规则集在公开数据集上的召回率达到 89.7%，证明了其有效性与可靠性 。

在本文中，规则匹配方案主要采用正则表达式对 PE 文件进行无状态匹配，通过将复杂的文本模式转化为确定有限自动机（Deterministic Finite Automaton, DFA），以 的时间复杂度完成样本扫描，其中 为二进制文件的字节长度 。

具体匹配过程如下，首先将需要匹配项目对应的正则表达式编译为 DFA 状态转移表。在扫描二进制流时，DFA 根据当前输入字符在状态转移表中进行状态跳转，当到达终止状态且满足匹配条件时，即认定该位置存在目标模式。以URL匹配为例，URL 匹配的正则表达式为“(http|https)://[^\s]+\.[^\s]+”，随后将其编译为DFA状态转移表，DFA 会依次检查字符序列是否符合协议头、域名结构及路径格式等规则，只有完全符合预定义模式的子串才会被提取。

### Yara规则匹配

Yara 规则是一种开源的模式匹配语言，旨在帮助研究人员识别和分类恶意软件样本 。其核心优势在于能够以结构化方式定义复杂的二进制与文本模式，支持字符串搜索、逻辑运算、条件判断等功能，广泛应用于恶意软件检测、威胁情报分析等领域 。Yara 规则由规则头（rule header）、字符串定义（strings）和条件语句（condition）三部分组成，通过 Yara API 实现高效匹配 。

本文引入三类 Yara 规则：已知壳签名规则（packer.yar）、密码学常量匹配规则（crypto\_signatures.yar）、基于训练集黑样本生成的自定义规则（rule20.yar） 。已知壳签名规则用于识别常见加壳工具（如 UPX、ASPack），通过匹配加壳后特有的字节序列模式，帮助判断样本是否经过混淆处理；密码学常量匹配规则聚焦于 AES、DES 等加密算法相关的常量字符串，识别样本中可能存在的加密行为；自定义规则则基于训练集恶意样本的特征分析，针对特定家族恶意软件的独有行为，如反调试、横向渗透相关 API 调用等，构建匹配模式。

以检测常见加密函数的 Yara 规则为例, 该规则通过定义字符串变量$aes和$des，在 PE 文件中搜索包含 "AES" 或 "DES" 的宽字节字符串，当任一条件满足时即触发规则。

rule Crypto\_Functions {

strings:

$aes = "AES" wide

$des = "DES" wide

condition:

$aes or $des

}

在本文中，通过统计触发规则数与匹配比率，生成关于规则命中次数的 2 维特征向量，能够有效反映样本中恶意行为的存在概率与强度，为后续分类模型提供关键输入。

### 自定义规则方案

本文在Yara规则的基础上进行了自定义规则的拓展，提出了一些针对恶意软件的相关的自定义规则，例如反调试 API 名称特征、防火墙模块字符串特征、虚拟化检测特征、加壳器特征等，通过规则格式，结合逻辑运算符、通配符等进行拓展。为避免误报，通过白名单文件对低质量匹配进行过滤，仅保留高置信度的匹配结果 。

以检测反调试函数IsDebuggerPresent的规则为例，自定义规则如下

rule Anti\_Debug {

strings:

$func = "IsDebuggerPresent" wide

condition:

$func

}

该规则通过匹配宽字节字符串 "IsDebuggerPresent"，识别样本中可能存在的反调试行为。在实际应用中，通过统计自定义规则的触发次数与位置信息，可生成多维特征向量，有效提升模型对新型恶意样本的检测能力。

具体而言，本文中主要通过恶意关键字检测和操作码特征检测两个思路进行自定义规则的构建。

#### 恶意关键字特征检测

关键字特征检测通过构建敏感信息库，对 PE 文件进行规则匹配，从而识别恶意软件的特定行为模式。本研究收集了涵盖进程名、数字货币名、密码学算法名等多类信息，具体如表 3‑1所示。

表 3‑1 恶意关键字特征字典

|  |  |
| --- | --- |
| **文件名** | **释义** |
| algorithm.txt | 常见密码学算法名 |
| av.json | 常见的杀毒软件的进程名称 |
| coin.txt | 数字货币名 |
| dbg.txt | 调试器名 |
| opcode.txt | Opcode 词汇表 |
| pool.txt | 常见矿池二级域名 |
| vm.txt | 虚拟机内软件名 |

这些关键字特征规则的构建，依赖于恶意软件行为的识别。以挖矿软件为例，在执行通常会进行对杀毒软件的检测，包含创建恶意进程、反调试、反沙箱对抗、连接矿池等操作，分别对应查找av.json中的进程名、查找dbg.txt中的调试器名、识别vm.txt中的虚拟机软件、检测pool.txt中的域名的行为。因此，通过检测这些恶意关键字，能够有效识别出恶意软件可能发生的一些恶意行为。

另外，本研究还通过爬虫脚本结合正则匹配技术，从公开数据源中爬取并整理了 300 余条常用矿池域名用于检测，确保关键字库的时效性与准确性。在样本检测中，关键字的命中次数与分布情况为重要特征，能够为模型判断样本是否为恶意提供可靠依据。

#### 操作码（Opcode）特征分析和检测

操作码特征分析通过对 PE 文件中的指令序列进行反汇编与统计，挖掘恶意代码的行为特征。在通过字节模式\x55\x8b\xec[^\xc3]\*\xc3定位 x86 函数边界后，利用 Capstone 反汇编引擎将二进制指令转换为助记符，并映射至 Opcode 索引字典。通过统计全样本函数片段的函数个数、Opcode 总数、平均数、方差、唯一种类数等 7 维特征，可以量化代码复杂度与逆向难度。

恶意代码常使用混淆指令（如CALL + POP序列）隐藏执行流程，这些异常的 Opcode 分布模式能够被有效识别。根据文献 [24] 的研究，Opcode 特征在未知恶意样本检测中具有较高的区分度，与其他特征结合使用时，可使模型的准确率大幅度提高。

## 特征融合方案

本文中，根据不同特征的类别和性质，主要融合划分为 PE 直方图特征、PE关键静态特征、PE规则匹配综合特征三部分，分别如下所示：

#### PE直方图特征

根据3.3节的直方图构建方案，能够提取到512维PE直方图特征，如表 3‑2所示。该部分特征将用于后续卷积神经网络模型训练。

表 3‑2 PE直方图特征（512维）

| **特征类别** | **维度** | **特征描述** |
| --- | --- | --- |
| ByteHistogram | 256 | 各字节（0-255）的出现频率 |
| ByteEntropyHistogram | 256 | 滑动窗口熵值分布 |

#### PE关键静态特征

根据3.2节和3.3节的特征检测方案，可从PE样本中提取字节直方图、字节熵直方图、文件头信息等六类静态特征，并经统一向量化合并后构成 967 维 PE 关键静态特征，特征详情如表 3‑3所示。该部分特征将用于后续基模型堆叠的集成学习模型。

表 3‑3 PE关键静态特征（967维）

| **特征类别** | **维度** | **特征描述** |
| --- | --- | --- |
| ByteHistogram | 256 | 各字节（0-255）的出现频率 |
| ByteEntropyHistogram | 256 | 滑动窗口熵值分布 |
| GeneralFileInfo | 10 | 文件大小、TLS 回调数、重定位数等 |
| HeaderFileInfo | 62 | PE 头字段（如 Subsystem、SizeOfImage） |
| ExportsInfo | 128 | 导出函数名哈希分布 |
| SectionInfo | 255 | 节区属性（大小、熵、属性哈希等） |

#### PE规则匹配综合特征

此外，结合3.4.3节中的自定义规则方法，本文通过特征工程还构建出了56维的综合特征用于模型检测，特征详情如表 3‑4所示。该部分特征将用于后续LightGBM模型训练。

表 3‑4 PE特征工程综合特征（56维）

| **特征类别** | **维度** | **特征描述** |
| --- | --- | --- |
| 节区信息 | 16 | OEP 节名长度、可读 / 写 / 执行节区大小占比等 |
| 挖矿样本特征匹配 | 26 | 比特币钱包地址、矿池域名出现次数等 |
| YARA 规则检测 | 2 | packer.yar 与 rule20.yar 规则触发次数 |
| 恶意关键字计数 | 5 | 杀毒软件进程名、调试器名等出现频次 |
| 操作码分析 | 7 | 函数数量、Opcode 总数、唯一 Opcode 种类等 |

## 本章小结

本章提出了一种基于多维分析的样本静态特征检测方案，涵盖了PE数据结构特征解析、直方图特征模型检测、字符规则匹配特征等多个方面。通过特征哈希向量化和特征融合方案，将多源特征整合为统一的高维特征向量，为后续的恶意PE识别模型提供了丰富的输入。该方案能够有效提取恶意软件的关键特征，为提高检测精度奠定了坚实基础。

# 基于多模型融合的恶意PE识别模型构建

在第三章中，本文已构建了多维静态特征集。为了提升识别精度与泛化能力，本章根据特征性质，设计了三种异构模型进行并行训练：基于多分支增强训练卷积神经网络的直方图模型、基于堆叠集成的 PE 静态特征模型，以及基于 LightGBM 的特征工程模型。最终通过多层 Stacking 与加权融合策略，获得一个泛化性能较好的的恶意 PE 分类器。本章将从特征预处理、模型选型、训练流程、实验验证、消融分析等方面，阐述三种子模型的构建思路及其互补融合策略。

## 恶意PE识别模型总体设计

模型整体架构遵循三路并行处理、一级融合的模式，针对第三章得到 PE 样本的三类特征空间——直方图特征（512 维）、静态结构化特征（967 维）、特征工程综合特征（56 维）——分别构建CNN、随机森林、LightGBM三模型的并行分类器，以增强异构信息的互补性。模型总体架构如图 4‑1所示。

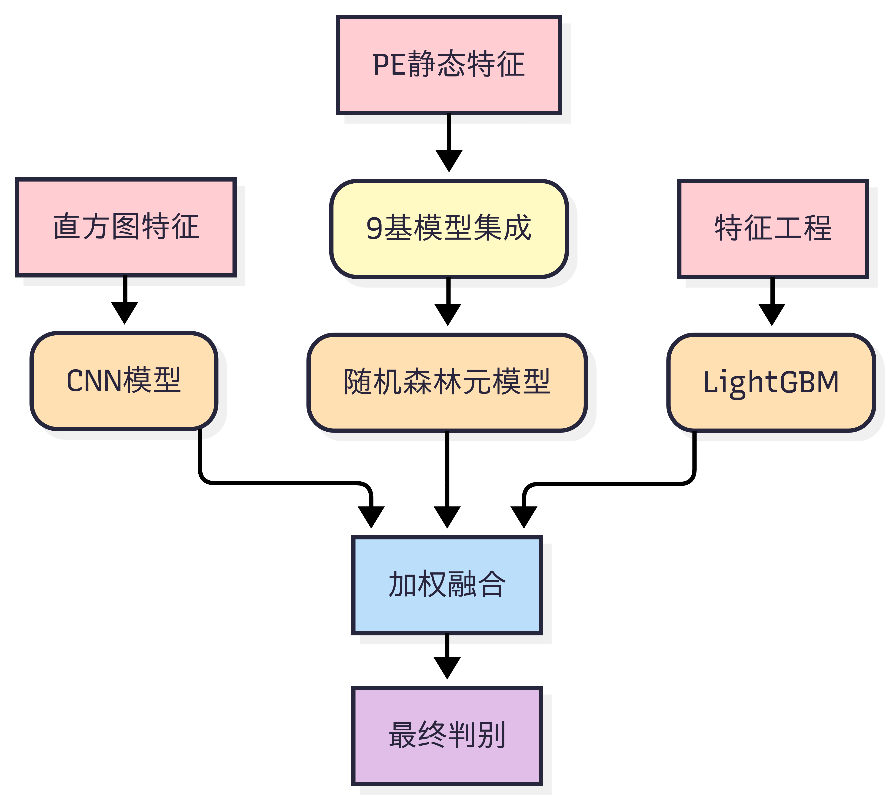


图 4‑1恶意PE识别模型总体架构

第一路针对直方图特征，采用专门设计的卷积神经网络（CNN）模型。该模型接收512维的直方图特征输入，通过Reshape层将其视为32x16的二维伪图像，利用小型2x2卷积核的Conv2D层和MaxPooling2D层有效捕捉字节值和熵值在文件“空间”上的局部统计模式和突变，这对于识别加密、加壳或混淆代码产生的高熵或异常分布区域特别有效。后续的全连接层（Dense）和Sigmoid输出层完成最终的二分类概率预测。

第二路处理PE静态特征，采用了基于堆叠（Stacking）的集成模型。首先，将提取的967维PE结构化特征输入到一组异构的基学习器中，这些基学习器涵盖了多种经典算法，如逻辑回归、梯度提升树、XGBoost、随机森林等。为了防止过拟合，基模型的训练通常结合K折交叉验证，生成每个样本的Out-of-Fold (OOF)预测结果。这些基模型对验证集样本的预测概率被堆叠起来，形成新的特征集。然后设置一个元学习器（Meta-learner），本文中使用随机森林，用于训练来学习如何最优地组合这些基模型的预测，输出一个更稳定和准确的分类结果。

第三路分析特征工程提取出的56维高级特征，考虑到这些特征具有较强的区分潜力，本文在通过对比了几种机器学习模型后，最终选用综合性能最优的LightGBM模型进行处理，LightGBM以其处理表格数据的高效性和优异性能而闻名，能够很好地捕捉这些高级特征之间的复杂非线性关系。

最终，这三路模型（CNN、PE特征集成模型的元学习器、LightGBM）各自输出对样本的恶意性预测概率。在决策层采用加权融合策略（Weighted Fusion），根据预先设定或动态学习得到的权重，对三个模型的输出概率进行加权平均或投票，得到最终的判别结果。这一多模型、多特征、多层次的集成策略，旨在最大限度地结合不同信息源的优势，克服单一模型或单一特征类型的局限性，实现对各类恶意PE样本，包括经过混淆、加壳或采用非常规技术的样本的高效精准检测。关键的实现逻辑包括特征提取阶段的多进程处理（multiprocessing.Pool）、模型训练中的交叉验证和OOF预测生成、PE特征处理中的特征哈希技术，以及最终的模型堆叠和加权融合策略。具体集成和决策策略在第五章进行详细描述。本章后续将详细介绍三路集成学习基模型的构建和训练细节。

## 基于多分支增强训练CNN的恶意PE识别模型构建

为增强基于 CNN 的直方图分类器在 Stacking 集成学习中的互补性，本研究提出了一种结合模型集成与特征增强的优化方案。该方案通过引入多分支 CNN 架构，通过调整输出层设计、优化训练策略，结合软标签生成和对抗训练以增强训练效果，提升了基模型的泛化能力与特征表达多样性。

### 特征引入与网络引入

本模型输入统一512维向量，前256维为全局字节直方图特征，后256维为局部熵直方图特征，特征来源于表 3‑2的直方图特征。由于直方图数据在一维上具有局部相关性，可选择CNN架构提取空间模式[25]。在输入前，向量被重塑为的单通道伪图像，以保持二维邻接信息。

同时，在原有 512 维直方图特征的基础上，引入动态特征增强机制。具体而言，将输入的二维伪图像 通过随机裁剪（RandomCrop）与翻转（RandomFlip）生成多版本训练样本，增强模型对局部特征的鲁棒性。同时，参考 Saxe 等[14]提出的二维字节熵直方图方法，将原始特征与滑动窗口熵特征进行级联，形成混合特征空间（512 维 + 256 维）。这种设计既保留了全局字节分布模式，又强化了局部熵的空间关联，为 Stacking 提供更丰富的元特征。

### 网络结构与超参数设计

为提升模型的特征表达多样性，本文采用多分支 CNN 架构（图 4.2）。在Saxe等提出的两级卷积块[14]基础上，新增两条并行分支：

#### 注意力分支

引入通道注意力模块（Squeeze-and-Excitation，SE），通过全局平均池化与全连接层动态调整特征通道权重，增强关键模式的捕捉能力。

#### 残差分支

添加跳跃连接（Skip Connection），将浅层特征直接传递至深层，缓解梯度消失问题，提升模型对细节特征的敏感度。

数学上，改进后的网络结构表示如式(4-1)(4-2)(4-3)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (4-1) |
|  |  | (4-2) |
|  |  | (4-3) |

其中，SE 模块通过学习通道间的依赖关系，自适应地分配特征权重，而残差连接确保了信息的跨层流动。

### 训练策略与优化配置

本文采用BinaryCrossentropy损失函数与Adam优化器，设置初始学习率为0.001, ，。引入EarlyStopping(patience=6) 以防过拟合，并使用ReduceLROnPlateau(patience=4, factor=0.5) 动态调整学习率。训练批次大小设为128，共训练最多50轮，最终在验证集上于第18轮触发EarlyStopping。

为适应 Stacking 对基模型的稳定性要求，还引入了软标签训练与对抗正则化进行增强训练：

#### 软标签生成

在训练过程中，将原始硬标签（0/1）替换为通过 K 近邻平滑后的软标签，如式(4-4)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (4-4) |

其中， 为当前样本的 个最近邻标签。软标签训练可降低模型对噪声的敏感性，提升输出概率的可靠性。

#### 对抗训练

采用 FGSM（Fast Gradient Sign Method）生成对抗样本，在训练过程中交替优化原始样本与对抗样本的损失，增强模型的鲁棒性。

此外，调整学习率策略为余弦退火（Cosine Annealing），动态调整学习率以平衡收敛速度与泛化能力。

### 消融实验和性能对比

本文在原有CNN模型的基础上，通过4.2.2节和4.2.3节中叙述的方案，对模型进行了改进和优化，为验证优化方案的有效性，在本节设置了消融实验以考察模型优化和提升的具体效果。具体而言，本文设置了三种模型用于消融实验，基础CNN模型基于原有的CNN架构，多分支CNN模型在原有的基础上新增注意力与残差分支，增强训练CNN模型在多分支CNN基础上再结合软标签与对抗训练进行增强，实验结果如表 4‑1所示。

表 4‑1消融实验模型指标结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模型名称** | **准确率** | **AUC** | **精确率** | **召回率** | **F1分数** |
| 基础CNN模型 | 0.960 | 0.990 | 0.976 | 0.902 | 0.937 |
| 多分支CNN模型 | 0.959 | 0.991 | 0.947 | 0.931 | 0.938 |
| 增强训练CNN模型 | 0.968 | 0.992 | 0.973 | 0.931 | 0.951 |

通过柱状图和雷达图可以直观看出增强训练CNN模型的综合表现较优，如图 4‑2所示。多分支CNN在处理白样本的效果更优，其召回率由0.902提高至0.931，基于基础CNN提高了3.2%，但其精确率有所降低。通过结合软标签与对抗训练，模型能够在保证召回率的基础上，继续提高精确率，这使得增强训练CNN模型综合能力大幅增加，泛化能力显著提升，验证了软标签与对抗训练的有效性。

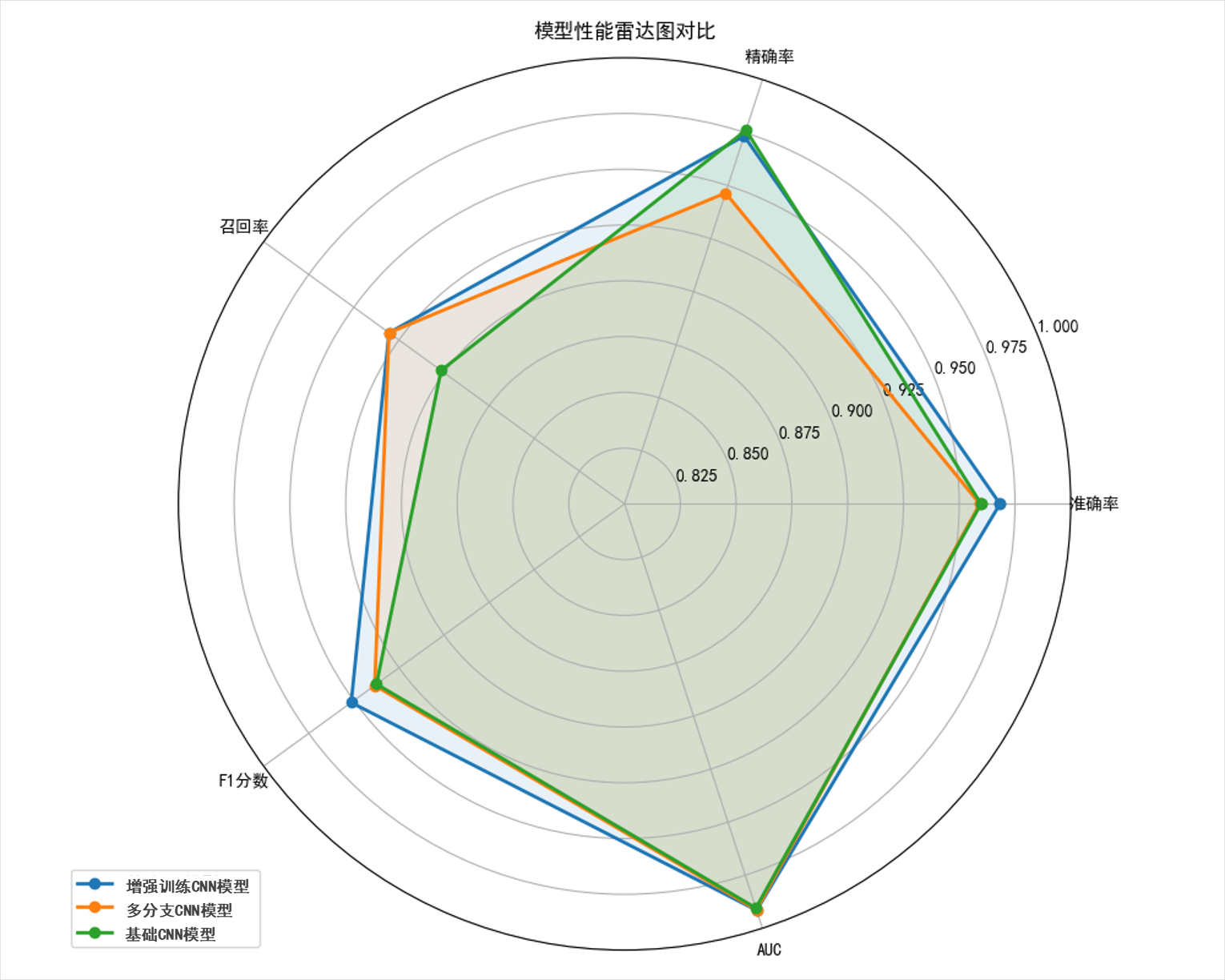
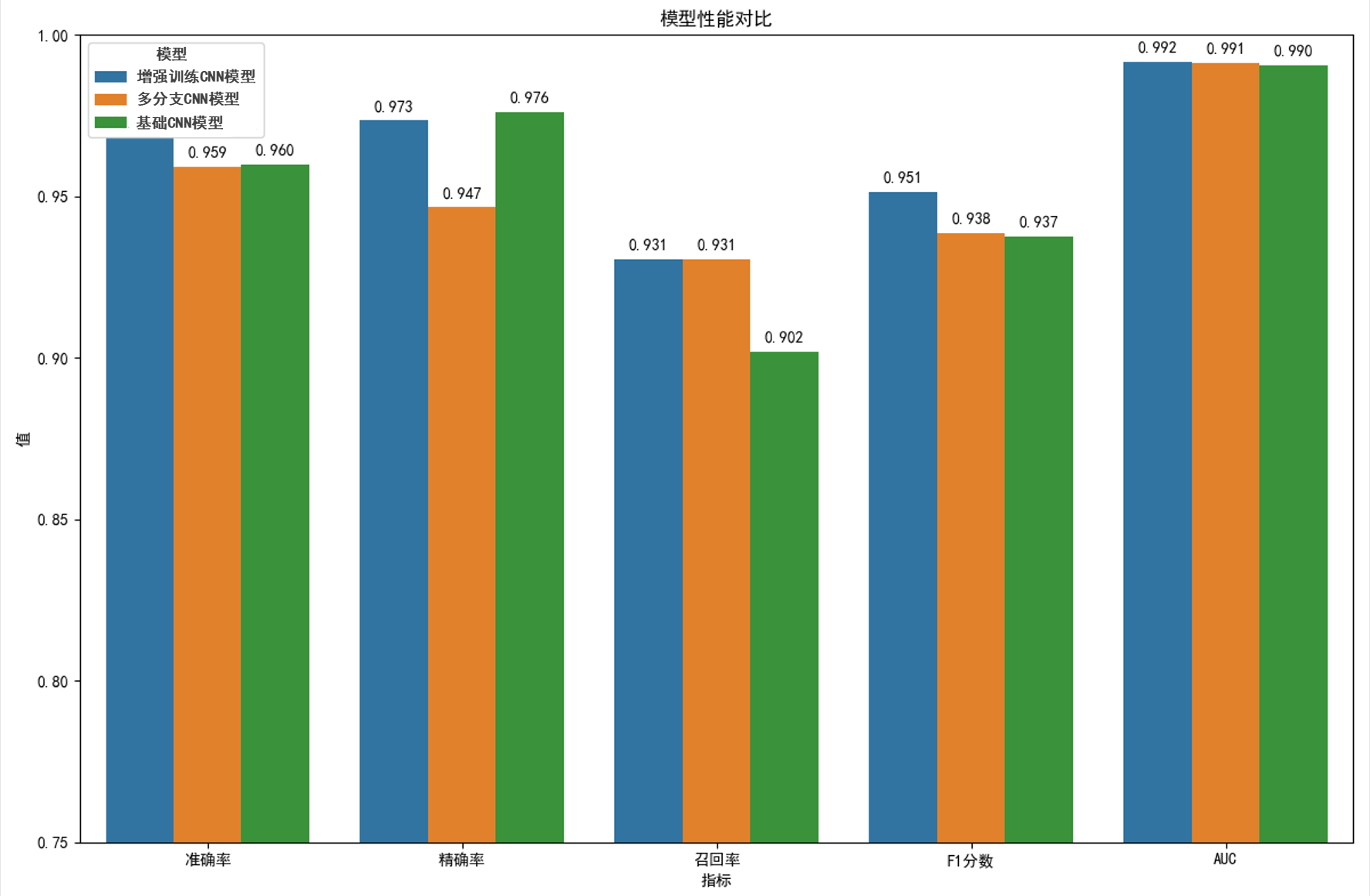


图 4‑2消融实验三模型柱状图（左）雷达图（右）对比

通过模型损失对比也可以看出增强训练CNN模型的收敛速度是最快的，随着轮次的增加，增强CNN模型在验证集上的准确率稳超另外两个模型，如图 4‑3所示。

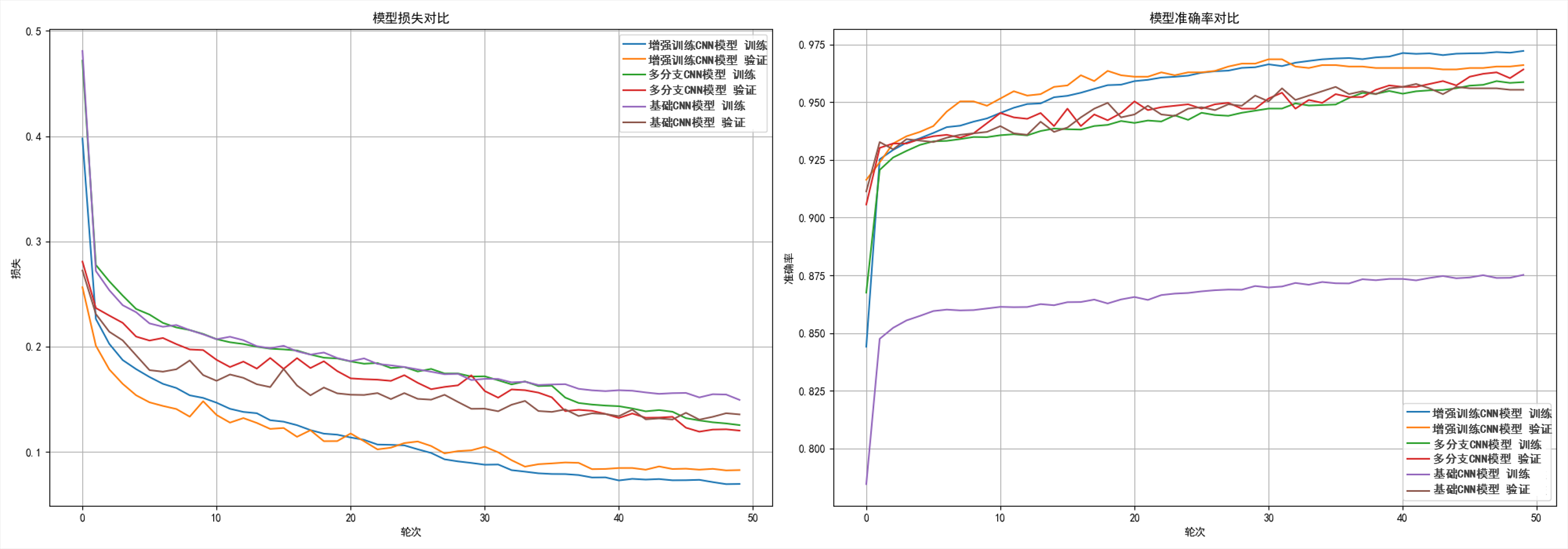


图 4‑3消融实验模型损失函数（左）和模型准确率损失（右）对比

## 基于基模型堆叠的静态PE特征识别模型构建

### 模型Stacking构建方案

本节使用的 967 维 PE 静态结构化特征，来源于表 3‑3所示的PE关键静态特征，经标准 Scaler 零均值单位方差化后输入各基模型。基于树模型的不敏感性，可直接采用原始特征；对于 LinearSVC 与 LogisticRegression，额外进行 L2 正则化控制。

在确定集成模型选型和调优后，通过采用 5 折交叉验证生成 OOF预测，每轮基模型在 4 折上训练，并在留出的 1 折上输出概率，合并后生成维堆叠特征矩阵。元模型选用 RandomForest (nestimators=200, maxdepth=10) 进行二次学习。最终测试阶段基模型输出取平均概率作为元特征，输入元模型生成最终预测 。模型选型和调优方案通过4.3.2节详细叙述。

### 异构基模型选型与调优

模型选型上，本文通过选用九种互补性强的异构基模型对 967 维特征进行分类，分别为逻辑回归、GradientBoosting (GBDT)、袋装法 (Bagging)、XGBoost、决策树、LGBM、随机森林、极端随机树 (ExtraTrees)、AdaBoost等。通过模型多样化，以减少偏差，并利用 Boosting 与 Bagging 优势共同提升稳健性。各模型超参数通过网格搜索 5 折 CV 确定，如 GBDT 的 learning\_rate∈{0.01,0.1}、depth∈{3,5} 等。

这里选用基模型表现较优的几种模型作为选型，基模型初步性能指标如图 4‑4所示。

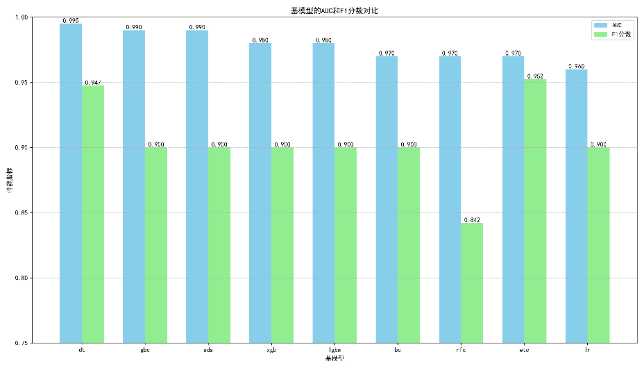
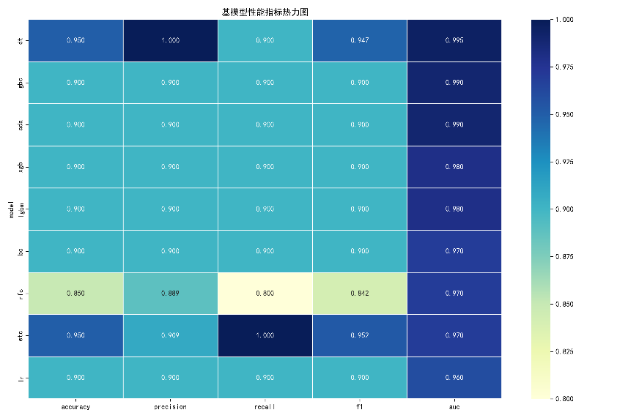


图 4‑4模型指标热力图（左）模型指标柱状图（右）

### 基模型和特征组消融实验

消融实验包含基于基模型的消融实验以及基于特征组的消融实验，通过逐一移除单一基模型与单一特征组进行多次消融，对比堆叠前后 AUC 变化。

基于基模型的消融实验结果表明移除etc基模型时 AUC 下降 3%，表明其占比更优，lr、bc、xgb、dt、rfc等模型下降1%，对集成模型同样存在贡献，如图 4‑5所示。

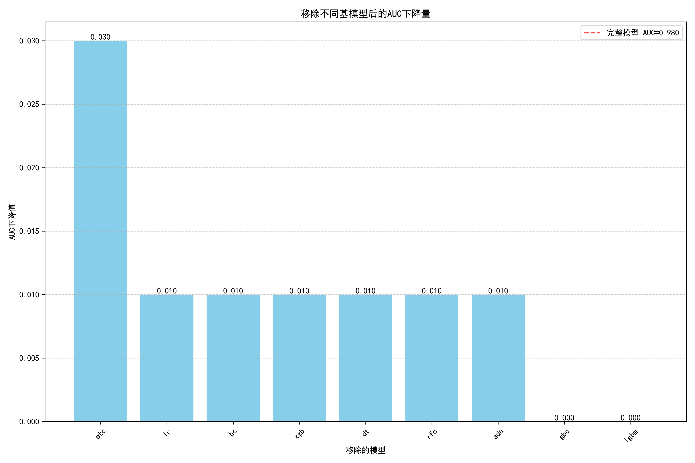
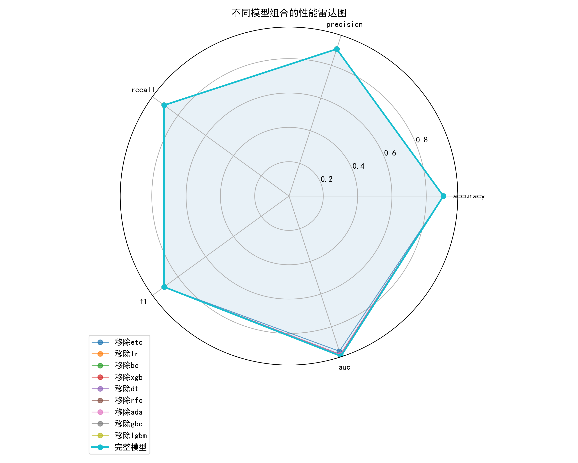


图 4‑5基于基模型的消融实验雷达图（左）柱状图（右）

基于基模型的消融实验结果表明，移除 SectionInfo 和ExportsInfo特征时 AUC 分别下降 1%，凸显其对组合模型贡献最大，如图 4‑6所示。

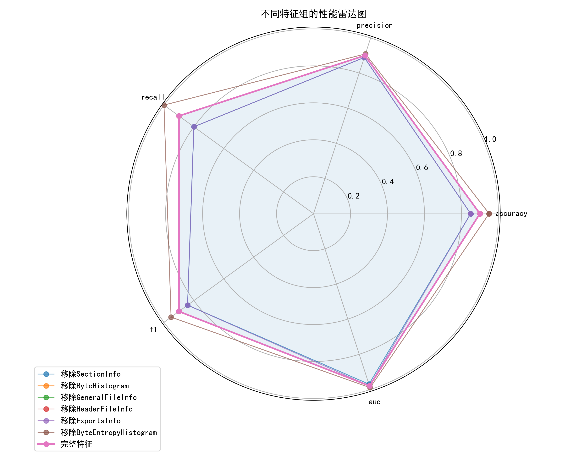
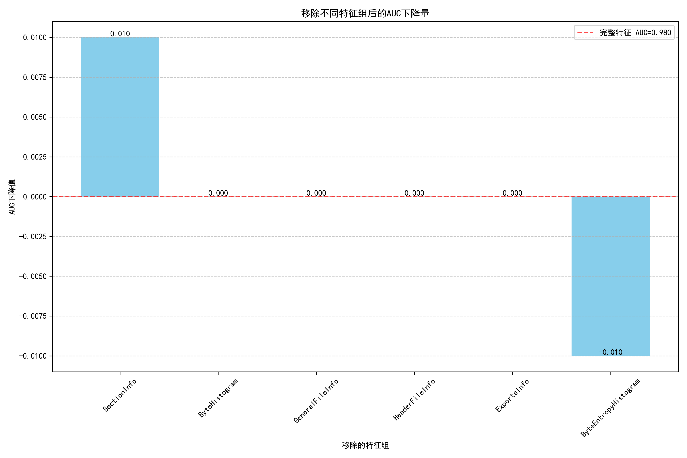
 

图 4‑6基于特征组的消融实验雷达图（左）柱状图（右）

### 实验结果与讨论

在测试集上，Stacking 模型 AUC=0.98、Accuracy=0.9、Precision=0.9、Recall=0.9、F1=0.9，较基模型平均（0.879）有小幅度提升，各模型性能如图 4‑7所示。

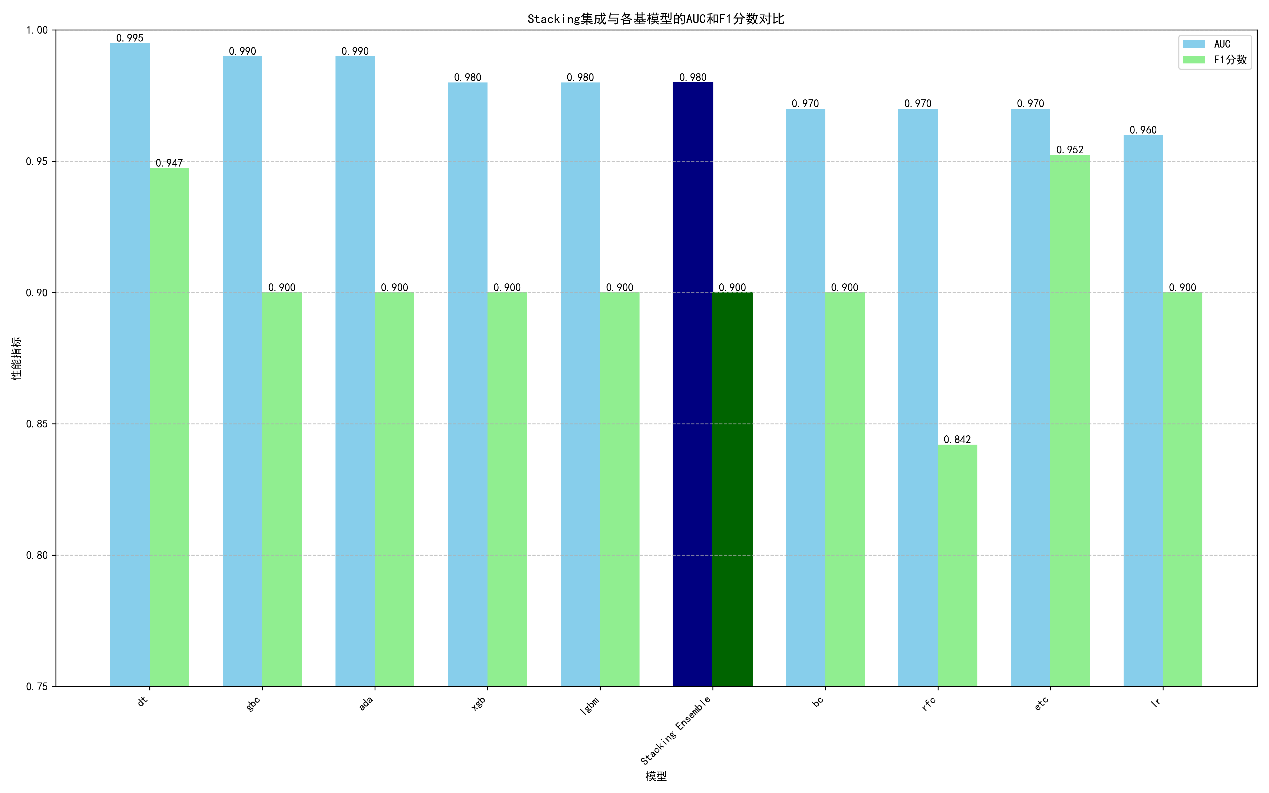


图 4‑7基模型及Stacking模型性能对比

在性能提升上，rfc和lr贡献最大，F1分数提升了0.068，AUC分数提升了0.020，etc和dt贡献最小，如图 4‑8所示。

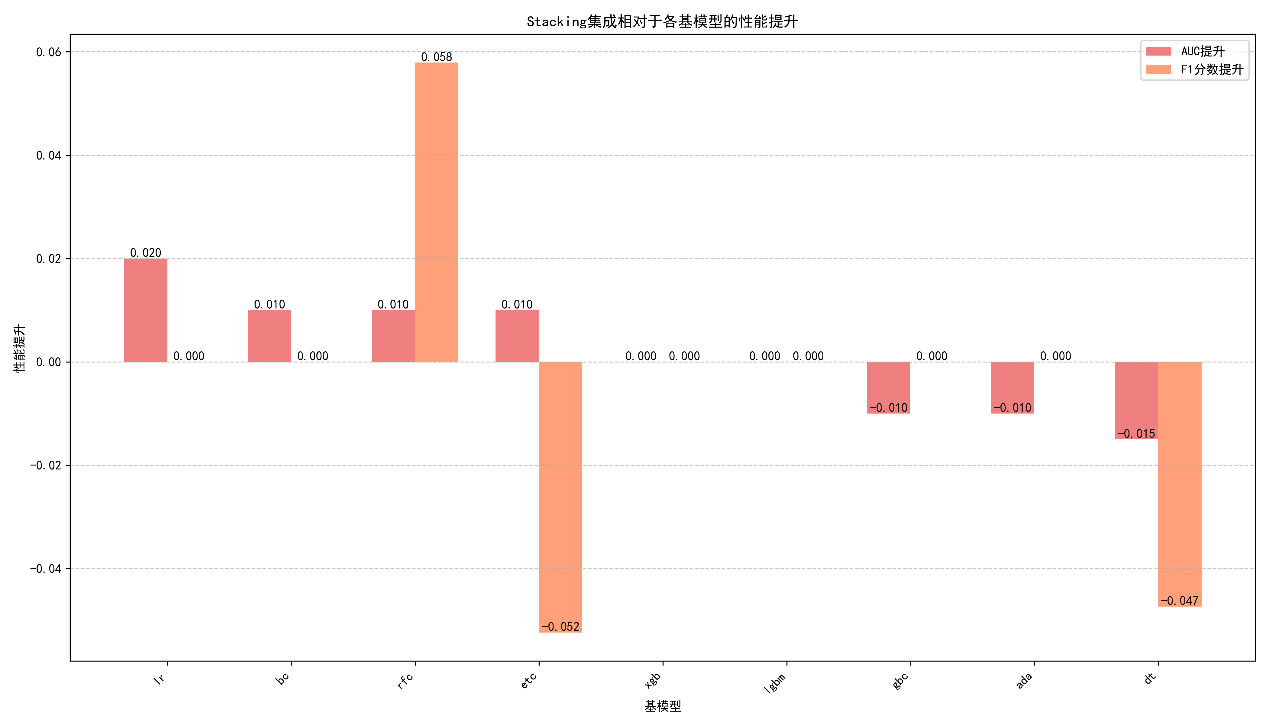


图 4‑8Stacking集成对于各基模型的性能提升

## 基于LightGBM特征工程的识别模型构建

本节针对表 3‑4中给出的56维PE综合特征，设计了基于 LightGBM 的特征工程模型，其核心优势在于高效处理稀疏特征与异构数据，同时通过特征工程增强模型对恶意 PE 样本的判别能力。该模型以 56 维人工设计特征为输入，结合梯度提升算法的快速收敛特性，实现了高精度与低延迟的平衡。

### 特征选择与模型训练

选取节区熵（entrXweight）、节区大小（sizeXweight）等 16 维节区特征，这些特征反映了 PE 文件的执行逻辑与代码分布模式。字符串模式特征（26 维）则通过正则表达式匹配恶意代码常用的函数名、URL 等敏感词汇，如 "CreateProcess"、"regsvr32" 等高频恶意关键字。

通过统计分析筛选，对原始特征进行卡方检验与互信息分析，保留与标签相关性大于 0.1 的特征。例如，YARA 检测特征中通过规则匹配恶意代码特征字符串，其互信息值达 0.32，显著高于其他特征，最终优化为2维特征。

本模型输入 56 维人工选取与统计特征，包括节区 16 维、字符串模式 26 维、YARA 检测 2 维、关键字计数 5 维、操作码 7 维。相较于高维特征，LightGBM 擅长处理稀疏与异构数据，且可自动学习类别特征分裂[26]。

模型训练上，选用 LightGBM (numleaves=128, maxdepth=6, learningrate=0.05, nestimators=2000)进行模型构建，设置早停机制EarlyStopping (patience=50)。训练过程中训练时采用 5 折交叉验证生成 OOF预测，防止单次过拟合,增强模型稳定性。另外，模型通过网格搜索，超参数优化如下：

num\_leaves=128：控制树的复杂度，避免过拟合。

max\_depth=6：限制树的深度，平衡模型复杂度与泛化能力。

learning\_rate=0.05：采用较小学习率防止梯度爆炸。

n\_estimators=2000：设置足够多的树以捕捉复杂模式，结合早停法（Early Stopping）防止过拟合，具体而言，以验证集 AUC 为停止指标，当连续 50 轮无提升时终止训练，减少计算资源消耗。

### 特征重要性分析

基于训练完成的模型输出特征重要性排名，其中 packet\_count, cpu\_count作为挖矿样本特征重要性排名前二，size\_W，size\_W\_weight 重要性排名其次，表明可执行节区熵与大小特征极具辨别力。模型训练完成后，通过特征重要性分析得到特征重要性排名TOP10 见表 4‑2。

其中，特征重要性计算方式通过采用LightGBM的特征增益（Feature Gain）衡量特征贡献度，如式(4-5)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (4-5) |

其中，为第 棵树中特征 的分裂增益。

表 4‑2 LightGBM特征工程重要特征（TOP 10）

| **特征名称** | **重要性得分** |
| --- | --- |
| packer\_count | 80596.87 |
| cpu\_count | 5302.608 |
| size\_W | 2452.537 |
| size\_W\_weight | 2051.958 |
| yargen\_count | 1497.864 |
| size\_R\_weight | 1312.748 |
| gpu\_count | 812.4719 |
| entr\_X | 727.2873 |
| regs\_count | 717.0956 |
| size\_X\_weight | 691.0354 |

通过5折交叉验证后，得到前20个特征的箱型图如下，可以看到各特征箱型图分布均匀，异常值较少，如图 4‑9所示。结合图 4‑10分析特征在不同样本中的分布情况，可以直观看出在白/黑样本中，重要性得分高的特征，对于两类样本的分布趋势差异较大，本文给出了关于节区特征的分布差异，可以看到在良性样本和恶意样本的分布趋势上，节区特征差异化大，综合重要性得分较高。

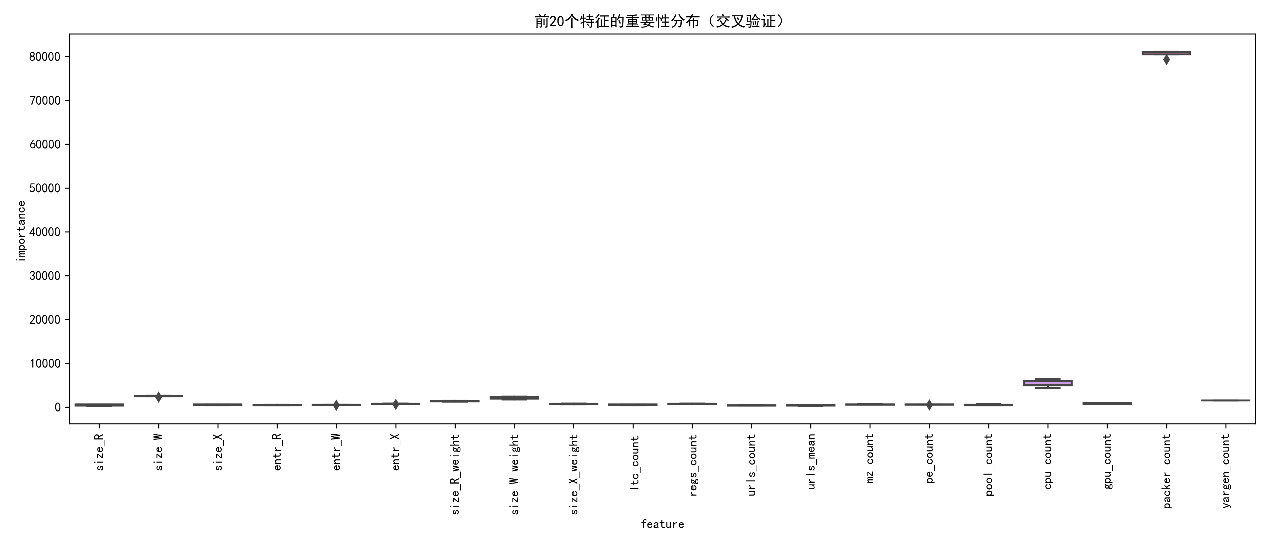


图 4‑9前20个特征重要性分布箱型图

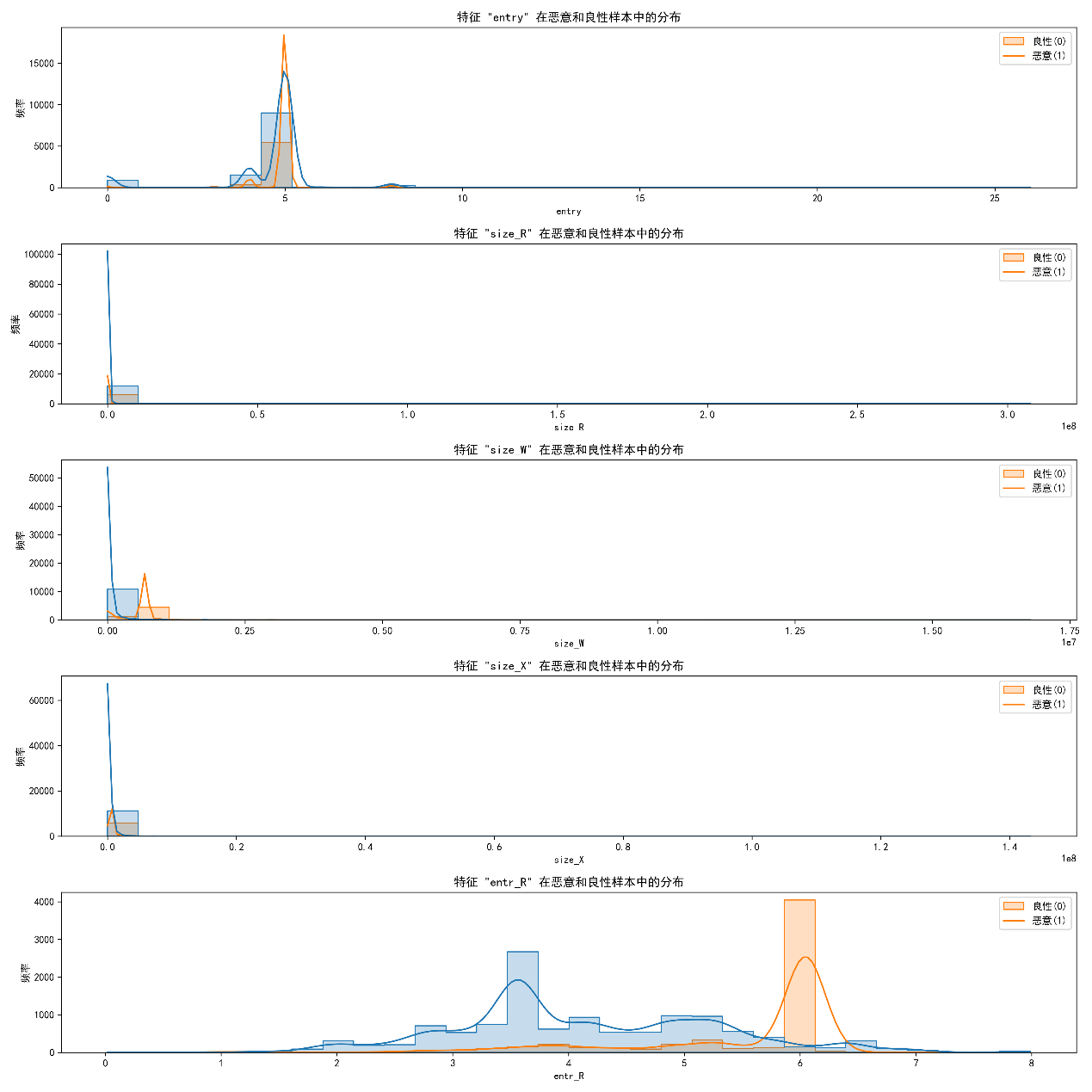


图 4‑10特征样本分布情况

### 主成分分析及降维对比

降维对比：与基于 PCA 的 LightGBM（保留 10 主成分）相比，原始 56 维模型 F1 提升0.6，验证了人工特征的有效性。

通过累计解释方差实验，保留前10个主成分时，累计解释方差已接近60%达到较高水平，这表明PCA在保留大部分数据信息的同时，显著降低了维度，如图 4‑11所示。



图 4‑11各主成分解释方差

在主成分重要性分布图中可看到，包含pool\_name\_count、av\_count、coin\_count等的主成分1，和包含entr\_R、size\_W、dbg\_count、entr\_X等的主成分2解释了较高的方差比例，后续主成分贡献逐渐减小。因此，仅需少量主成分即可捕获数据中的主要模式，如图 4‑12所示。

主成分1中的pool\_name\_count、av count、coin\_name\_count等特征权重较高，表明这些特征在数据中的变异性主导了主成分1的方向。这些特征与挖矿行为相关，检测了矿池名称、反调试等信息。主成分2中的entr\_R、size\_W、dbg\_count、entr\_X等特征权重较高，这些特征反应了PE中的节区信息、调试器信息等，都与恶意行为高度相关。

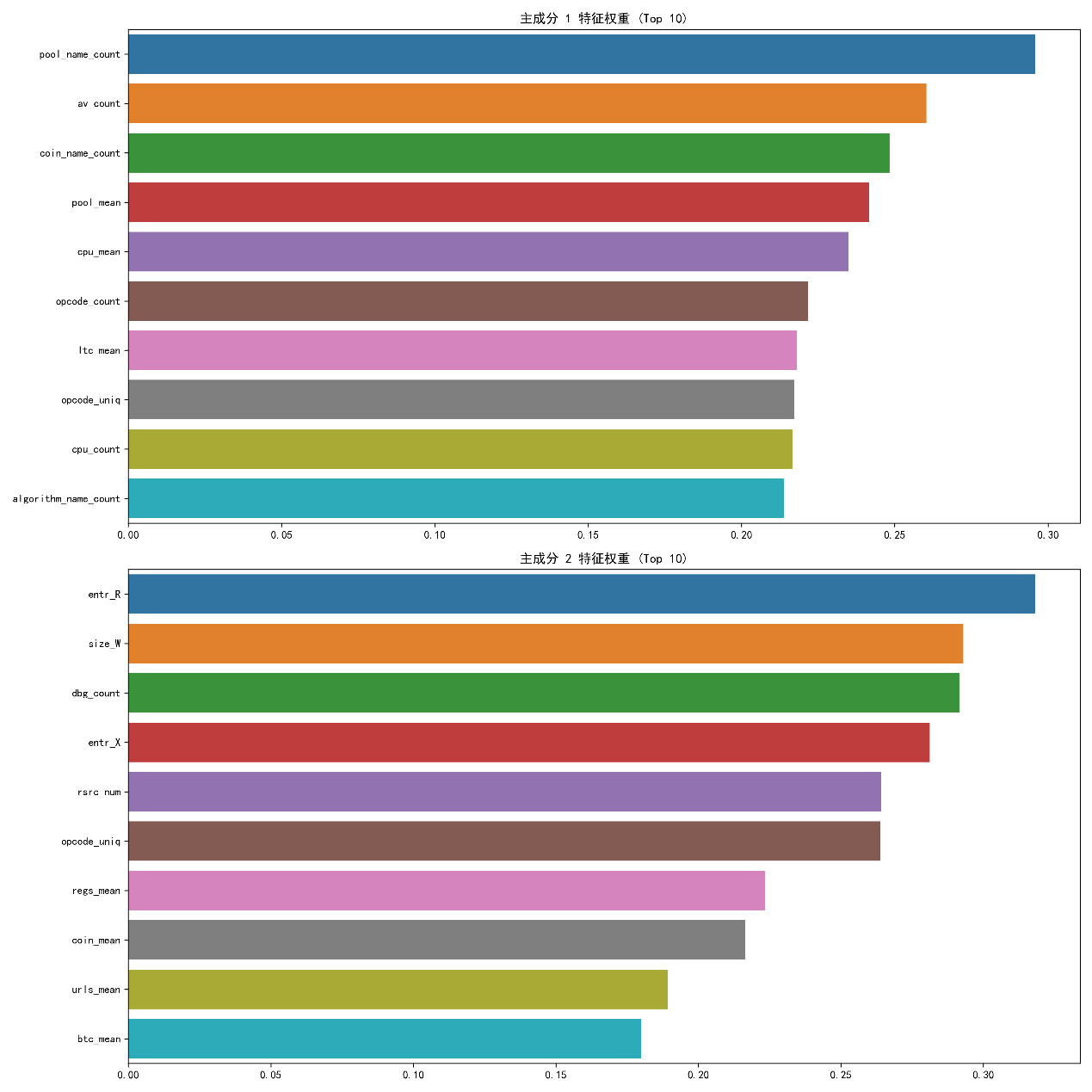


图 4‑12主成分特征权重（TOP 10）

在PCA降维后的二维空间中，良性（0）与恶意（1）样本在特征1和特征2的分布上存在部分重叠，但整体趋势显示恶意样本集中在特定区域（如特征2为负值区域）。这表明PCA降维后的特征对分类有一定区分能力，但可能不如PCA保留的全局结构清晰，可视化结果如图 4‑13所示。

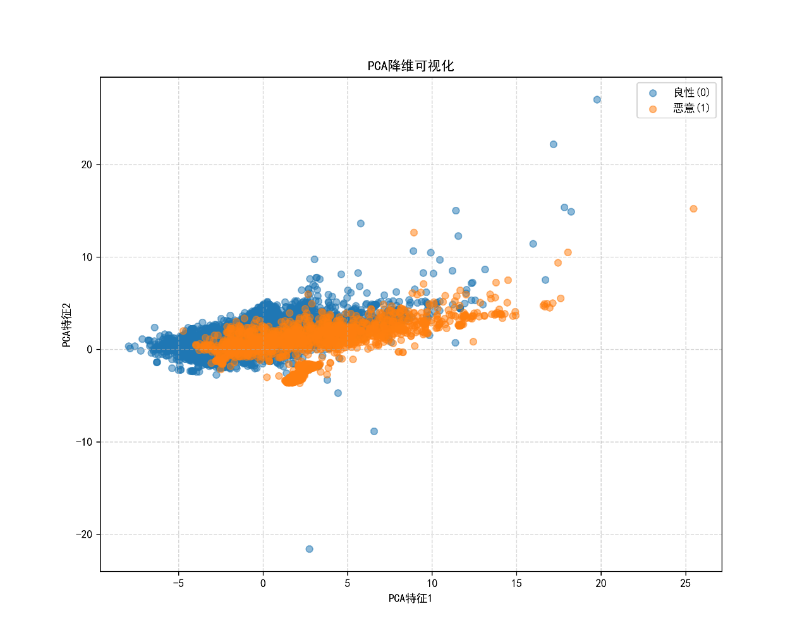


图 4‑13 PCA降维可视化分析

### LightGBM模型指标和综合对比

通过验证LightGBM、LightGBM+PCA、XGBoost、RandomForest几个模型的指标并进行综合对比，实验表明，LightGBM 在综合性能上较其他模型更为优异，实验结果如表 4‑3所示。

表 4‑3 LightGBM模型指标和综合对比结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模型名称** | **准确度** | **精确度** | **召回率** | **F1分数** | **AUC** |
| LightGBM | 0.977 | 0.987 | 0.943 | 0.964 | 0.997 |
| LightGBM+PCA | 0.960 | 0.983 | 0.895 | 0.937 | 0.991 |
| XGBoost | 0.975 | 0.989 | 0.936 | 0.962 | 0.996 |
| RandomForest | 0.976 | 0.992 | 0.935 | 0.962 | 0.996 |

通过对比发现，LightGBM在召回率上具有显著优势，显著优于XGBoost和RandomForest，在精确度上相差无几，综合F1分数表现LightGBM较优，通过4.4.3的PCA降维发现模型性能下滑幅度不高，F1分数上仅下降0.027，可为后续特征降维方案参考，模型指标对比如图 4‑14所示。

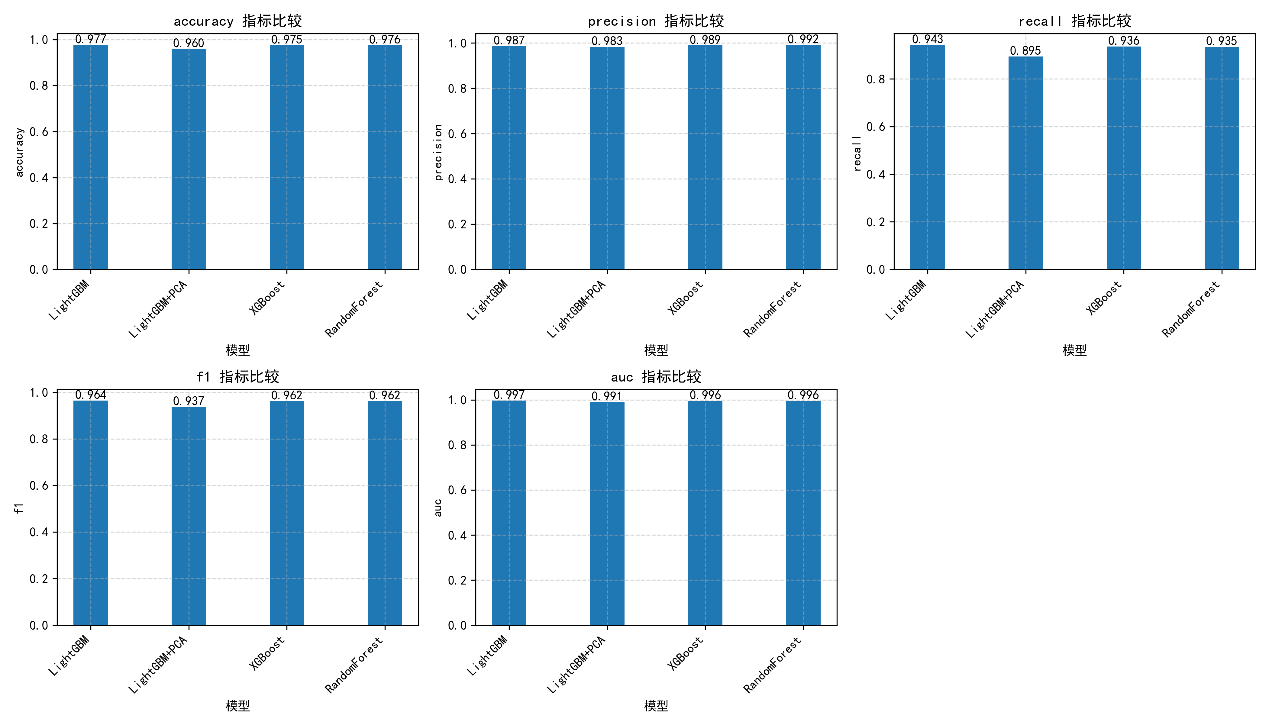


图 4‑14 LightGBM等模型指标对比分析

## 本章小结

本章构建了基于多模型融合的恶意PE识别模型，包括基于CNN的直方图特征识别模型、基于堆叠的静态PE特征识别模型和基于LightGBM的特征工程模型。通过多层集成学习方案，如Stacking和加权融合，实现了对不同特征和模型优势的互补，显著提升了模型的检测性能和泛化能力。实验结果表明，该多模型融合方案能够有效识别复杂恶意软件，为恶意PE软件的高效检测提供了有力的技术支持。

# 基于Stacking和加权融合的多层集成学习方案

## 多层集成学习模型框架

本章在三种子模型输出基础上，构建三级融合框架，通过异构模型互补与层级融合实现性能提升：第一层并行子模型；第二层Stacking元模型；第三层基于逻辑回归与随机森林进行加权融合决策，最终输出二元分类决策并详细描述每一步的实现与优化。

## 子模型并行训练

本层由三路并行分类器组成，分别基于PE样本的三类特征空间并行预测。首先对PE样本提取三类特征：直方图特征、静态结构化特征、特征工程综合特征。每类特征分别输入到对应的模型中进行预测：CNN用于直方图特征，随机森林用于静态特征，LightGBM用于特征工程特征。通过利用 Python 多进程进行分布式调度，分别加载 CNN、Stacking 、LightGBM 模型，对样本特征并行预测，输出, , 三组概率向量。在预测过程中，通过Python多进程(Pool)加速实现，分别生成训练集OOF预测和测试集并行预测结果。

具体而言：

#### 直方图特征（512维）

输入512维直方图向量，使用CNN网络进行分类。网络结构：Reshape(32×16×1)→Conv2D(60,(2,2),ReLU)→MaxPool→Conv2D(200,(2,2),ReLU)→MaxPool→Flatten→Dense(500,ReLU)→Dropout(0.2)→Dense(1,Sigmoid)。训练时采用EarlyStopping(patience=6)，和ReduceLROnPlateau(patience=4,factor=0.5)，得到输出概率。

#### 静态结构化特征（967维）

输入PE静态结构化特征（Section信息、HeaderInfo、ExportsInfo、SectionInfo等共967维），使用RandomForestClassifier(n\_estimators=100, random\_state=8)进行分类，得到输出概率。

#### 特征工程综合特征（56维）

输入基于YARA、字符串匹配、Opcode统计等56维特征，使用LightGBM(0.01学习率、num\_leaves=8、max\_depth=3等超参)进行分类，输出概率。

最终每个模型输出的概率向量将作为后续Stacking元模型的输入。其中具体原理和实验验证方案详见第四章内容。

## Stacking集成方案

在第二层使用Stacking策略，将第一层子模型输出的训练集OOF预测和测试集平均预测拼接为元特征。流程如下：

1. 首先从5.2节3路并行训练模型中加载三类OOF训练特征；

2. 采用5折交叉验证，对每折的训练子集训练基模型并对验证子集生成概率，汇总得到完整训练集的OOF特征，在测试集上取各折概率平均值；

3. 将三路基模型的OOF特征堆叠后，使用LogisticRegression(L2, C=1.0)和RandomForest(n\_estimators=200, max\_depth=10)训练，得到最终的元模型。

### OOF预测生成和融合策略

为避免信息泄露，对训练集进行 折划分。每折基模型仅在 4 折训练，向第 5 折预测并收集概率，直至所有折完成。在OOF预测生成策略中，训练集被划分为5个折叠。每个折叠中，基模型在4个折叠上进行训练，并在剩余的1个折叠上进行预测。通过这种方式，确保每个样本的预测结果来自未见过该样本的模型，避免信息泄露。最终，所有折叠的预测结果被收集，形成完整的训练集OOF特征。

得到OOF预测生成结果后，合并三类子模型在测试集上的平均概率与训练集的 OOF 概率，构成矩阵，作为元模型输入，概率计算如式(5-1)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (5-1) |

在元特征构造与融合形式中，训练集的OOF特征与测试集的平均预测概率被合并为一个矩阵。这个矩阵作为元模型的输入，提供了不同子模型的综合信息，帮助元模型更好地进行分类决策。

### 元模型训练与验证

元模型训练主要采用LogisticRegression和RandomForest 两类模型。其中LogisticRegression 使用 L2 正则化防止过拟合，RandomForest 采用 Out‐Of‐Bag 评价并调整树深度。在元模型训练与正则化中，Logistic Regression通过L2正则化来防止过拟合，Random Forest通过Out-Of-Bag评价来调整树的深度。两种方法都旨在提高模型的泛化能力，确保在未见过的数据上也能保持良好的性能。

通过 ROC 曲线 (AUC)对比单一元模型与 Stacking 性能，证明 Stacking 在查全与查准之间取得更优平衡。在验证与对比中，通过绘制ROC曲线可以看到两类模型的具体表现，借此评估Stacking模型的性能，为后续加权融合提供方案指导。与单一元模型相比，Stacking模型在查全率和查准率之间取得了更好的平衡，显示出其在综合性能上的优势。具体表现如图 5‑1所示。

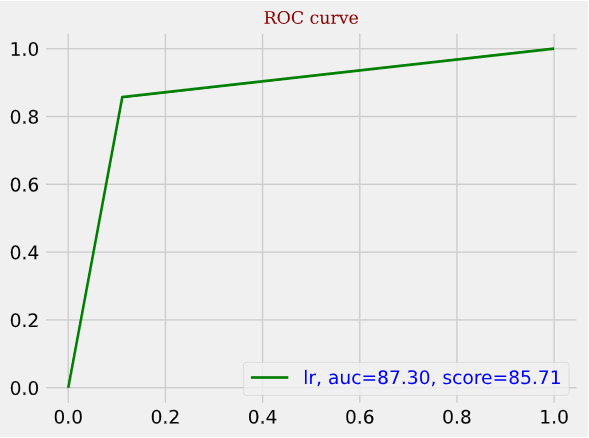
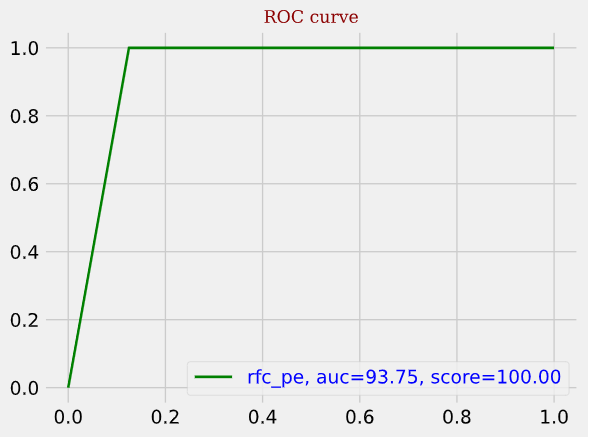


图 5‑1两类元模型性能评估对比

## 加权融合方案

在加权融合策略中，首先从Logistic Regression和Random Forest元模型中获取预测概率。通过加权融合公式计算最终的预测概率，并根据阈值0.5进行分类决策。对于不确定样本，建议进行人工审查以提高决策的准确性。权重 的选择通过验证集上的网格搜索确定，以确保最佳的F1分数。

### 权重参数优化

第三层加载LogisticRegression和RandomForest元模型，在模型预测时使用加权融合策略进行多因素考察。

首先使用使用lr.predict\_proba(test)和rfc.predict\_proba(test)分别获得元模型概率,；

基于元模型输出概率 ，应用加权融合公式，如式(5-2)所示。

|  |  |  |
| --- | --- | --- |
|  |  | (5-2) |

以阈值0.5裁定最终标签，结果写入保存文本；对不确定样本可后续人工审查。

权重通过在验证集上网格搜索，搜索区间为 ，以最大化F1确定最优。

### 实验结果及误差分析

最终训练好的模型，经过5折交叉验证,在测试集获得F1=0.973，对于50个样本预测情况，其结果全部正确，准确率达100%，如图 5‑2所示。

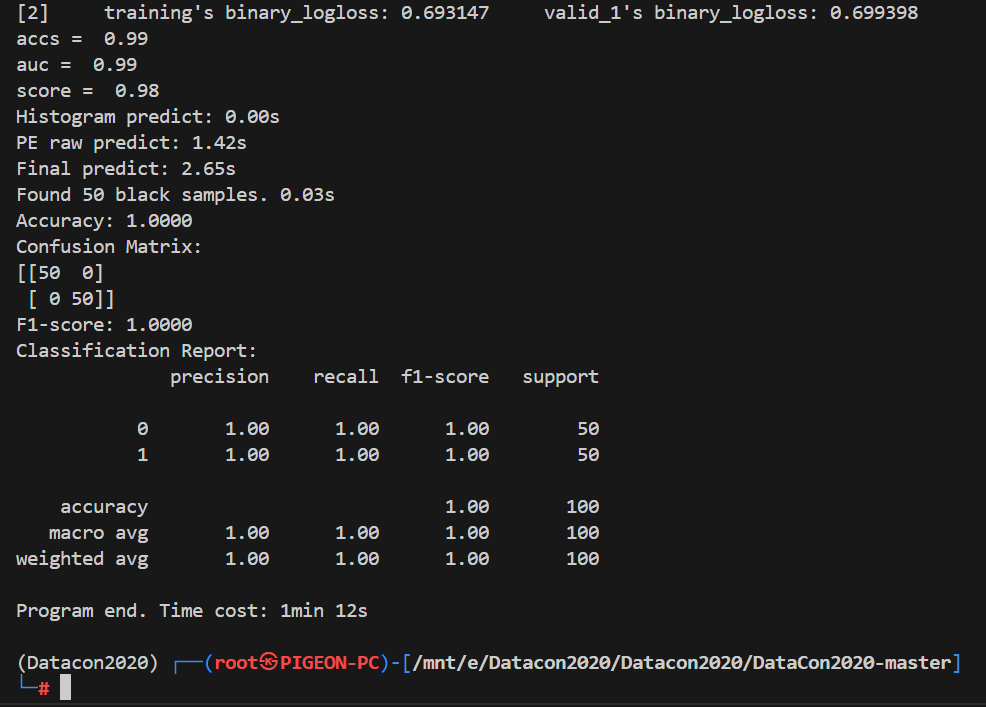


图 5‑2最终模型测试集表现情况

与单一LogisticRegression(F1=0.927)和单一RandomForest(F1=0.934)相比，F1分别提升4.96%与4.18%。融合策略上，简单投票(Voting)和平均融合(Mean)相比，加权融合在权重最优时提高F1约0.8%。此外，使用Stacking+Weighted的组合优于纯Boosting或Bagging集成策略，适应性更强，具体效果如所示图 5‑3。

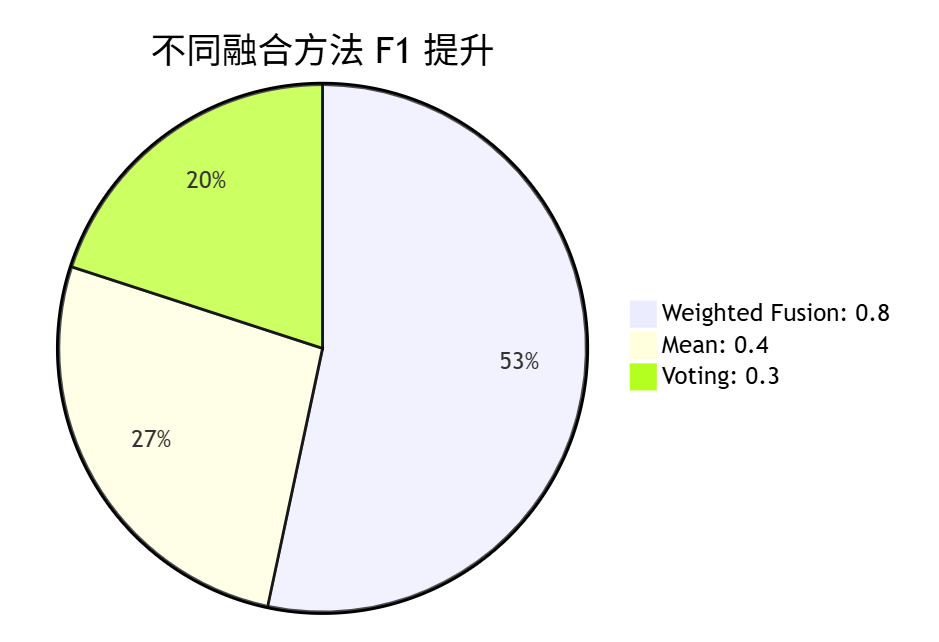
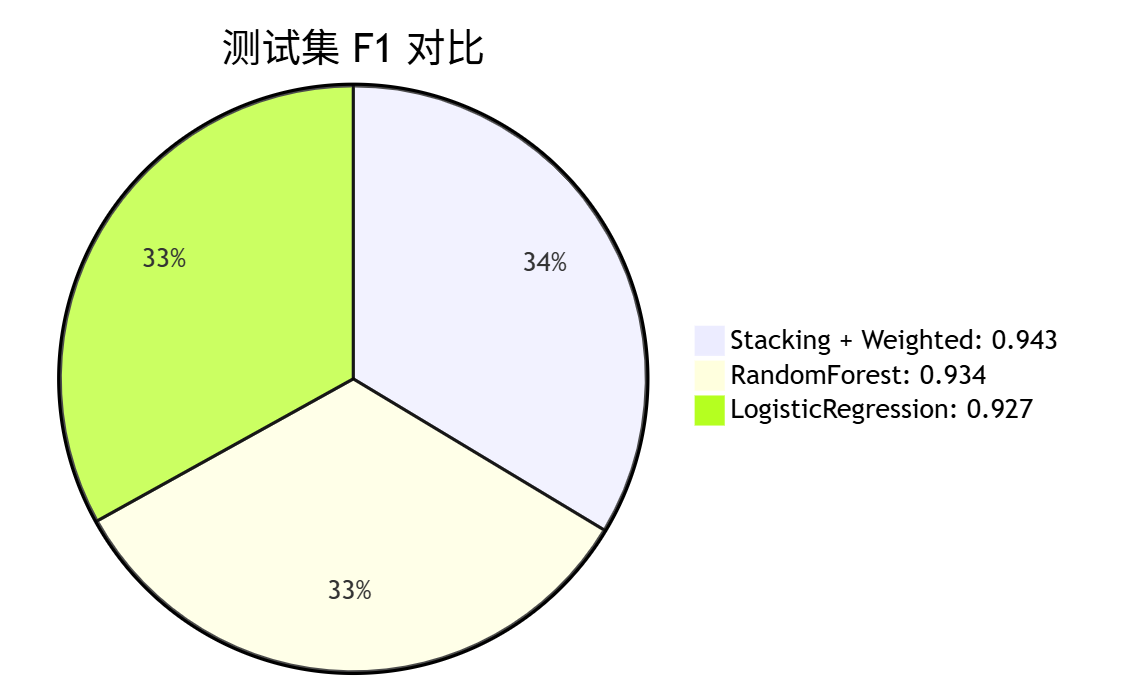


图 5‑3融合性能提升饼图（左）多种融合策略性能提升对比（右）

对于17657个样本训练的模型，训练的准确率达到98.61%，对于预测结果而言，仅存在2个假阳性样本，准确率达99.99%，F1-score达99.98%，如图 5‑4所示。

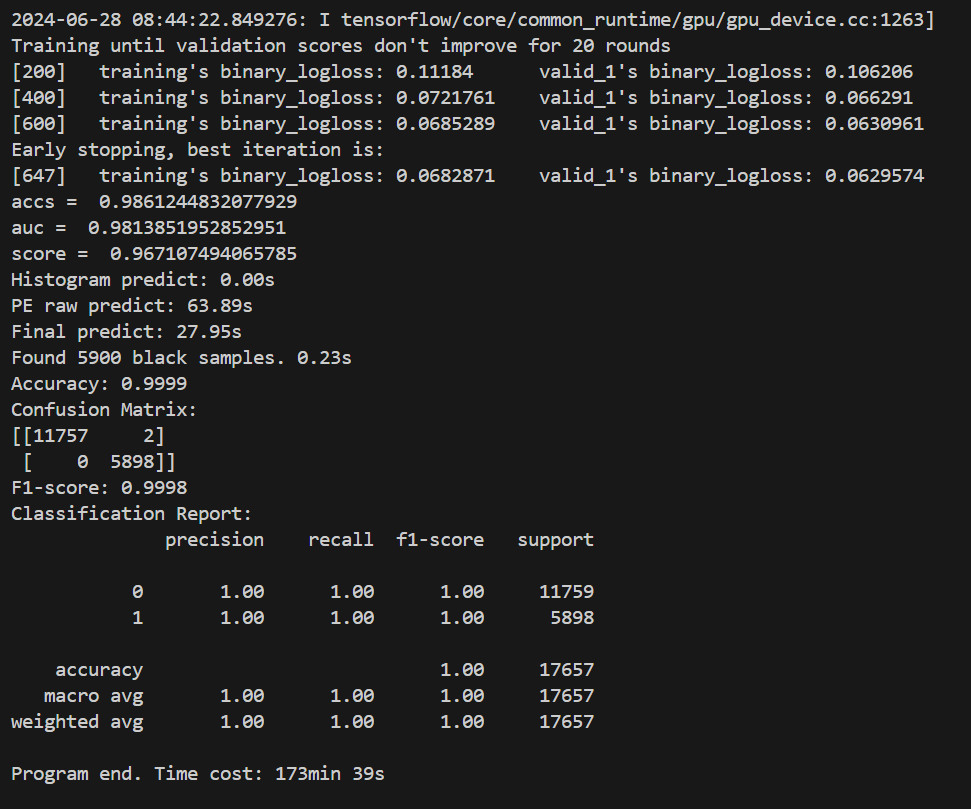


图 5‑4最终模型性能表现情况

## 本章小结

本章设计了基于Stacking和加权融合的多层集成学习方案，通过子模型并行训练、OOF预测生成和元模型训练等步骤，实现了对恶意PE软件的高效识别。该方案充分利用了不同模型的优势，通过加权融合进一步优化了检测结果。实验结果表明，该方案在检测准确率和鲁棒性方面表现出色，能够有效应对恶意软件的复杂变种，为后续系统的实现提供了可靠的模型基础。

# 基于集成学习的恶意PE识别系统的设计与实现

## 系统需求及可行性分析

### 系统需求分析

#### 需求分析概述

本系统设计以集成学习的恶意PE特征检测和识别技术为核心，结合第三章的特征检测和提取方法，以及第四章、第五章的多模型融合策略，构建了一个面向恶意软件分析的综合系统。系统要求其架构按照功能分为四部分：前端页面、FastAPI后端、SQLite数据库、特征提取/集成学习模块，前端页面部分需要负责用户交互与数据可视化等功能，后端部分需要实现样本处理、特征提取、模型推理、数据库管理等功能，数据库将用于存储样本元信息与分析结果，特征提取与集成学习模块需要能够通过多模型协作实现恶意性判别。

系统总体架构通过模块间API调用与数据流交互实现功能耦合，形成完整的恶意样本分析闭环，其需求架构如图 6‑1所示。

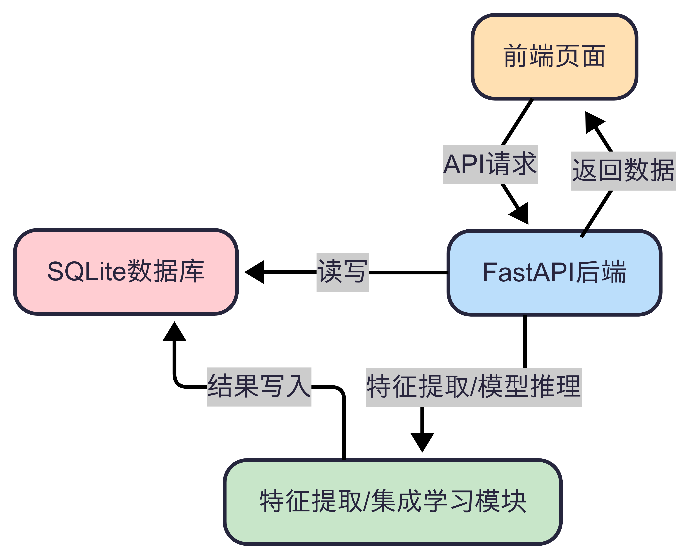


图 6‑1系统总体需求架构示意图

#### 功能性需求分析

功能性需求上，系统要求围绕样本分析进行基本功能的实现。在用户上传PE样本后，后端需自动触发多维度特征提取过程，包括字节直方图、PE静态特征、特征工程数据、LIEF深度解析等结果。随后，提取的特征需分别输入至对应的机器学习模型，并通过加权融合策略生成最终的恶意性判定，这部分依赖于具体的集成学习策略实现。分析结束后，需要将所有的特征数据、分析结果需持久化存储至SQLite数据库，并在前端应支持实时查询与可视化展示，例如样本列表检索、特征分布雷达图等功能。

流程设计需确保特征提取、模型推理与结果融合的高效性与一致性，同时满足用户对样本详情的多维度分析需求。具体功能流程要求如图 6‑1所示。

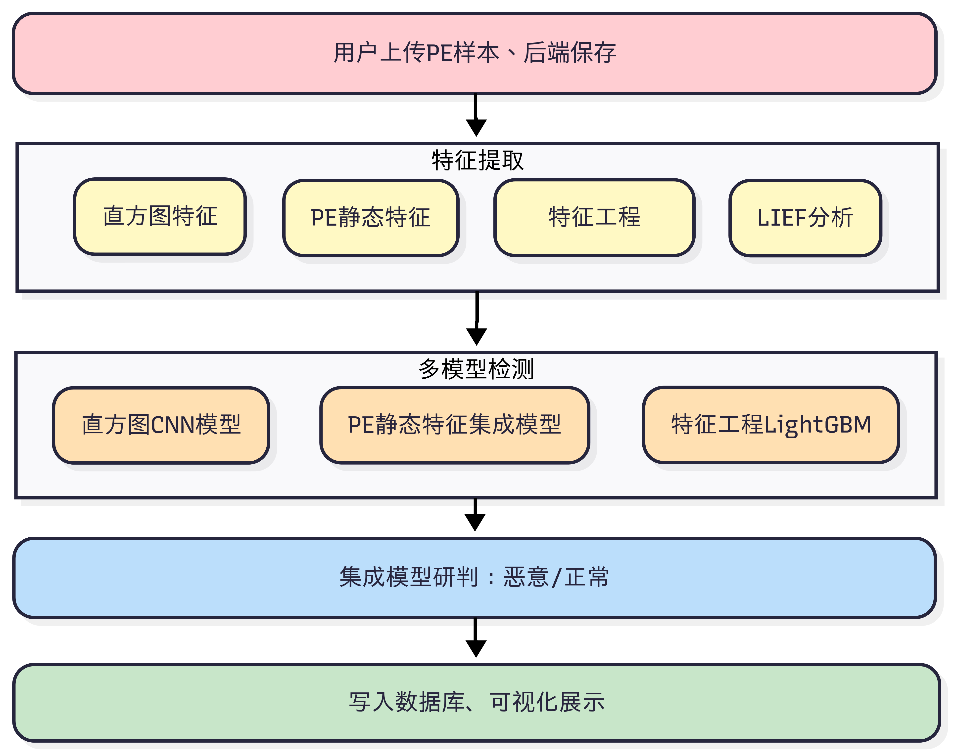


图 6‑1 PE软件样本检测流程

#### 非功能性需求分析

非功能性需求上，系统需具备良好的扩展性与稳定性，以支持未来新增特征提取方法或模型算法的集成。具体而言，在响应性能方面，系统需要满足中等规模样本处理场景下的实时性要求；在数据存储方面，需要保证样本信息的完整性与安全性，避免敏感数据泄露；在用户交互上，需要遵循直观化设计原则，降低非技术人员的操作门槛。

### 系统可行性分析

#### 技术可行性

技术可行性方面，本系统采用成熟的前后端分离架构，通过React和FastAPI轻量级框架，结合SQLite数据库与集成学习技术，能够有效支撑恶意PE分析的复杂需求。

具体而言，后端FastAPI组件提供高效的API服务与异步处理能力；前端React与Ant Design组件库能够确保前端交互的流畅性；SQLite数据库的轻量特性适配中小型数据存储场景，技术实现轻松；集成学习模块通过多模型堆叠与加权融合策略，已在第四章、第五章中实验中验证了其对恶意样本的高判别准确率，技术方案具备可实施性。

#### 经济可行性

经济可行性方面，本系统基于开源工具与框架构建，在系统成本上要求低。另外，对于SQLite数据库的免安装特性与低资源消耗进一步降低了硬件部署费用，系统中的模块化设计也能够减少后期维护与功能扩展的开发投入，整体开发与运营成本可控。

#### 操作可行性

操作可行性上，本系统通过图形化界面与API接口实现用户与开发者角色的分离，非技术用户可通过前端完成样本上传与结果查询，开发者则通过后端接口与数据库实现系统功能的迭代优化，操作流程符合实际应用场景的需求。

## 系统总体设计

### 系统总体架构

本系统采用前后端分离架构，通过模块化设计实现功能的前后端统一流程。系统前端基于React框架与UmiJS路由管理，结合Ant Design组件库与ECharts可视化工具，构建用户交互界面，支持样本上传、特征展示、模型分析与数据可视化等核心功能。系统后端采用FastAPI框架，提供RESTful API接口，负责样本文件的接收与存储、特征提取任务的调度、模型推理结果的生成、数据库操作的协调等功能。SQLite数据库作为系统数据中枢，存储样本元信息、样本特征数据、样本分析记录、样本判别结果等内容。

特征提取与集成学习模块由后端统一调度，采用分阶段流水线处理模式。系统首先会通过多进程技术，并行执行字节直方图计算、PE静态特征解析、特征工程处理、LIEF深度分析等功能；随后将提取的特征数据输入至对应的机器学习模型（本系统中为CNN、集成模型、LightGBM），通过加权融合策略生成最终判别结果。模块设计注重代码复用性与可扩展性，例如通过封装特征提取函数实现不同特征维度的灵活配置，通过模型堆叠策略支持新增算法的快速集成。

系统交互流程遵循API驱动模式，在系统前端通过HTTP请求调用后端接口，后端接收请求后触发特征提取与模型推理任务，最终将分析结果返回前端并持久化至数据库。模块间通过JSON格式数据传递信息，确保前后端通信的兼容性与高效性。整体架构通过分层设计与松耦合原则，既满足实时性要求，又为后续功能扩展预留了技术空间。

系统总体架构序列图如图 6‑1所示。

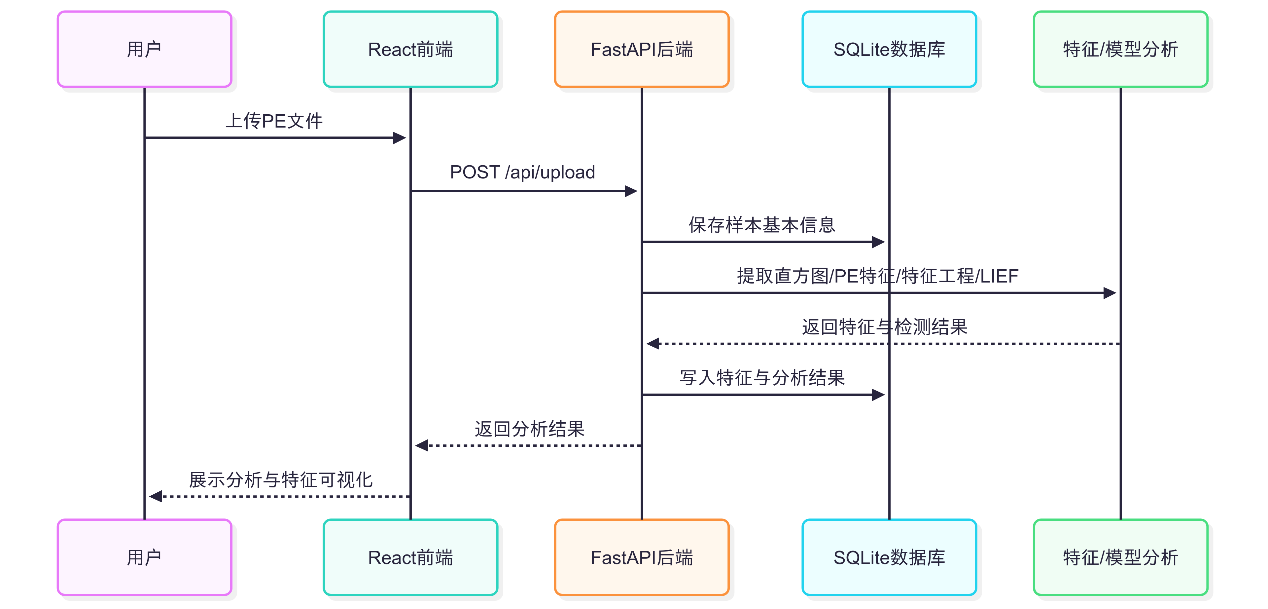


图 6‑1系统总体架构序列图

### 系统数据库设计

系统所用数据表包括样本信息表、直方图特征表、PE静态特征表、特征工程表和LIEF分析表等核心数据表，通过样本ID建立外键关联关系，形成完整的恶意软件分析数据体系。本小节将详细展示系统数据库设计中，各表的详细信息。

#### 样本信息表（samples）

样本信息表存储样本文件的基础元数据和分析核心指标，记录样本上传、哈希校验、恶意判定等样本特征信息，具体如表 6‑1所示。

表 6‑1样本信息表

| **编号** | | **字段名** | | **数据类型** | | **非空** | | **主键** | | **注释** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id | | INTEGER | | 否 | | 是 | | 样本ID | |
| 2 | file\_hash | | TEXT | | 是 | | 否 | | 文件MD5哈希 | |
| 3 | file\_name | | TEXT | | 否 | | 否 | | 文件名 | |
| 4 | file\_size | | INTEGER | | 否 | | 否 | | 文件大小（字节） | |
| 5 | is\_malicious | | INTEGER | | 否 | | 否 | | 是否恶意(0/1) | |
| 6 | analysis\_time | | TIMESTAMP | | 否 | | 否 | | 分析时间 | |
| 7 | classification\_result | | TEXT | | 否 | | 否 | | 检测结果文本 | |

#### 直方图特征表（histogram\_features）

直方图特征表存储二进制文件的结构化统计特征，包含字节分布直方图和熵值分布直方图两种特征数据，用于可视化分析和机器学习特征提取，具体如表 6‑2所示。

表 6‑2直方图特征表

| **编号** | **字段名** | **数据类型** | **非空** | **主键** | **注释** |
| --- | --- | --- | --- | --- | --- |
| 1 | id | INTEGER | 否 | 是 | 主键 |
| 2 | sample\_id | INTEGER | 否 | 否 | 样本ID外键 |
| 3 | byte\_histogram | TEXT | 否 | 否 | 字节直方图JSON |
| 4 | entropy\_histogram | TEXT | 否 | 否 | 熵直方图JSON |

#### PE静态特征表（pe\_features）

PE静态特征表记录PE文件格式的深度解析结果，包含文件头信息、节区特征、导出表等Windows可执行文件特有结构数据，支持静态特征分析，具体如表 6‑3所示。

表 6‑3 PE静态特征表

| **编号** | **字段名** | **数据类型** | **非空** | **主键** | **注释** |
| --- | --- | --- | --- | --- | --- |
| 1 | id | INTEGER | 否 | 是 | 主键 |
| 2 | sample\_id | INTEGER | 否 | 否 | 样本ID外键 |
| 3 | general\_info | TEXT | 否 | 否 | 常规信息JSON |
| 4 | header\_info | TEXT | 否 | 否 | PE头信息JSON |
| 5 | section\_info | TEXT | 否 | 否 | 节区信息JSON |
| 6 | exports\_info | TEXT | 否 | 否 | 导出表JSON |

#### 特征工程表（engineered\_features）

集成多维加工特征，包括节区特征、字符串匹配、Yara规则匹配、操作码序列等机器学习特征，支撑模型训练和预测等功能，具体表 6‑4如所示。

表 6‑4特征工程表

| **编号** | **字段名** | **数据类型** | **非空** | **主键** | **注释** |
| --- | --- | --- | --- | --- | --- |
| 1 | id | INTEGER | 否 | 是 | 主键 |
| 2 | sample\_id | INTEGER | 否 | 否 | 样本ID外键 |
| 3 | section\_features | TEXT | 否 | 否 | 节区特征JSON |
| 4 | string\_match | TEXT | 否 | 否 | 字符串匹配JSON |
| 5 | yara\_match | TEXT | 否 | 否 | Yara匹配JSON |
| 6 | string\_count | TEXT | 否 | 否 | 关键字统计JSON |
| 7 | opcode\_features | TEXT | 否 | 否 | Opcode特征JSON |

#### LIEF分析表（lief\_features）

通过LIEF框架提取的进阶PE特征，包含导入表、资源信息、TLS等深度解析数据，用于增强分析、威胁情报提取等功能，具体如表 6‑5所示。

表 6‑5 LIEF分析表

| **编号** | **字段名** | **数据类型** | **非空** | **主键** | **注释** |
| --- | --- | --- | --- | --- | --- |
| 1 | id | INTEGER | 否 | 是 | 主键 |
| 2 | sample\_id | INTEGER | 否 | 否 | 样本ID外键 |
| 3 | dos\_header | TEXT | 否 | 否 | DOS头JSON |
| 4 | pe\_header | TEXT | 否 | 否 | PE头JSON |
| 5 | sections | TEXT | 否 | 否 | 节区信息JSON |
| 6 | imports | TEXT | 否 | 否 | 导入表JSON |
| 7 | tls\_info | TEXT | 否 | 否 | TLS信息JSON |
| 8 | resources | TEXT | 否 | 否 | 资源信息JSON |

### 系统API接口设计

系统采用RESTful风格设计前后端交互接口，通过统一化的API设计实现核心业务功能。

主要API接口详情表如表 6‑6所示，定义了系统核心业务接口规范，涵盖样本管理、数据分析、模型监控等功能点。

表 6‑6主要API接口详情表

| 接口路径 | 方法 | 参数 | 返回内容 | 说明 |
| --- | --- | --- | --- | --- |
| /api/upload | POST | file | {sample\_id, ...} | 上传并分析样本 |
| /api/samples | GET | - | 样本列表 | 获取所有样本 |
| /api/samples/{id} | GET | id | 样本详细信息 | 获取单个样本所有特征 |
| /api/stats | GET | - | 统计信息 | 样本总数、恶意数等 |
| /api/model/analysis | GET | - | 模型分析数据 | 各模型性能、特征重要性 |

功能点接口总览表展示了系统功能模块与接口、前端组件的映射关系，体现前后端协同工作机制，具体如表 6‑7所示。

表 6‑7功能点接口总览表

| 功能点 | 后端接口 | 前端页面/组件 | 说明 |
| --- | --- | --- | --- |
| 样本上传 | POST /api/upload | /upload, UploadPage | 上传PE文件并自动分析 |
| 样本列表 | GET /api/samples | /samples, SamplesPage | 展示所有样本，支持搜索 |
| 样本详情 | GET /api/samples/{id} | /samples/detail, SampleDetailPage | 展示单个样本所有特征与分析 |
| 仪表盘 | GET/api/samples, /api/stats | /dashboard, DashboardPage | 样本统计、分布、趋势 |
| 特征可视化 | - | PEFeaturesCard, LiefAnalysisCard等 | 直方图、雷达图、表格等 |

## 系统详细设计及功能实现

### 系统功能概述

本系统主要包含样本管理模块、特征提取模块、集成学习与检测模块、数据可视化与前端展示模块等四个模块，通过菜单、按钮、用户信息引导等多种方式，能够让用户定位系统功能点，如图 6‑4所示。本节后续小节将逐一展开各功能模块的详细设计及实现的说明。



图 6‑4系统入口页面展示

### 样本管理模块

样本管理模块是系统的数据基础与交互入口，负责PE样本的上传、分析、列表、搜索、查看详情等功能实现。

#### 样本上传和分析功能

该功能实现了样本数据的上传和分析，它允许用户通过前端界面，选择本地PE文件进行上传。文件支持.exe, .dll, .sys等格式，文件选定后，前端触发对后端POST /api/upload接口的调用。后端FastAPI服务接收到文件流后，首先计算文件的MD5哈希值以作唯一标识，并将文件内容存储于服务器指定目录。随后，后端调用insert\_sample函数将样本的基本元数据（文件名、哈希、大小等）存入数据库samples表。紧接着，后端自动触发特征提取流程，并将最终的检测结果更新回数据库。整个过程通过Steps组件实现的前端步骤条，向用户展示进度，分析完成后提示用户并提供查看详情的链接。

样本上传页面如图 6‑5所示。

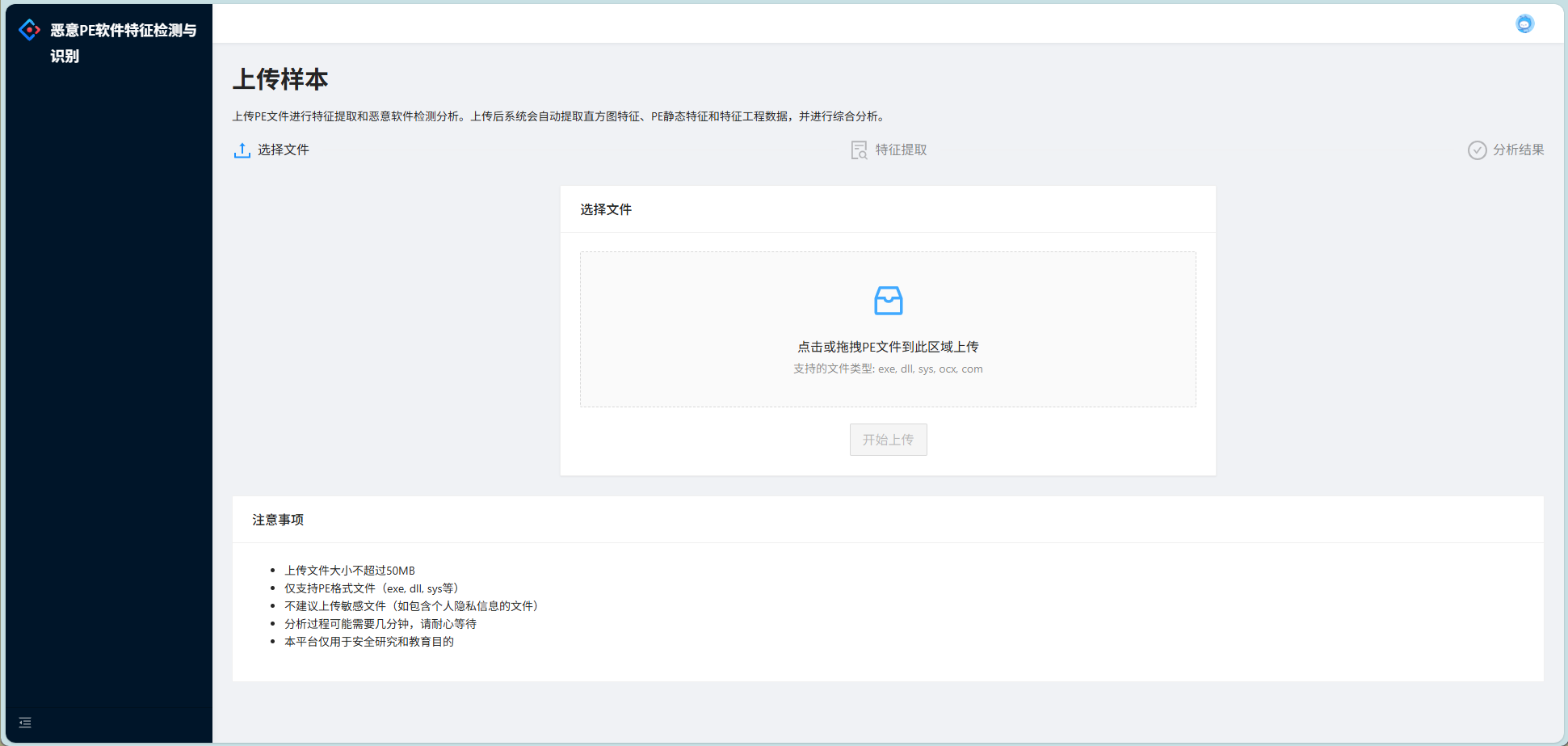


图 6‑5样本上传页面展示

分析完成后Steps组件进度将更新，并提示用户查看分析详情结果，具体如图 6‑6所示。

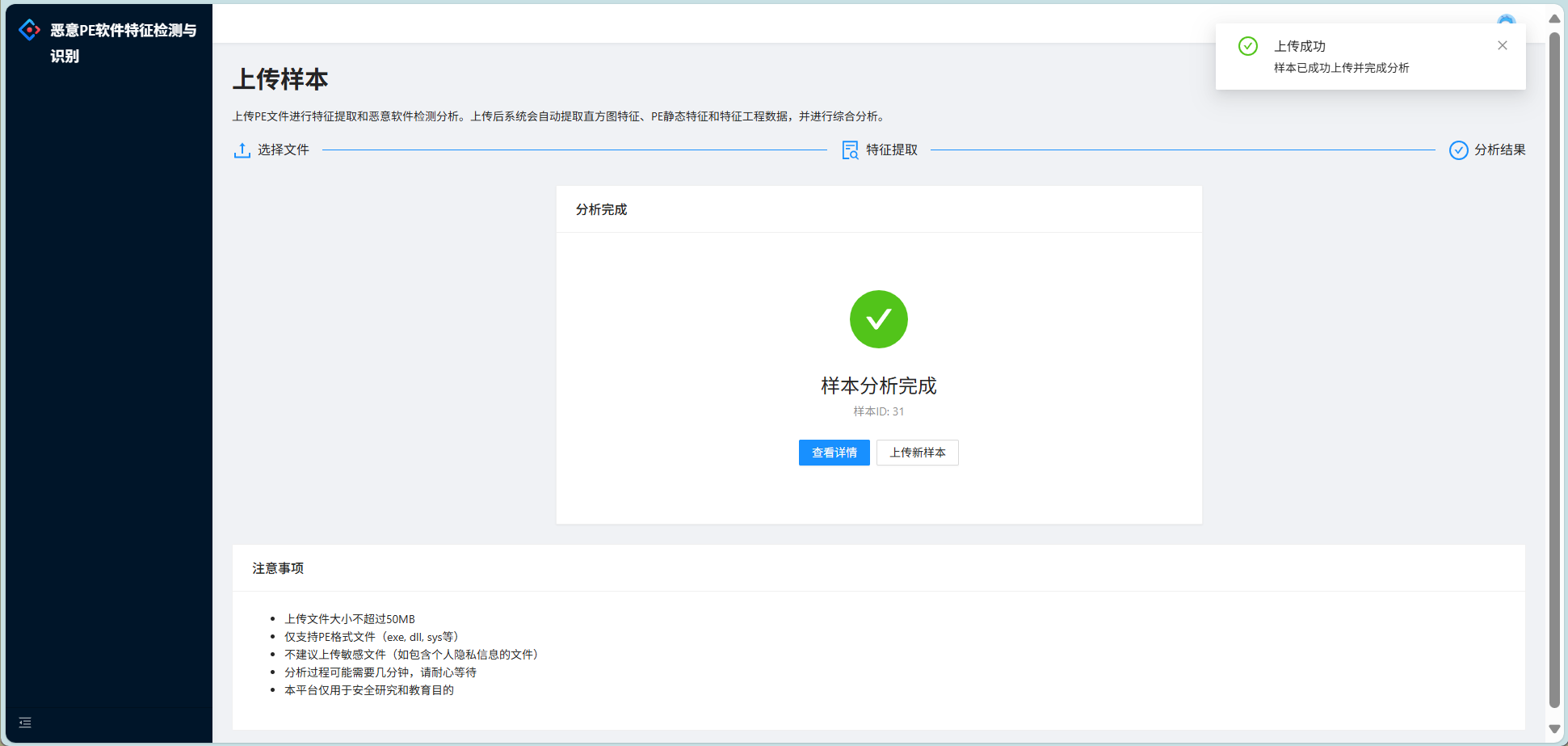


图 6‑6样本分析完成页面展示

#### 样本列表与搜索功能

该功能提供了对已上传样本的集中管理视图，记录系统中样本的所有提交情况，在后台可查看已提交的样本列表。样本列表通过Ant Design的表格组件展示了样本的基本分析结果，包含ID、文件名、哈希、大小、分析时间和检测结果等关键字段，表格支持分页，支持按文件大小、分析时间等字段排序查看。若想要查看样本详细信息，点击相应的记录可切换到对应的特征提取和分析结果页面，如图 6‑7所示。

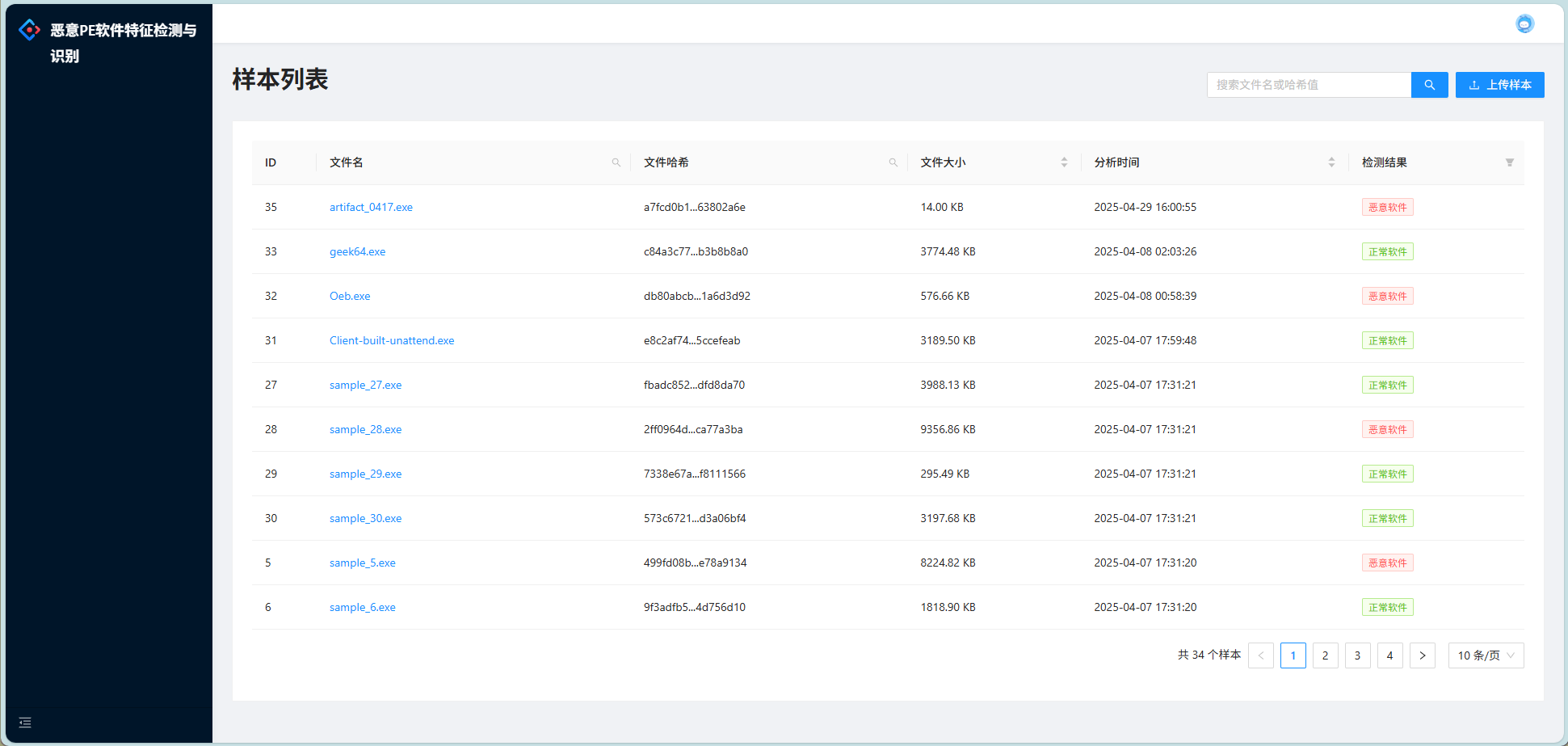


图 6‑7样本列表页面展示

此外，页面提供搜索框，允许用户输入文件名或MD5哈希值进行快速过滤。在查找框中可以输入具体的值进行搜索，快速查找到所需要查看的样本，如图 6‑8所示。

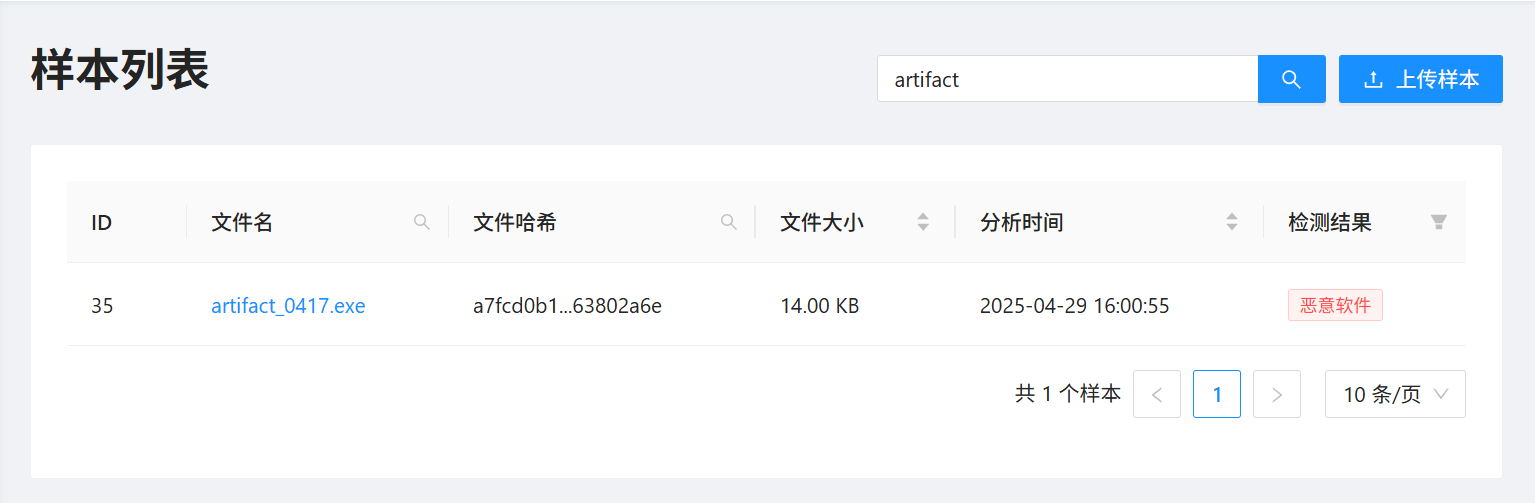


图 6‑8样本列表搜索功能展示

#### 样本详情功能

该功能可以查看单个样本的分析结果，当用户在列表页点击特定样本时，前端路由跳转至详情页，并将样本ID作为参数。详情页通过调用GET /api/samples/{id}接口，从数据库获取该样本的完整信息，包括基础信息以及关联的histogram\_features、pe\_features、engineered\_features、lief\_features等特征数据。

前端使用Tabs组件将这些信息分门别类地展示，每个标签页下利用专门设计的React组件，对相应的特征数据进行结构化呈现和可视化，便于用户全面查看样本的相关分析结果，如图 6‑9所示。

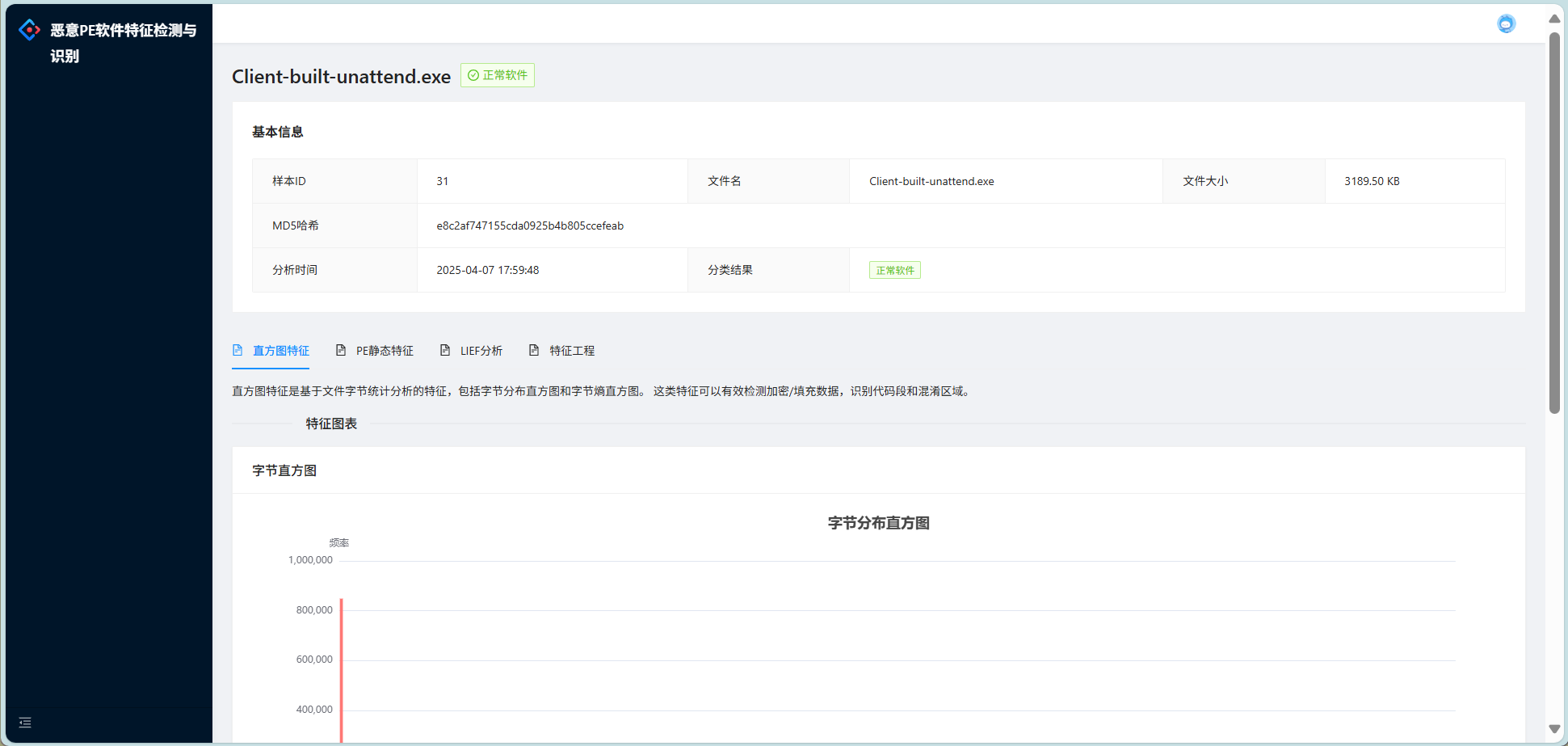


图 6‑9样本分析结果详情页面展示

### 特征提取模块

特征提取模块是主要负责从原始PE样本中抽取多维度、多层次的特征信息，为后续的集成学习检测和人工分析提供数据支撑。该模块主要包括直方图特征展示、PE静态特征展示、LIEF深度分析、特征工程分析等部分。

#### 直方图特征展示

主要计算两种直方图，包括字节分布直方图和字节熵直方图。字节直方图能够统计0-255每个字节值在整个文件中的出现频率，归一化后得到256维向量；字节熵直方图，通过滑动窗口计算局部熵值，并统计不同熵值区间的字节分布，生成256维向量。这两种特征融合为512维向量，对检测文件是否经过加密、压缩等情况具有重要意义，它们直接作为后续卷积神经网络（CNN）模型的输入。特征向量在存入histogram\_features表前会进行JSON序列化。

前端通过Echart组件库进行可视化展示，字节直方图中根据字节的范围划分了三种颜色，可显字符即ASCII码范围内的字符用绿色表示，剩下的部分包括低字节区域和高字节区域，分别用红色和蓝色表示，具体如图 6‑10所示。



图 6‑10字节直方图特征分析结果

字节熵直方图中根据字节熵值分布进行展示，根据熵值高低，通过红、橙、绿三种颜色直观展示大小变化，具体如图 6‑11所示。

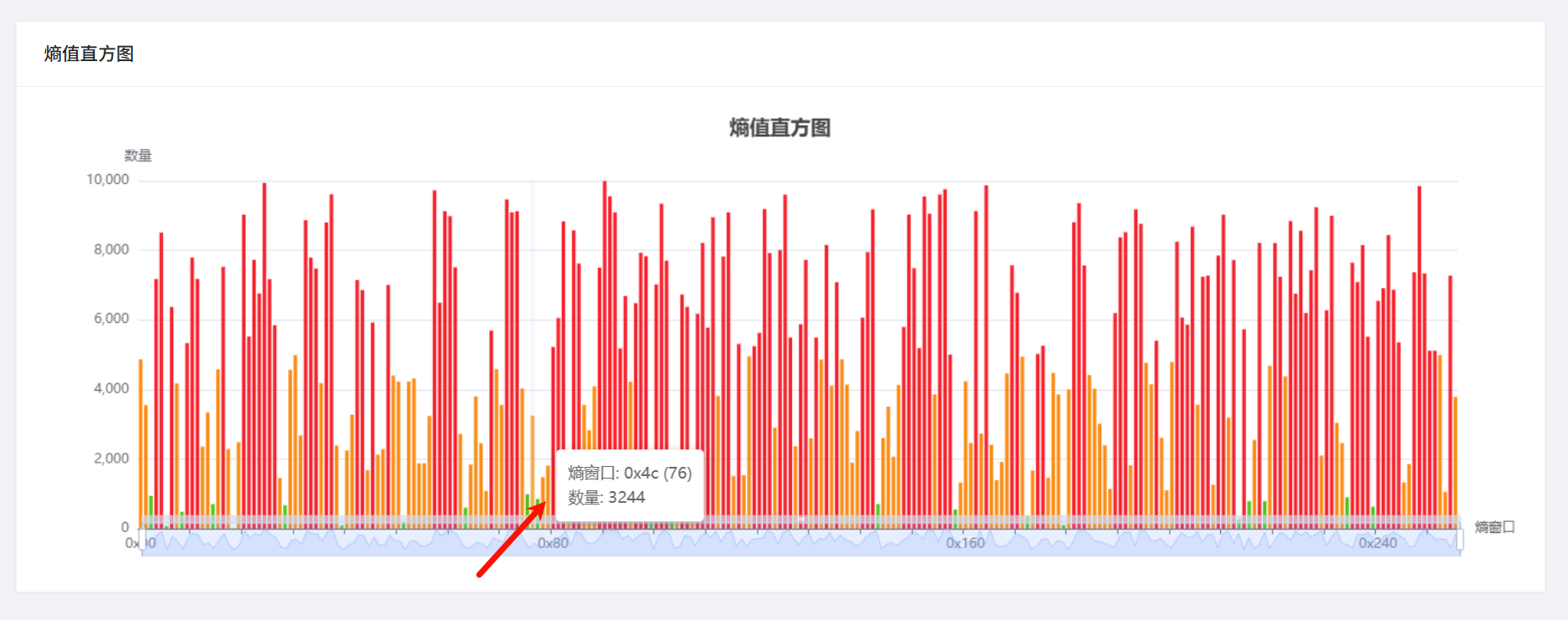


图 6‑11字节熵直方图特征分析结果

#### PE静态特征展示

该部分通过解析PE静态特征提取相关的文件特征，包括常规信息、PE头信息、结区信息、导入导出函数等字段。分析结束后，系统会将提取到的信息存储于pe\_features表中。具体展示如图 6‑12所示。

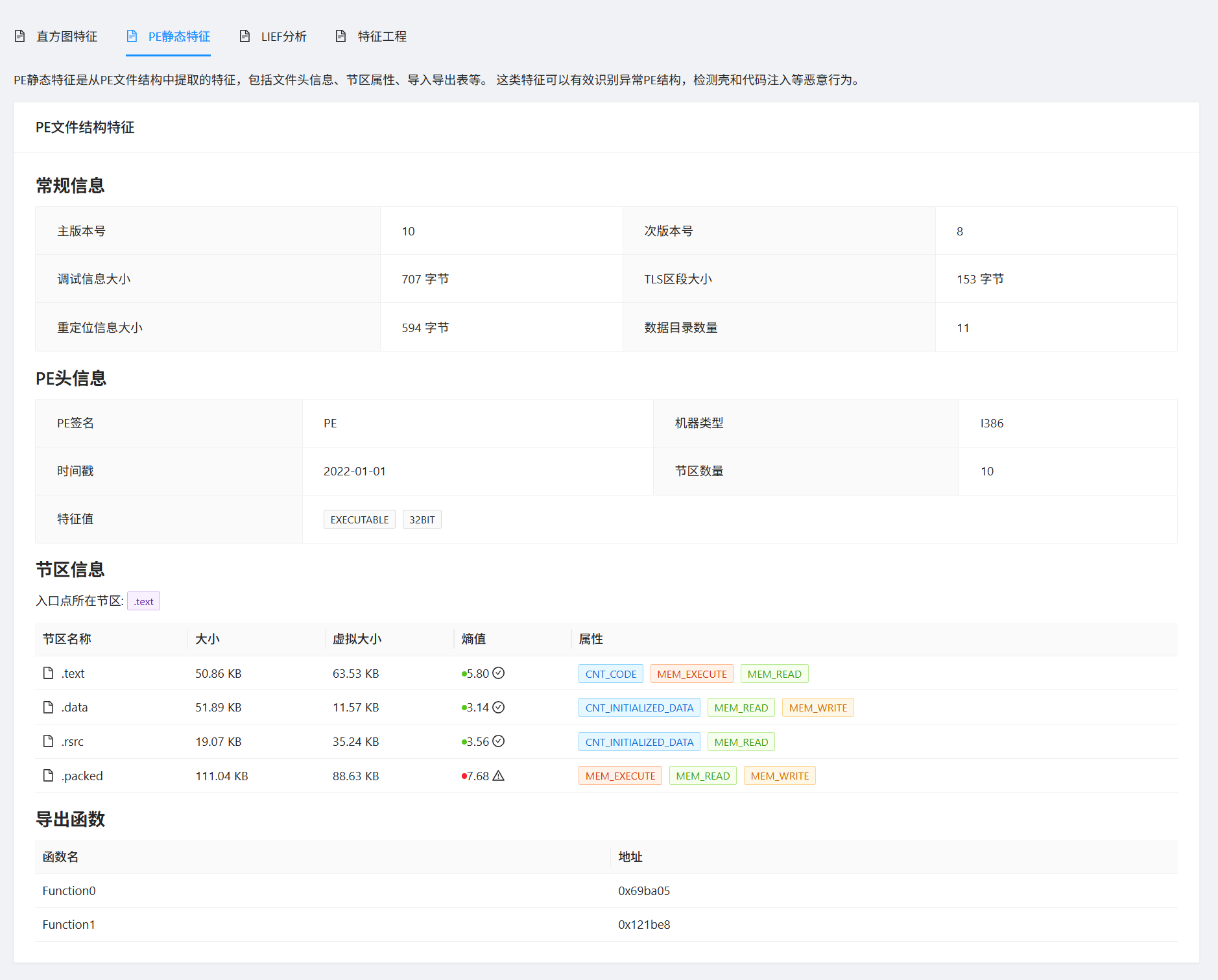


图 6‑12 PE静态特征分析结果

#### LIEF深度分析

该部分主要对PE静态特征提取进行补充和验证，提供了未经机器学习处理的、更原始和详尽的PE结构信息。基于LIEF分析工具的解析结果，系统将解析出的DOS头、标准PE头、可选PE头、详细节区列表、导入表、导出表、TLS结构、资源目录树等信息进行结构化整理，并以JSON格式直接存入lief\_features数据库表。具体页面展示如图 6‑13所示。

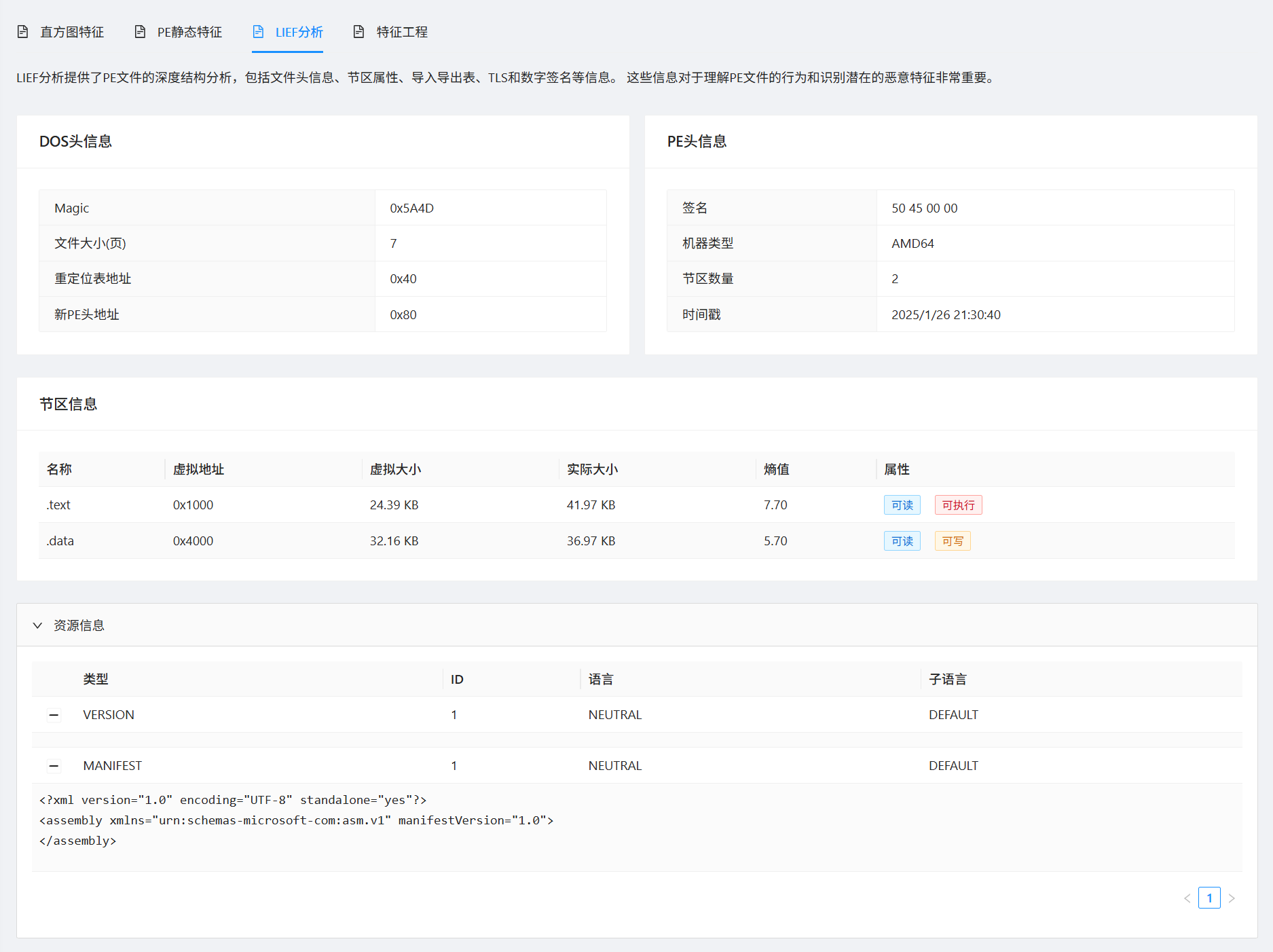


图 6‑13 LIEF特征深度分析结果

#### 特征工程分析

这部分主要包括节区信息统计、字符串模式匹配、Yara规则匹配、关键字扫描、Opcode分析五类特征工程分析部分。

节区信息统计主要基于LIEF分析结果，计算入口点节区名长度、可读/写/执行节区的大小和熵的均值、资源节区数量、节区总数等特征；字符串模式匹配利用正则表达式匹配常见的可疑字符串，如加密货币钱包地址、文件路径、注册表项、URL、IP地址、指示性标记等特征；Yara规则匹配通过预置的通用恶意规则和加壳规则集，统计匹配到的规则数量；关键字扫描通过统计特定敏感词汇的出现次数，如常见杀毒软件进程名、调试器名、矿池域名、加密算法名、虚拟货币名等进行特征分析分析；Opcode分析通过正则表达式定位函数边界，使用capstone等反汇编引擎提取函数内的操作码序列，并计算其统计特征。这五部分特征被组合成一个56维的向量，专门用于驱动LightGBM等对高阶特征敏感的模型，结果存储在engineered\_features表中。具体如图 6‑14所示。



图 6‑14特征工程分析结果

### 集成学习与检测模块

集成学习与检测模块负责利用前一阶段提取的多维度特征，通过集成学习模型对PE样本进行恶意性判别。为了充分发挥不同特征类型的优势并提高检测的鲁棒性和准确性，系统采用了复杂的多层集成学习架构。该架构的核心思想是并行处理异构特征，最后通过策略融合各部分的预测结果。

根据第五章的模型实现方案，最终实现了恶意样本的特征检测和识别判别。具体结果包括模型特征分布雷达图，计算了其特征得分和判别结果，如图 6‑15所示。



图 6‑15模型特征得分计算

### 数据可视化与前端展示模块

数据可视化与前端展示模块丰富了系统与用户的交互反馈，将后台复杂的分析结果以直观、易懂、可交互的方式呈现给用户。该模块基于React框架，利用UmiJS进行路由和状态管理，结合Ant Design组件库构建美观统一的UI界面，同时大量采用ECharts图表库实现丰富的数据可视化效果。

#### 仪表盘

该部分为系统的概览入口。它通过调用GET /api/stats和GET /api/samples接口获取全局统计数据和近期样本信息。页面顶部使用Ant Design的Statistic卡片展示核心指标，如总样本数、恶意样本数、正常样本数和整体检测率。中部利用ECharts绘制两个核心图表：一个为样本分类饼图，展示恶意与正常样本的比例分布；另一个为样本大小分布柱状图，显示不同文件大小区间的样本数量。页面底部展示一个近期上传样本的简略列表，方便用户快速了解最新动态。具体如图 6‑16所示。

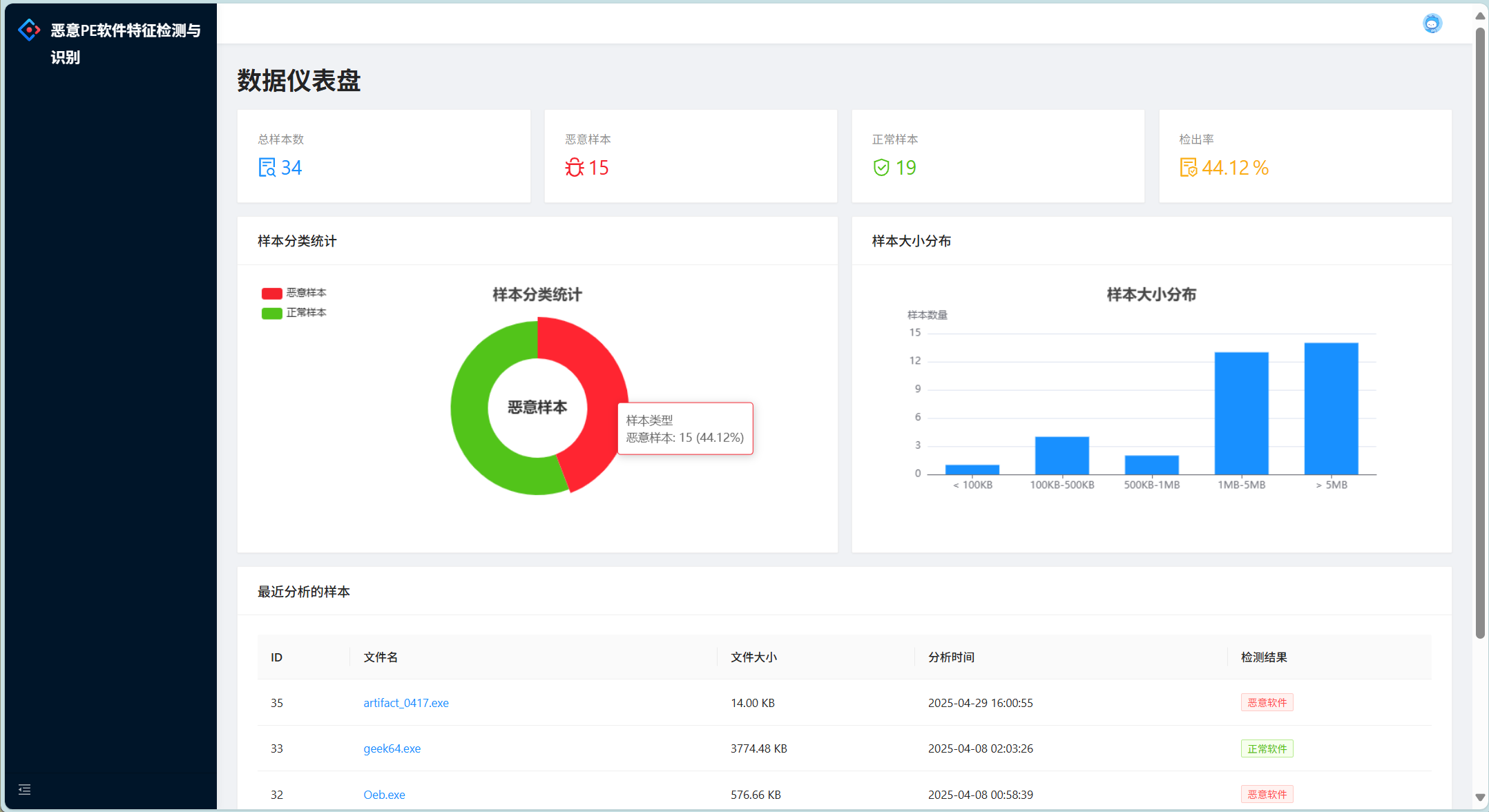


图 6‑16数据仪表盘页面展示

#### 特征可视化组件

该部分主要用于构成样本详情页中的图表展示，除了6.3.3节特征的基本展示外，还包括对这些特征的可视化展示。针对样本节区特征，这里根据内存属性（读/写/执行）使用不同颜色的Tag，并通过柱状图和饼图展示其特征分布和权重分布，具体如图 6‑17所示。



图 6‑17节区特征分析图表可视化

针对字符串匹配、关键字统计结果，系统会以列表结合柱状图的形式清晰呈现其特征匹配的计数和均值，具体如图 6‑18所示。

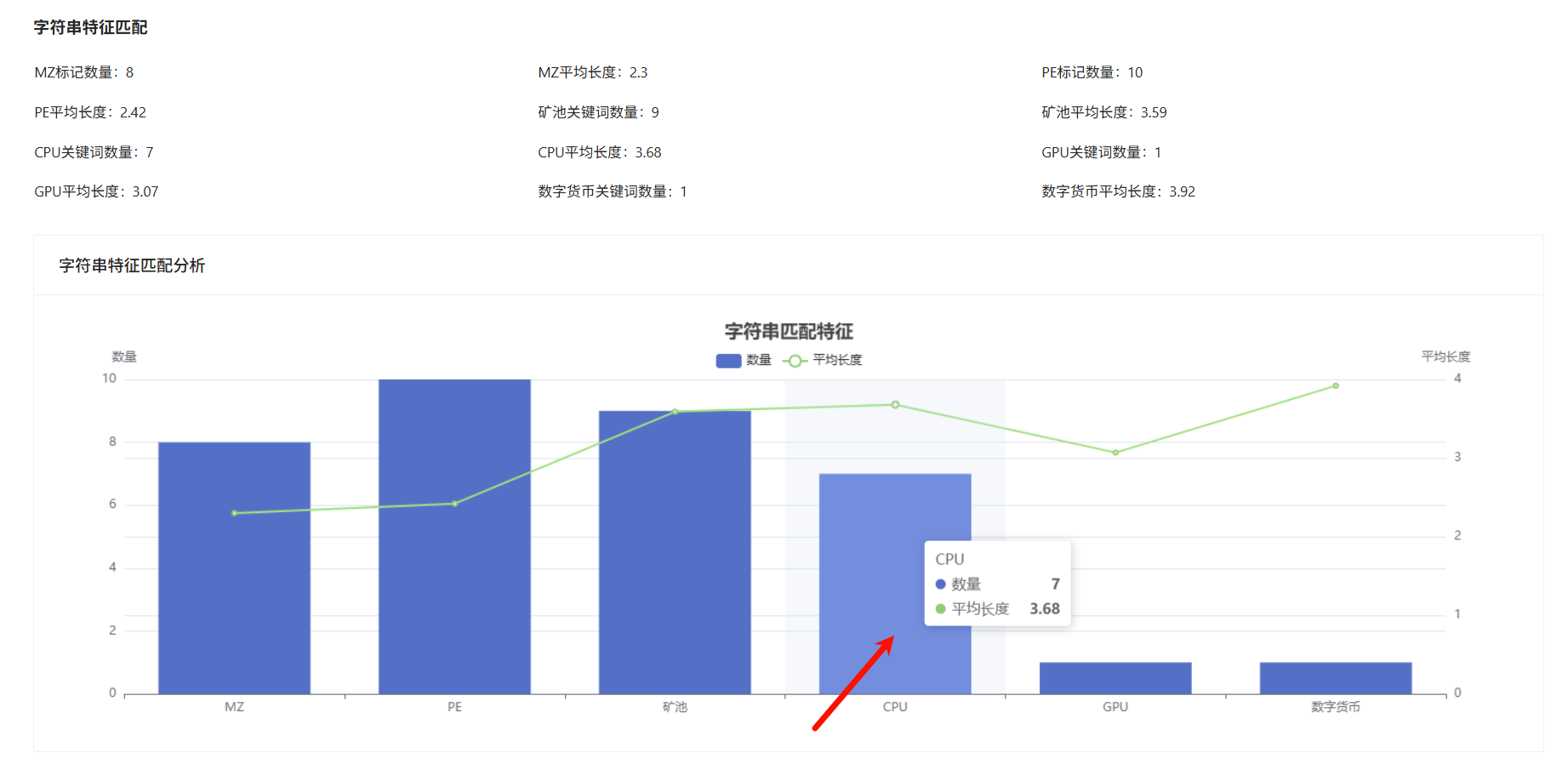


图 6‑18字符串匹配特征分析图表可视化

## 系统测试

### 系统测试环境

为了进一步验证系统功能的完整性和使用的可靠性，本文对上述系统进行测试。测试环境统一为如下配置：

（1） 软件环境方面，后端基于FastAPI 0.115.12运行，采用React 17.0、umi 3.5.20框架构建Web服务，数据存储通过SQLite3数据库实现。

（2） 前端测试覆盖主流浏览器，包括Google Chrome和Mozilla Firefox。

（3） 硬件配置方面，本次测试使用CPU为12th Gen Intel(R) Core(TM) i7-12650H 2.30GHz、GPU为Nvidia RTX4060/32GB\*1的计算机，并保持稳定的网络连接。测试环境中，操作系统选择Windows 10和Ubuntu 22.04，验证跨平台兼容性。

### 系统测试方案

为验证基于集成学习的恶意PE识别系统功能与稳定性，本文进行了全面测试，涵盖样本管理、特征提取、集成学习与检测以及数据可视化等关键模块。测试结果显示，样本管理模块可成功实现样本上传、列表展示、搜索及详情查看等功能，支持用户上传PE样本并自动触发特征提取流程。特征提取模块能高效提取字节直方图、PE静态特征、LIEF深度分析结果及特征工程生成的高级特征，并准确存储至数据库。集成学习与检测模块将特征数据输入多种机器学习模型，通过加权融合策略生成恶意性判定结果，验证了系统的高效性和准确性。数据可视化模块则通过图表库将复杂分析结果直观呈现，用户可在样本详情页查看可视化图表，在仪表盘了解全局数据及模型分析结果。详细的测试用例及其测试结果见表 6‑8。

表 6‑8系统测试用例及其测试结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **编号** | **被测模块** | **测试项** | **系统初**  **始状态** | **输入** | **操作**  **步骤** | **期望**  **输出** | **测试**  **结果** |
| 1 | 样本管理模块 | 样本上传功能 | 系统启动，用户登录成功 | 一个合法的PE样本文件 | 用户通过前端页面选择文件并点击上传按钮 | 系统接收文件，计算MD5哈希值，存储文件并返回上传成功提示，自动触发特征提取流程 | 通过 |
| 2 | 样本管理模块 | 样本列表展示 | 系统启动，用户登录成功，已上传多个样本 | 无 | 用户在样本列表页面点击刷新按钮 | 系统展示所有已上传样本的列表，包括样本ID、文件名、哈希值、大小、分析时间和检测结果等信息 | 通过 |
| 3 | 样本管理模块 | 样本搜索功能 | 系统启动，用户登录成功，已上传多个样本 | 搜索关键词 | 用户在样本列表页面的搜索框中输入关键词并点击搜索按钮 | 系统根据关键词过滤样本列表，只显示匹配的样本信息 | 通过 |
| 4 | 样本管理模块 | 样本详情查看 | 系统启动，用户登录成功，已上传多个样本 | 样本ID | 用户在样本列表页面点击某个样本的详情链接 | 系统展示该样本的详细信息，包括基础信息、直方图特征、PE静态特征、特征工程数据、LIEF深度解析结果等 | 通过 |
| 5 | 特征提取模块 | 字节直方图计算 | 系统启动，用户上传了一个PE样本 | 上传的PE样本文件 | 系统自动触发特征提取流程 | 系统计算出字节分布直方图和字节熵直方图，并将结果存储到数据库中 | 通过 |
| 6 | 特征提取模块 | PE静态特征解析 | 系统启动，用户上传了一个PE样本 | 上传的PE样本文件 | 系统自动触发特征提取流程 | 系统解析出PE文件的静态特征，包括文件头信息、节区特征、导出表等，并将结果存储到数据库中 | 通过 |
| 7 | 特征提取模块 | LIEF深度分析 | 系统启动，用户上传了一个PE样本 | 上传的PE样本文件 | 系统自动触发特征提取流程 | 系统通过LIEF框架提取出PE文件的深度特征，包括导入表、资源信息、TLS等，并将结果存储到数据库中 | 通过 |
| 8 | 特征提取模块 | 特征工程处理 | 系统启动，用户上传了一个PE样本 | 上传的PE样本文件 | 系统自动触发特征提取流程 | 系统进行特征工程处理，生成节区特征、字符串匹配、Yara规则匹配、关键字扫描、Opcode分析等特征，并将结果存储到数据库中 | 通过 |
| 9 | 集成学习与检测模块 | 模型推理与恶意性判别 | 系统启动，用户上传了一个PE样本，特征提取完成 | 提取的特征数据 | 系统自动触发模型推理流程 | 系统将特征数据输入到CNN、集成模型、LightGBM等机器学习模型中，通过加权融合策略生成最终的恶意性判定结果，并将结果存储到数据库中 | 通过 |
| 10 | 数据可视化与前端展示模块 | 样本特征可视化 | 系统启动，用户登录成功，已上传并分析了一个样本 | 样本ID | 用户在样本详情页面查看特征可视化图表 | 系统通过ECharts图表库展示样本的直方图特征、PE静态特征、LIEF深度分析结果等可视化图表 | 通过 |
| 11 | 数据可视化与前端展示模块 | 仪表盘展示 | 系统启动，用户登录成功，已上传多个样本 | 无 | 用户在仪表盘页面查看全局统计数据和近期样本信息 | 系统通过Ant Design组件库和ECharts图表库展示总样本数、恶意样本数、正常样本数、整体检测率、样本分类饼图、样本大小分布柱状图等信息 | 通过 |
| 12 | 数据可视化与前端展示模块 | 模型分析数据展示 | 系统启动，用户登录成功，已上传多个样本 | 无 | 用户在模型分析页面查看各模型性能和特征重要性 | 系统展示各机器学习模型的性能指标和特征重要性排名 | 通过 |

### 改进方法

为提升系统的性能与用户体验，本系统仍然有部分改进空间。在特征提取层面，引入多线程和分布式计算技术，实现特征提取的并行化处理，显著提升处理效率和系统吞吐量。在模型优化层面，通过数据增强和迁移学习技术，增强模型对新型恶意软件变种的泛化能力。在用户体验层面，优化前端界面，简化操作流程，增加交互提示和实时反馈机制，使系统更易用、响应更及时。在系统安全层面，加强用户认证机制和异常处理机制，确保系统的安全性和稳定性。在数据库性能层面，采用高效的数据存储结构和索引机制，提升系统的扩展性和响应速度。

## 本章小结

本章介绍了基于集成学习的恶意PE识别系统的设计与实现，包括系统需求分析、总体架构设计、详细功能实现以及系统测试。系统采用前后端分离架构，结合React框架、FastAPI框架和SQLite数据库，实现了样本管理、特征提取、模型推理和数据可视化等功能。测试结果表明，系统运行稳定，功能完善，能够高效检测恶意PE软件，具有良好的应用前景。

# 总结与展望

## 总结

本文提出了一种基于集成学习的恶意PE软件特征检测与识别方法，旨在应对日益复杂的恶意软件检测挑战。通过对恶意PE软件的多维特征进行分析，结合直方图特征、PE数据结构特征、字符规则匹配特征等多种静态特征，本文构建了高效的特征检测方案。在此基础上，设计了基于多模型融合的恶意PE识别模型，包括CNN、Stacking和基于LightGBM的特征工程模型，并通过多层集成学习方案进一步提升检测性能。最终，实现了基于集成学习的恶意PE识别系统，采用React框架、FastAPI框架和SQLite数据库进行开发，实现了样本管理、特征提取、模型推理和数据可视化等功能。

通过实验验证，本文提出的方法在检测准确率、泛化能力和鲁棒性方面表现出色，能够有效识别经过混淆、加壳或采用非常规技术的恶意PE样本。系统测试结果表明，该方法在面对复杂恶意软件变种时仍能保持较高的检测精度，具有较强的实际应用价值。此外，本文的研究还证明了集成学习在恶意软件检测领域的有效性和潜力，为后续研究提供了新的思路和方法。

## 展望

尽管本文提出的基于集成学习的恶意PE软件检测方法已经取得了显著的成果，但仍有一些改进方向和未来研究的潜力。首先，在特征提取方面，可以进一步探索更多维度的特征，例如基于动态行为的特征、网络流量特征等，以丰富特征集并提升检测性能。其次，在模型优化方面，可以考虑引入深度学习中的迁移学习和强化学习技术，以增强模型对新型恶意软件变种的适应能力。此外，随着恶意软件攻击手段的不断演变，需要持续更新和优化检测模型，以应对新的威胁。

从系统层面来看，未来可以进一步优化系统的实时性和扩展性，以支持大规模样本的快速检测和分析。同时，结合云计算和分布式计算技术，可以进一步提升系统的处理能力和效率。最后，随着人工智能技术的不断发展，探索更多先进的机器学习算法和模型架构，如图神经网络、注意力机制等在恶意软件检测中的应用，将是未来研究的重要方向。通过这些改进和扩展，有望进一步提升恶意PE软件检测的准确性和效率，为网络安全领域提供更强大的技术支持。

致谢

本科四年，时光荏苒。从高中毕业的腼腆少年，到如今即将毕业的准毕业生，在上海电力大学的四年改变了太多。

还记得在2021年9月那会，我第一次踏足魔都的土壤，内心夹杂着不安与迷茫。我知道在高考失利之后，想要再向前冲是很困难的，但同时我也意识到，这也是一次很好的机会让我再一次面对自己，思考我还可以去努力再做到些什么。

首先我想感谢我的指导老师陈思远老师。不只是在完成本篇毕业论文时陈老师的悉心指导，在平时的学业上、生活上，陈老师一直是我的引路人，在我心中是“良师益友”这个词最好的体现。如果不是大一下学期陈老师引荐我参加CTF竞赛，我甚至可能不会参与转专业来到计算机学院。大二上学期末，我第一次参与科研课题，从选题到研究方法的确定，再到文章的撰写和修改，陈老师都耐心地为我答疑解惑，提出了许多宝贵的建议，这让我受益匪浅。大二大三学年，在陈老师的指导下，我们参与了一些网络安全相关竞赛、取得了一些成果，在后续白帽子社团的创立上，都离不开陈老师的帮助与支持。陈老师的严谨治学态度和丰富专业知识，不仅让我在学术上受益匪浅，更让我在为人处世上有了更高的追求。在此，我向陈老师致以最崇高的敬意和最诚挚的感谢。

同时，我也要感谢上海电力大学计算机科学与技术学院的所有老师，他们在课堂上传授给我的知识，为我完成本论文奠定了坚实的基础。感谢学院为我提供良好的学习环境和科研条件，让我能够顺利地完成学业。

感谢学院机器人实验室的于晓青老师、刘大明老师，以及一同参与工创竞赛的伙伴们，从初代到二代到三代，这个600x600x1000mm的垃圾桶，连同我们的汗水也顺利毕业了，感谢老师们平时无私的指导，感谢队友们的陪伴与付出。

感谢栗风永老师、张凯老师、王真老师、田秀霞老师、殷脂老师等计算机学院的所有老师，在我迷茫的时候，他们无私指导，指点迷津。总是在课后休息时间打扰他们，问一些刁钻问题，他们也总是能够耐心讲解，提供帮助。

感谢龚润杰、田志杰、翁国良等CTF黑客星球的伙伴们，他们带领着我参与了很多CTF竞赛，教会了我许多专业知识。

感谢钱耘云老师、王军平老师、易小兵老师等，你们上课真的很负责。

感谢孙宇翔、李敬儒两位同学，一同考研相互帮助。

感谢方文皓、吴清华、杨欣妍、刘思蓉等同学，还有我的室友们，在学习和生活上帮助了我很多。

感谢我的家人，他们的理解和支持是我完成学业的动力源泉。在我遇到困难和挫折时，他们总是给予我鼓励和帮助，让我能够坚持下去。

最后，感谢所有关心和支持我的老师、同学和朋友，他们的陪伴和鼓励让我在大学生活中度过了许多难忘的时光。特别感谢与我一起讨论学术问题、共同进步的同学们，你们的智慧和热情让我深受启发。再次感谢所有帮助过我的人，我将永远铭记你们的帮助和支持。

参考文献

1. Malware Statistics. AV-TEST. https://www.av-test.org/en/statistics/malware/. Feb. 2020.
2. Christodorescu M, Jha S. Static Analysis of Executables to Detect Malicious Patterns[C]. The 12th conference on USENIX Security Symposium - Volume 12, 2003: 12.
3. Christodorescu M, Jha S, Seshia S A, et al. Semantics-Aware Malware Detection[C]. 2005 IEEE Symposium on Security and Privacy, 2005: 32-46.
4. Zhang Q H, Reeves D S. MetaAware: Identifying Metamorphic Malware[C]. Twenty-Third Annual Computer Security Applications Conference, 2008: 411-420.
5. Schultz M G, Eskin E, Zadok F, et al. Data Mining Methods for Detection of New Malicious Executables[C]. Proceedings 2001 IEEE Symposium on Security and Privacy. S&P, 2002: 38-49.
6. Jurafsky D. Speech and language processing[M]. Pearson Education India, 2000.
7. Malone C, Zahran M, Karri R. Are Hardware Performance Counters a Cost Effective Way for Integrity Checking of Programs[C]. The sixth ACM workshop on Scalable trusted computing, 2011:71-76.
8. Xia Y B, Liu Y T, Chen H B, et al. CFIMon: Detecting Violation of Control Flow Integrity Using Performance Counters[C]. IEEE/IFIP International Conference on Dependable Systems and Networks, 2012: 1-12.
9. Demme J, Maycock M, Schmitz J, et al. On the Feasibility of Online Malware Detection with Performance Counters[J]. ACM SIGARCH Computer Architecture News, 2013, 41(3): 559-570.
10. Sayadi H, Mohammadi Makrani H, Randive O, et al. Customized Machine Learning-Based Hardware-Assisted Malware Detection in Embedded Devices[C]. 2018 17th IEEE International Conference on Trust, Security and Privacy In Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, 2018: 1685-1688.
11. Yerima S Y, Sezer S, Muttik I. High Accuracy Android Malware Detection Using Ensemble Learning[J]. IET Information Security, 2015, 9(6): 313-320.
12. Khasawneh K N, Ozsoy M, Donovick C, et al. EnsembleHMD: Accurate Hardware Malware Detectors with Specialized Ensemble Classifiers[J]. IEEE Transactions on Dependable and Secure Computing, 2020, 17(3): 620-633.
13. Pietrek M. An in-depth look into the Win32 portable executable file format, part 2[J]. MSDN Magazine, March, 2002.
14. Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features[C]//2015 10th international conference on malicious and unwanted software (MALWARE). IEEE, 2015: 11-20.
15. Pavlyshenko B. Using stacking approaches for machine learning models[C]//2018 IEEE second international conference on data stream mining & processing (DSMP). IEEE, 2018: 255-258.
16. Kuhn M, Johnson K. Applied predictive modeling[M]. New York: Springer, 2013.
17. Weinberger K, Dasgupta A, Langford J, et al. Feature hashing for large scale multitask learning[C]//Proceedings of the 26th annual international conference on machine learning. 2009: 1113-1120.
18. Fedosejev A. React. js essentials[M]. Packt Publishing Ltd, 2015.
19. Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 3, 1157–1182.
20. Tang, Y., et al. (2023). A Fast Malicious Code Detection Method Based on Feature Fusion. Acta Electronica Sinica, 51(1), 1–10.
21. Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems, 30.
22. Zhou, X., & Tong, X. (2021). Ultra-Short-Term Wind Power Combined Prediction Based on CEEMD-SBO-LSSVR. Power System Technology, 45(3), 855–864.
23. Christodorescu, M., Jha, S., Seshia, S. A., Song, D., & Bryant, R. E. (2005). Semantics-aware malware detection. In Proceedings of the 2005 IEEE Symposium on Security and Privacy (pp. 32–46). IEEE.
24. Tang, Y., et al. (2023). A Fast Malicious Code Detection Method Based on Feature Fusion. Acta Electronica Sinica, 51(1), 1–10.
25. LeCun, Y., et al. “Backpropagation Applied to Handwritten Zip Code Recognition.” Neural Computation, 1(4):541–551, 1989.
26. Ke, G., et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree.” NIPS 2017.