基于多特征融合和集成学习的

涉诈APP智能识别分析系统

**项**

**目**

**文**

**档**

2024年7月15日

**目录**

**[第1章 项目概述](#_Toc7518)** [1](#_Toc7518)

**[第2章 问题定义](#_Toc27824)** [3](#_Toc27824)

[2.1问题来源 3](#_Toc18900)

[2.2现有技术方案 3](#_Toc15949)

[2.3解决问题的思路 5](#_Toc13736)

[2.3.1总体思路 5](#_Toc10228)

[2.3.2作品功能 6](#_Toc999)

[2.3.4使用数据集 6](#_Toc25532)

**[第3章 总体设计](#_Toc1180)** [7](#_Toc1180)

[3.1设计方案 7](#_Toc20110)

[3.1.1静态分析 7](#_Toc25077)

[3.1.2动态分析 8](#_Toc12090)

[3.1.3通联地址分析 8](#_Toc24431)

[3.1.4特征工程 10](#_Toc15216)

[3.1.5模型构建 11](#_Toc893)

[3.2数据库设计 11](#_Toc29012)

[3.3系统架构 12](#_Toc1386)

[3.3.1前端技术 12](#_Toc14174)

[3.3.2后端技术 13](#_Toc26616)

[3.3.3系统流程 13](#_Toc21881)

**[第4章 详细设计](#_Toc22259)** [15](#_Toc22259)

[4.1基于Androguard和keytool的信息提取 15](#_Toc23745)

[4.2基于APKtool的静态资源扫描 15](#_Toc13334)

[4.3基于adb和Monkey的动态分析 16](#_Toc7563)

[4.4基于Aho-Corasick算法的地址过滤 17](#_Toc21952)

[4.5基于集成学习的研判模型 17](#_Toc18069)

[4.5.1数据处理 17](#_Toc553)

[4.5.2特征编码 19](#_Toc21986)

[4.5.3模型预测 19](#_Toc30030)

**[第5章 功能介绍](#_Toc32115)** [21](#_Toc32115)

[5.1 APK上传 21](#_Toc20282)

[5.2静态分析 22](#_Toc30100)

[5.3动态分析 23](#_Toc2712)

[5.4通联地址分析 23](#_Toc28514)

[5.5模型研判 24](#_Toc27879)

[5.6信息展示 24](#_Toc13534)

[5.7报告导出 25](#_Toc11282)

[5.8功能实现度汇总 26](#_Toc14869)

**[第6章 项目总结](#_Toc11243)** [27](#_Toc11243)

[6.1项目成果 27](#_Toc5372)

[6.2亮点与创新 27](#_Toc10108)

# 

# 第1章 项目概述

在智能手机和移动通信技术高速发展的时代背景下，各种终端应用软件应运而生。Android平台因其开放性和高效性的特点，深受用户和第三方开发人员青睐，由于系统生态的开放性，APP可以选择在正规应用市场进行上架，也可以通过分发平台等渠道进行推广、安装和使用。正是基于这一特点，电信网络诈骗活动的作案手法从最初的打电话、发短信，转向利用APP等网络工具进行诈骗，一些电信网络诈骗犯罪分子往往通过URL链接、二维码以及直接发送APK安装包等方式诱导受害人下载并使用非正常的APP。2020年以来，通过虚假APP实施的电信网络诈骗案件高发多发，严重侵犯人民群众财产权利和人身权利，严重威胁我国社会安全,成为全社会影响最大的犯罪形式之一。

电信网络诈骗犯罪运用通信信息网络开展犯罪活动,其虚拟性和易篡改性的特点，导致犯罪证据非常容易被犯罪分子毁灭,不易于取证和保存。APP涉诈、涉赌、涉黄、强制授权、过度索权、超范围收集个人信息的现象普遍存在。其中，网络兼职刷单、快速贷款、虚假投资理财等诈骗APP较多，特别是有一些仿冒各大银行和金融平台的APP具有较大迷惑性和欺骗性，给公安机关案件侦破带来巨大困难。侦查部门过去主要是对资金流和信息流进行研判,对涉诈APP相关信息缺乏知识支撑，面对勘查到的海量涉诈APP数据无处下手，导致大量有效信息被搁置。因此，有必要加强对涉诈APP相关信息的分析与研判识别。如何及时发现并遏制涉诈APP的传播，提高APP应用的合法合规使用，已成为移动互联网行业亟待解决的重要问题。

为解决这一问题，我们研发了“基于多特征融合和集成学习的涉诈APP智能识别分析系统”。该系统由APK采集模块、APK分析模块、APK研判模块、信息展示模块构成。系统综合采用了多种分析工具和手段，以自动化的形式，实现了从APK获取到分析研判结果展示的全过程。

APK采集模块负责收集APP的安装包文件，通过下载链接、下载二维码和本地文件上传三种方式采集APK文件。

APK分析模块负责从安装包中提取静态特征，如APP元数据、权限请求、文件结构等。通过动态分析工具运行APP，提取动态特征，如运行时行为、系统调用、网络请求等。

APK研判模块使用机器学习或深度学习算法对提取的特征进行训练，建立诈骗APP的识别模型。利用已知的诈骗APP和正常APP的数据集进行模型训练。

信息展示模块负责生成APP分析识别的信息，提高用户对APP类别的特征的感知能力。

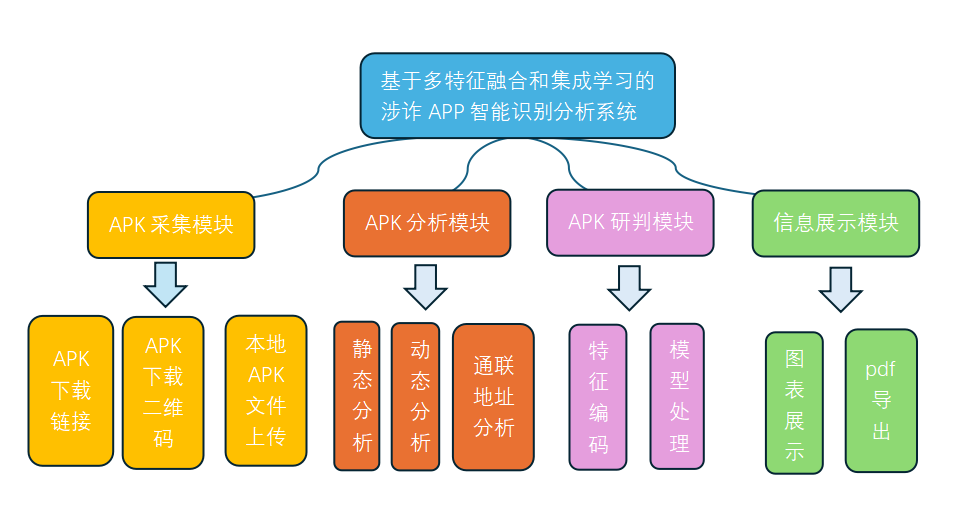


图1.1 系统模块构成

# 第2章 问题定义

## 2.1问题来源

近年来，跨境赌博、电信网络诈骗、黑灰产等外部欺诈违法犯罪形势日益严峻，呈现线上化、产业化、团伙化等特征，国家层面高度重视反欺诈治理工作，执法和监管机构对反欺诈管理要求日趋严格。党的二十大的报告中发布了“以人民为中心”、“统筹发展与安全”的重要指示。包括基础电信运营商在内的电信业务经营者需筑牢反欺诈“防火墙”，为业务的健康发展保驾护航；践行电信为民，坚决守护人民群众的财产安全，提高人民群众的幸福感、安全感和获得感；牢牢把握电信反诈工作的政治性、人民性，提升专业性。2022年12月，国务院正式施行的《中华人民共和国反电信网络诈骗法》，把反欺诈工作上升到了法律层面。

然而，针对如何发现非法APP的问题，目前的解决方案主要集中在利用工具进行人工静动态分析，例如APK Messenger、jadx-gui、Android逆向助手等。虽然不乏一些商业软件，如星源APP溯源分析平台、MSIE-20移动互联网APP检测取证系统、金斗云在线服务平台等。但是这些工具、平台或系统，以及它们之间组合使用的一些分析方法，都具有较强的局限性。同时，部分工具和方法操作复杂，对于需要侦破大量电信诈骗案件的公安机关人员来说，无疑是一项繁琐的工作。

因此，如何开发和运用有效的科技和大数据系统，分析涉赌、涉诈、涉黄和黑灰产等非正常APP的特征，优化风险监测模型，及时发现并遏制涉诈 APP 的传播，提高 APP 应用的合法合规使用，已成为移动互联网行业**亟待解决**的重要问题。

## 2.2 现有技术方案

**1、白名单正版APP签名证书对比方法：**通过对AndroidManifest.xml中的各项信息，初步筛选出第一级疑似涉诈APP，然后用白名单正版APP的签名证书排除正规的软件，获得第二级疑似涉诈APP。该方法基于简单的静态分析技术所提取的信息做出判断，效率较高，但受到白名单更新滞后的影响，存在误报、漏报风险。

**2、网络服务器更新特征库方法：**通过网络服务器更新TF-IDF特征词典和白名单正版APP签名证书特征，可以实时针对最新的关键词和排除正规的金融机构的APP，提高检测的准确性和实时性。然而，跨国、跨境电信网络诈骗的识别、拦截、追踪和定责必须在国际协同合作的框架下解决，这在一定程度上增加了治理的复杂性。

**3、图像识别与文本分析方法：**这种方法包括对运行的APP截屏，获得APP运行的界面图像，然后对界面图像进行图像识别，提取文本信息。对提取的文本信息进行分词得到词组，对词组进行分析计算，得出APP涉诈的可能性高低值。分析算法可以包括TF-IDF、WORD2VEC或BERT等。图像识别和文本分析的效果很大程度上依赖于训练数据的质量和数量。如果训练数据不充分或质量不高，模型的识别能力将受到限制。

**4、多模态融合识别方法：**通过获取APP的安装程序文件，对安装程序文件进行特征提取，得到安装程序文件的特征。然后对这些特征进行多模态融合，基于融合后的特征和涉诈特征库，对APP进行识别。涉诈特征库包括诈骗APP的原值特征集合和向量融合特征，以实现对诈骗APP的高效识别。多模态融合技术需要处理和整合来自不同模态的信息，如文本、图像、视频和音频等。这些模态之间的语义对齐和信息融合是一个技术难点。多模态图像识别任务通常需要大量的数据，但是在实际应用中，数据可能不足以训练一个高性能的模型。

上述Android软件分类识别方法，**主要用于解决二分类问题**，即正常APP与恶意APP的分类。对于识别多分类的APP上，目前没有提出有效的解决方案。本项目基于集成学习提出一种面向多分类的恶意APP检测与判别识别系统，基本原理是通过静态检测、动态检测等程序分析技术，提取不同的特征，描述待分析样本的不同特征与行为信息。譬如，通过逆向工程获取源程序代码、图像等文件,从文件中提取权限、API、组件等，以及在真实的Android设备、Android虚拟机或沙箱上运行应用程序,并对应用程序的真实行为进行监测,同时记录各个行为的数据,如短信发送、文件读取写入、API调用、权限调用等。然后每一个样本均用一个固定维度向量表示, 最后借助于现有的集成学习算法对已知标签的样本进行训练并构建基分类器和元分类器, 从而能够对未知样本进行多类别输出的预测判断。

## 2.3解决问题的思路

#### 2.3.1总体思路

基于现有的分析技术和Android软件分类方案，我们进行了整合与改进，并集成为一个Web系统，实现全过程自动化地分析和研判。

对于APP的分析，我们首先提供了三种APK采集方式：上传下载链接、上传下载二维码、上传本地APK文件。随后，针对采集的APK文件，依次开展静态分析和动态分析，从中提取有利信息，为后续研判做准备。在这个过程中，我们采用了一些功能强大、应用范围专注的工具进行综合处理，如Androguard、APKtool、Monkey等，以求全面提取到APP的信息。

对于APP的分类，我们设计了一个基于集成学习的研判模型。该模型在以往Android软件分类判别的思路上进行了改进，将二分类问题拓展到五分类问题，可以很好地适用于本项目所解决的问题中。首先，我们对APP分析过程中提取并筛选的有利信息做进一步的抽象，形成模型所需要的特征向量，随后输入到模型中，基于训练集对输入数据进行预测。

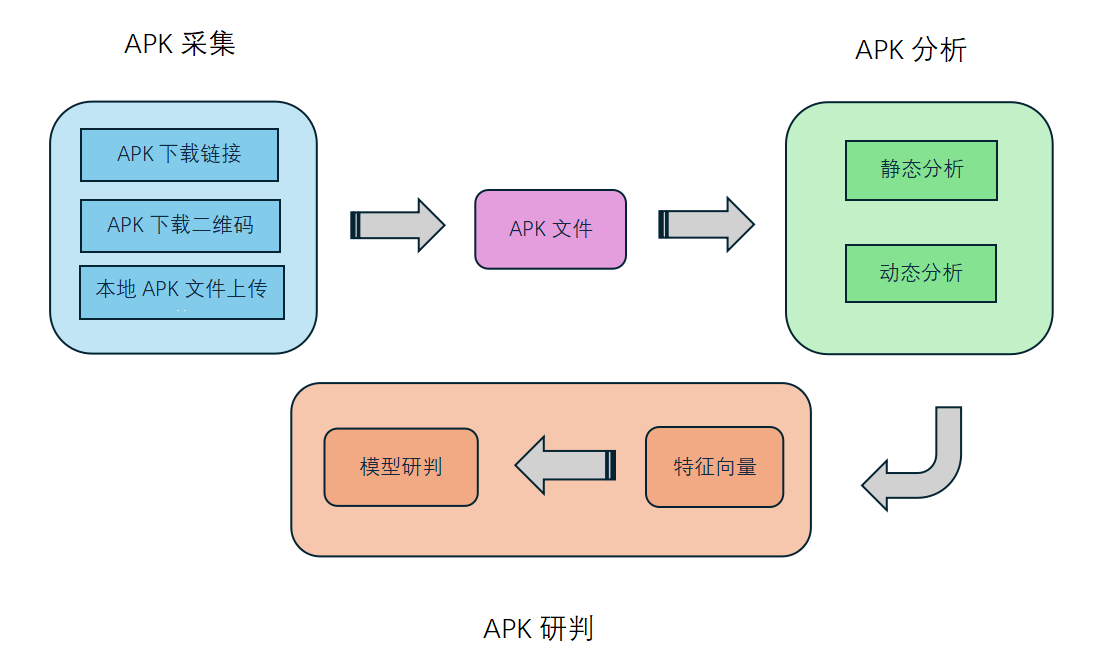


图2.1 总体思路

#### 2.3.2作品功能

**（1）APP采集**

系统可以通过基于APK下载链接、基于APK下载二维码和用户上传本地APK安装包三种方式采集APK。对于前两种方式，系统会针对最终获取到的下载链接是否以.apk结尾，判断是否采用多线程下载方式。

**（2）黑白名单配置和过滤**

如果APP在白名单中，系统将给出相应提示，并不再进行后续的分析和研判。否则，系统在给出相应提示的之后，会继续进行分析和研判，用户可进入系统查看详细的APP分析信息和研判过程。

**（3）APP分析**

系统将对采集到的APK进行静态分析和动态分析，用户可以看到实时分析过程。分析完成后，系统会将分析过程中获取到的信息，通过图表形式可视化展现。 在静态分析中，系统将全面解析APP安装包。获取应用程序名称、版本号、大小、文件MD5等基本信息，以及签名证书、静态图片资源和静态存在于文件中的通联地址。同时，系统将通过提取到的多维度数据，构建研判模型的特征向量。

在动态分析中，系统将在模拟器中自动安装运行卸载APP，随机模拟用户的操作过程，如点按、滑动等。随后，系统将对动态运行中产生的日志信息进行处理，提取动态运行时的通联地址。

**（4）模型研判**

在对APP进行静动态分析之后，系统会将构建好的特征向量输入到研判模型中，最终得到模型研判的结果，并展示给用户。用户可以在PDF导出功能中查看具体的模型研判的过程，包括APP的基本信息、特征向量构建过程、研判结果、相关的URL和IP地址等，检测研判结果的置信度。

#### 2.3.4使用数据集

本项目中使用的数据集来自中国移动官方提供的真实APK文件，这些APK按照五种类型进行了划分，我们从中分析提取并构建特征工程，得到模型所需要的训练集和测试集。

# 第3章 总体设计

## 3.1设计方案

#### 3.1.1静态分析

静态分析基于Androguard、APKtool和keytool工具，提取如下信息：APK文件大小、APK文件MD5值、APP图标、APP名称、版本名、版本号、SDK版本、包名、程序入口、签名证书、签名方案、四大组件、申请权限。同时，扫描反编译后的APK文件，提取所有图片资源，以及存在于文件中的所有通联地址。

Androguard是使用Python编写的，用于分析和逆向工程Android应用程序的工具，它主要用于静态分析APK文件，已集成在Python类库中。Androguard能够解析 Android 应用程序中的各种文件和组件，提取信息，并提供分析和调试功能。

APKTool是一款常用的反编译和编译Android应用程序的工具。它可以将APK文件解析成smali代码，并且可以对smali代码进行修改后重新编译生成新的APK文件，广泛用于Android应用的修改、调试和逆向工程。

keytool是一个用于管理加密密钥和证书的命令行工具，通常用于创建和管理Java KeyStore (JKS)文件。它是Java开发工具包 (JDK) 的一部分，并在处理加密操作和安全通信（如 SSL/TLS）时非常有用。

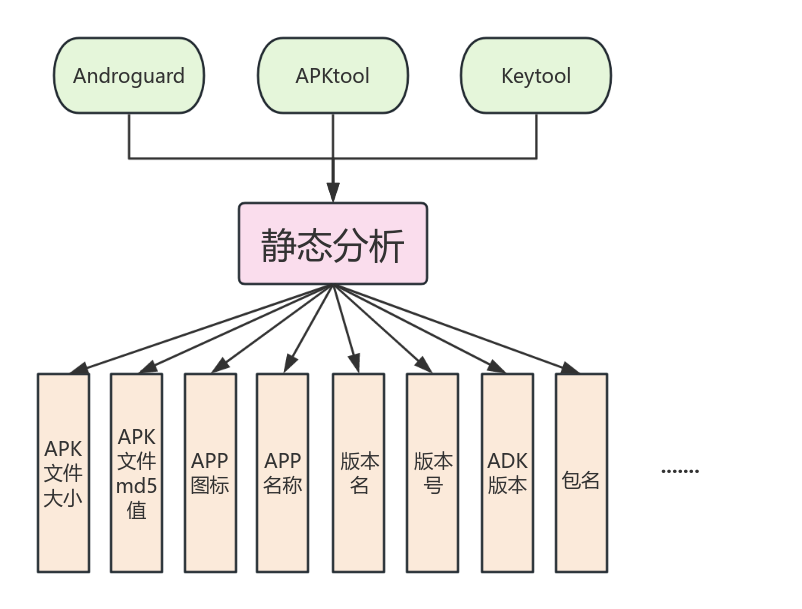


图3.1 静态分析方案

#### 3.1.2动态分析

动态分析基于安卓模拟器、adb和Monkey工具，用于模拟APP的运行。用户可以看到APP运行的过程，从人工的视角判断APP的类型，验证研判模型的检测结果。

adb 全称为Android Debug Bridge，是 Android SDK 提供的一个工具。adb采用C/S架构，Client端是在PC端的adb工具，通信内容可以是命令、文件、LOG。Server端解析来自于Client端的命令，并在模拟器上执行。使用adb命令可以完成APK文件从PC上传到模拟器、在模拟器中安装/卸载，以及导出APP运行过程中产生的LOG信息。

Monkey是Google提供的一套自动化测试方法，可以模拟用户触摸屏幕、滑动、按键等操作触发Android APP自动运行，可以通过控制随机事件的数量和频率，控制APP在模拟器中的运行时间。

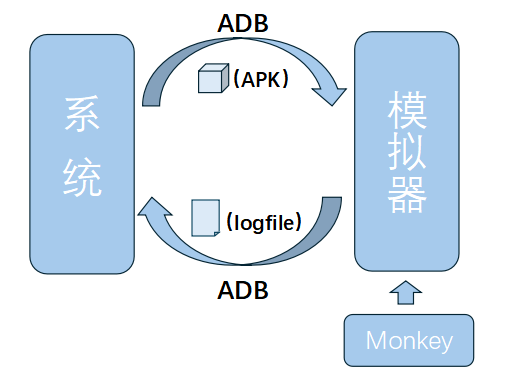


图3.2 动态分析方案

#### 3.1.3通联地址分析

对于通联地址，我们设计的方案是：静态扫描+动态捕获+图片提取（暂未实现）。

我们首先对反编译后的APK文件进行了扫描，提取静态存在于APK中的地址。

对于动态运行中的通联地址，我们采取的方案是：筛选出运行日志中出现的所有互联网访问活动，从中提取出网络访问地址。

起初，我们对于动态通联地址的捕获与提取，采用的策略是：启用tcpdump工具监听模拟器对应的端口，抓取该端口下所有网络数据包，随后导出.tcap文件，对该文件进行流量分析。经过大量的实验测试，我们发现实际的捕获效果并不好。在网络流量文件中，繁杂的协议交互、大量的CDN站点以及模拟器自身进程产生的地址，都对流量分析产生了极大的干扰。

因此，我们改进了策略，通过对运行APP运行过程的捕获，记录运行日志，从中着重分析网络访问的活动，提取访问地址。这样，我们极大地缩小了分析范围，实验显示该策略所提取到的通联地址十分准确。

同时，通过对APP的人工静动态分析，以及对APP图像资源的观察，我们发现更多以及更核心的通联地址，往往存在于图片当中，尤其是涉赌和涉黄类APP。因此，我们还提出了一种方案：对APP图像资源进行文本提取，从中筛选出IP和URL，作为最高优先级的通联地址。



图3.3 图片中的通联地址

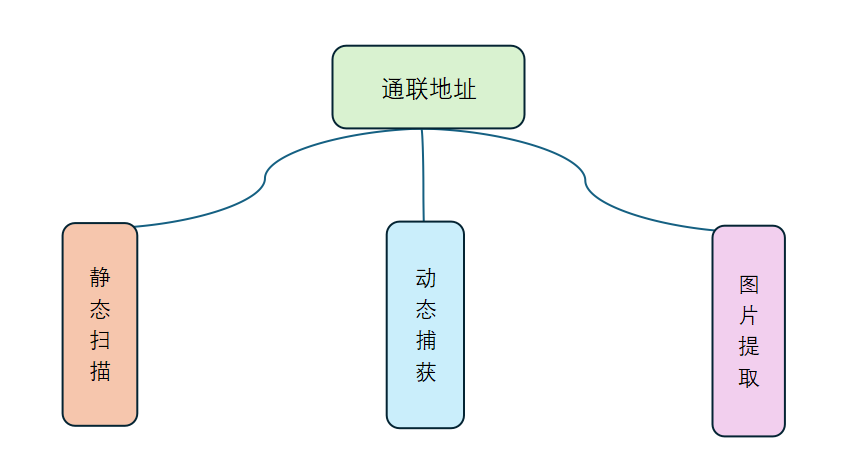


图3.4 通联地址分析方案

#### 3.1.4特征工程

我们基于静态分析获取到的多维度信息，进行进一步的筛选、补充和抽象表达。在静态分析提取的信息中，筛选出APK文件大小、APP名称、版本名、版本号、SDK版本、包名、程序入口、证书持有/发布者、证书有效期、签名算法名称、证书主体公钥算法位数、签名方案、四大组件、申请权限、是否白名单这几个维度，并将四大组件转化为组件总数、申请权限转化为权限总数，补充APK文件名长度、APK图片是否涉黄两个维度，再对APK中的所有字符串进行语义类别划分（暂未实现），总共18个维度作为特征工程。

在我们的方案中，签名证书占有很大的比重。Android系统要求每一个Android应用程序必须要经过数字签名才能够安装到系统中，Android通过数字签名来标识应用程序的作者和在应用程序之间建立信任关系。数字签名由应用程序的作者完成，并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证的。但是，对于正常的APP，签名证书可以验证应用程序的作者和确保应用程序在安装时没有被篡改。

我们发现较多的涉诈类APP都有一个共性，它们或仿照正常的APP制作，或直接篡改正常的APP。这将导致该APP的签名证书与正常APP不一致，或无法正常解析签名证书，这对于模型的研判而言至关重要。

我们还发现，除了签名证书的内容外，签名证书的方案也十分重要。Android 现在已经支持三种应用签名方案：

（1）v1方案：基于JAR 签名。

（2）v2方案：在Android 7.0引入。

（3）v3方案：在Android 9.0引入。

v1到v2是颠覆性的，为了解决JAR签名方案的安全性问题，而到了v3方案，其结构上并没有太大的调整，可以理解为 v2 签名方案的升级版。我们发现，正常的APP，尤其是与银行金融相关的APP，往往采用v3签名方案。v3签名方案相较其它两个更为复杂，也进一步确保了APP的正确性。

通过对签名证书的广泛了解，以及基于对多个不同类别的APK的对比观察，我们认为签名证书及签名方案对于APK类型研判十分重要，特别是区分涉诈类别的APK。

1表3.1 特征维度

|  |  |  |
| --- | --- | --- |
| APK文件大小 | 程序入口 | 四大组件总数 |
| APP名称 | 证书持有/发布者 | 权限总数 |
| 版本名 | 证书有效期 | 是否白名单 |
| 版本号 | 签名算法名称 | APK文件名长度 |
| SDK版本 | 证书主体公钥算法位数 | APK图片是否涉黄 |
| 包名 | 签名方案 | 字符串语义类别划分 |

#### 3.1.5模型构建

我们采用Stacking集成学习算法来构建针对APP类别的研判模型。

Stacking集成学习算法是一种叠加式分层集成算法。以两层为例，在第1层中使用不同的初级学习器对数据进行训练，在训练数据集时，为了避免因测试集比例划分过小造成泛化能力不强问题，通常采用交叉验证方式进行训练。然后将第1层产生的多次训练的数据结果作为新的训练集和测试集，第2层次级学习器采用新的训练集与测试集进行训练和预测。

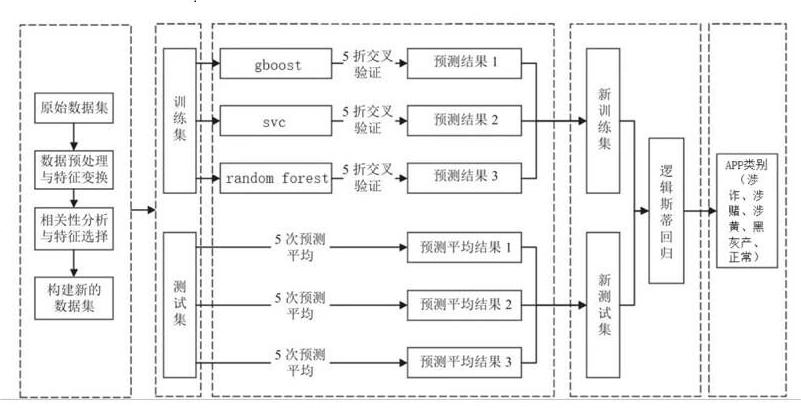


图3.5 模型构建思路

## 3.2数据库设计

**（1）黑白名单表**

本项目设置了一个白名单（whilelist）和四个黑名单（gamblelist、sexlist、scamlist、blacklist），均采用同样的组织方式，分别表示正常、涉赌、涉黄、涉诈、黑灰产APP。

以白名单（whitelist）为例：

表3.2 白名单表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段类型 | 释义 | 备注 |
| md5 | VARCHAR(32) | 文件MD5 | 主键 |
| appname | VARCHAR(1024) | APP名称 |  |
| packagename | VARCHAR(1024) | 包名 |  |

**（2）APP权限表（permission）**

该表存储了到目前为止Android系统的所有权限。

表3.3 权限表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段类型 | 释义 | 备注 |
| name | VARCHAR(50) | 权限名称 | 主键 |
| description | VARCHAR(1024) | 具体描述 | 非空 |
| level | VARCHAR(8) | 风险级别 | 非空 |

· level分为三种类型：正常、危险、未知。

## 3.3系统架构

本项目采用前后端分离的架构，前端基于vue和 element-ui等技术实现，后端基于Flask框架实现，采用MySQL进行数据存储。

#### 3.3.1前端技术

本项目的前端采用Vue作为框架，基于ES2015+、vue、vuex、vue-router、vue-cli、axios和element-ui实现，页面设计简单清晰，用户易于理解和操作。

Vue是一套用于构建用户界面的渐进式JavaScript框架，也是一个创建单页应用的Web应用框架。与其它大型框架不同的是，Vue被设计为可以自底向上逐层应用。Vue的核心库只关注视图层，不仅易于上手，还非常容易与第三方库或既有项目整合。

Element UI是一套基于Vue 2.0的桌面端组件库，提供了简单明了的设计指南,帮助产品设计人员搭建逻辑清晰、结构合理且高效易用的产品。

Axios是一个基于Promise的现代化HTTP客户端，可以在浏览器和Node.js 环境中使用，用于发送异步HTTP请求。Axios提供了简单、直观的API，使发送 HTTP请求变得更加简单和可控，可以通过简单的配置就能发送GET、POST、PUT、DELETE等请求。Axios自动将JSON数据转换为JavaScript对象，可以轻松的转换从后端传来的JSON数据以便于在前端页面上展示。

#### 3.3.2后端技术

本项目的后端基于Flask框架实现。Flask是一款基于WSGI的轻量级的Web框架，使用 Python语言编写，较其他同类型框架更为灵活、轻便且容易上手，小型团队在短时间内就可以完成功能丰富的中小型网站或Web服务的实现。

Flask具有很强的定制性，用户可以根据自己的需求来添加相应的功能，在保持核心功能简单的同时实现功能的丰富与扩展，具有很强的扩展性和兼容性。这对于像本项目这样需要灵活地集成分析工具和人工智能模型的Web系统来说，是十分友好的。

#### 3.3.3系统流程

用户上传APK下载链接、下载二维码或直接上传APK文件，随后服务器将对上传的内容进行处理：

（1）对于下载链接，首先判断是否以.apk结尾。如果是，则采用多线程下载方式下载APK文件；否则，先请求部分response数据，读取前几个字节并与APK文件的魔术数字 \x50\x4B\x03\x04进行比较，由此判断该链接是否是有效的下载链接，如果判断通过则继续下载。

（2）对于二维码，首先调用Open CV库对图片进行光学检测。如果图片有效，则从中解析出链接，采用上一步的方式处理链接。

（3）对于APK文件，直接保存到系统中。

服务器将对处理结果做出响应，并由前端反馈给用户。

当服务器采集APK成功后，客户端采用轮询的方式询问服务器分析进度。服务器首先获取APK文件的MD5值、包名和APP名称，随后根据这三个信息查询数据库。如果APK文件在白名单中，则不会进行后续的分析和研判，并告知用户。如果APK文件在黑名单中，服务器在告知客户端该APK属于黑名单之后，仍会进行后续分析和研判。

服务器在进行每一个具体分析步骤时，会同步告知客户端。当分析和研判成功后，用户可进入系统查看分析所得数据，以及PDF导出研判过程和结果。

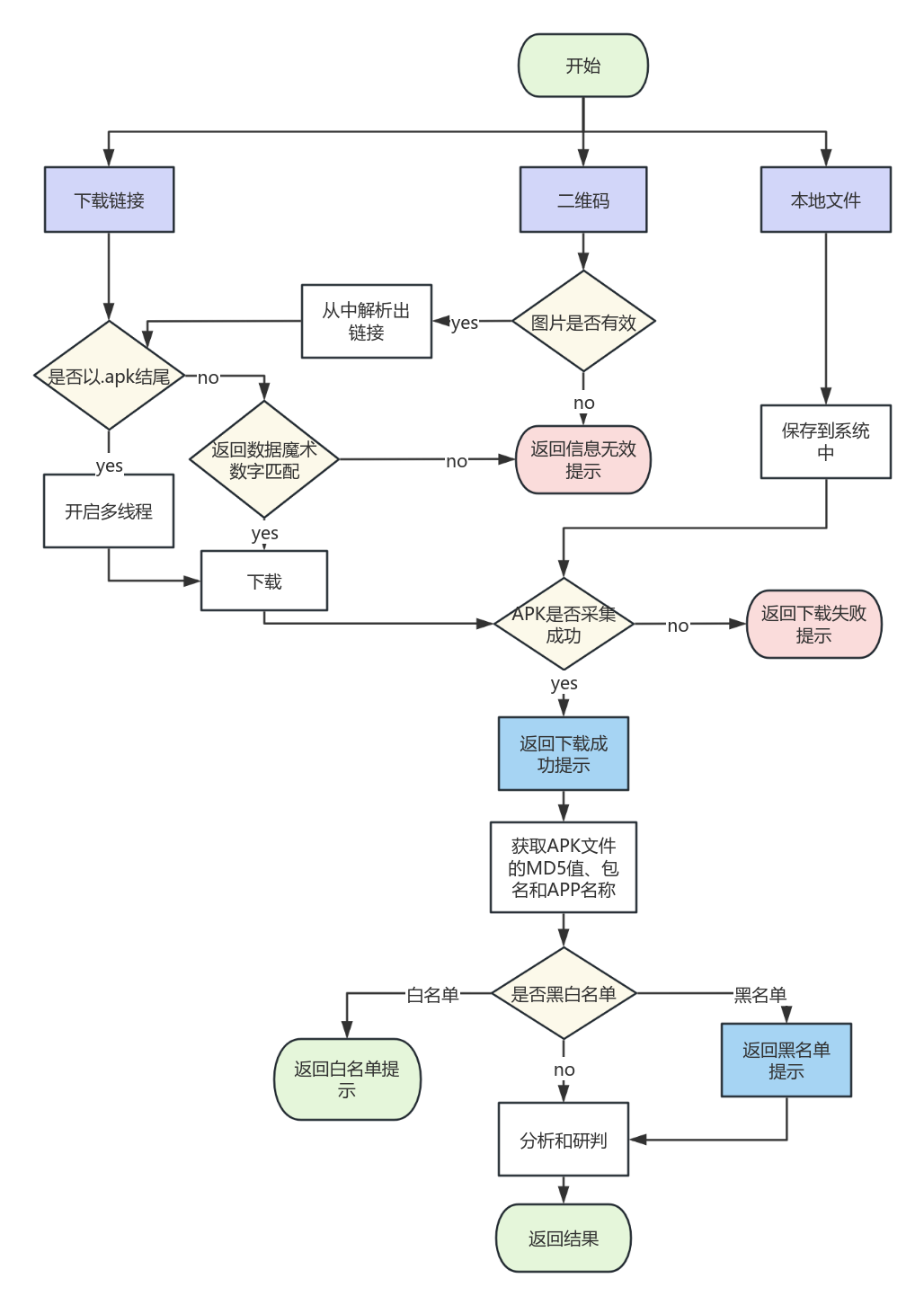


图3.6 系统流程图

# 第4章 详细设计

## 4.1基于Androguard和keytool的信息提取

由于Androguard已集成在Python类库中，我们直接从中调取APK方法解析APK文件，将得到一个APK对象。获取到APK对象后，可以直接调用相关方法，如get\_app\_name()、get\_package()、get\_activities()、get\_permissions()，获取到APP名称、包名、组件、申请权限等信息。

对于签名证书，Androguard虽然也提供了解析签名证书的方法，但解析的信息不够全面。我们采用keytool工具解析APK的签名证书，对其做进一步的处理，提取其中的有效信息。

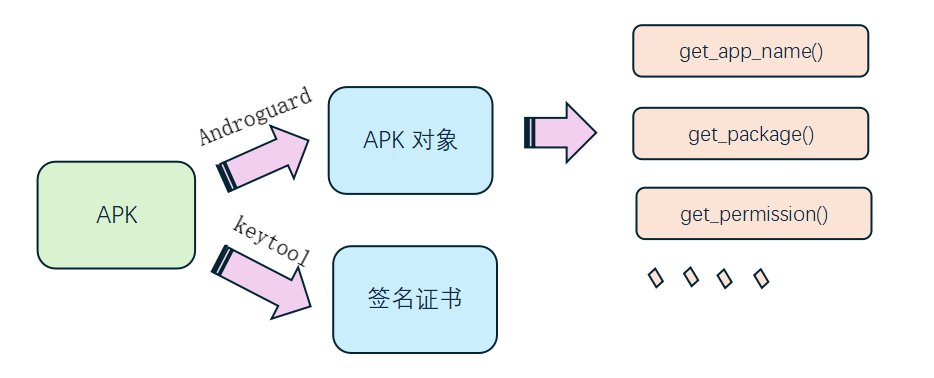


图4.1 信息提取

## 4.2基于APKtool的静态资源扫描

我们采用已集成到系统中的APKtool工具，用命令行的方式启动，对APK进行反编译，将其输出到指定文件夹中，随后对这些文件做进一步的处理。

首先解析APK签名方案。在META-INF文件夹下有三个文件：MANIFEST.MF、CERT.SF、CERT.RSA，它们是签名过程中生成的文件，其中CERT.SF文件中的X-Android-APK-Signed字段，标识了APK的签名方案，对该字段的值进行解析即可。

随后，我们对整个反编译的文件进行扫描。如果文件名以“.png”、“.jpg”、“.gif”、“.bmp”等图片后缀结尾，则将其提取出来，为后续模型研判中图片是否涉黄这个维度提供数据源。如果文件为文本类文件，则扫描文件内容，通过正则匹配的方式提取其中的IP地址和URL，为后续通联地址分析提供部分数据源。

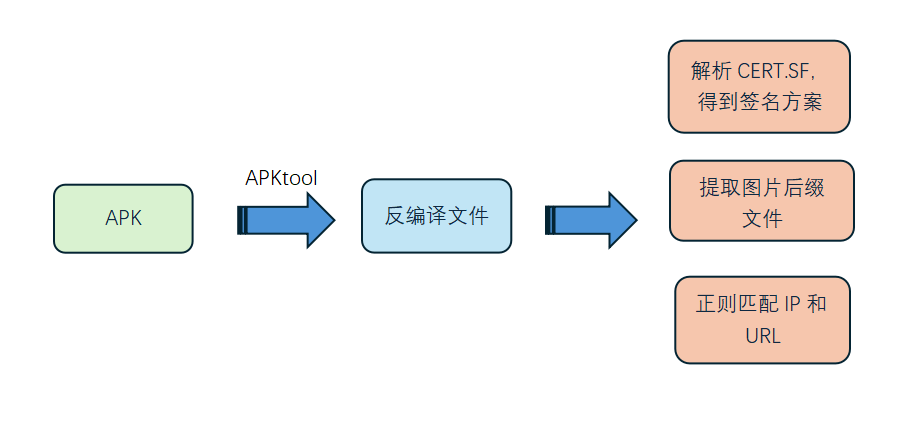


图4.2 静态资源扫描

## 4.3基于adb和Monkey的动态分析

在动态分析中，我们首先采用adb命令自动将模拟器提权至ROOT权限，随后创建多个线程，自动进行APK安装、权限授予、日志监听、导出日志、APK卸载等操作。

安装好APK后，用命令行调用Monkey工具随机模拟2000次用户的操作，如点击、滑动、输入等。随后导出该过程中的日志，对其中访问互联网的动作进行特别处理，提取通联地址，补充静态扫描中产生的数据源。

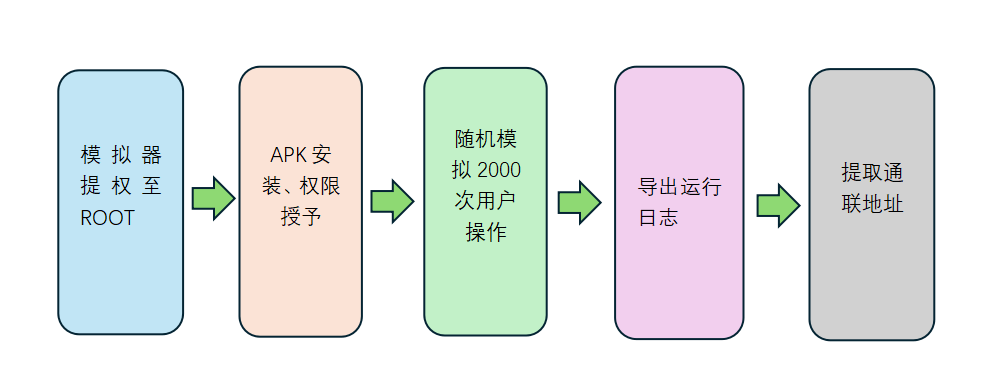


图4.3 动态分析流程

## 4.**4基于Aho-Corasick算法的地址过滤**

对于静态分析和动态分析中获取的IP和URL地址，我们先采用一定的过滤机制筛选出有效的地址。我们建立了一个包含大量正常字符串的字库，如“github”、“google”、“microsoft”，采用字符串匹配的方式消除正常的地址。

在这个过程中，我们使用Aho-Corasick算法进行字符串匹配。Aho-Corasick算法是一种适合多模式匹配的高效算法，可以在O(n+m) 时间内完成匹配，其中n是文本的长度，m是所有模式字符串的总长度。这比KMP算法在处理多个模式字符串时更有效。

对于筛选出来的地址，依次进行访问，获取返回的状态码或访问状态。由于非正常网址的生命周期一般较短，或者服务器架设在国外，国内无法访问。这都将通过状态码或访问状态告知用户。在这里，我们设置访问超时时间为3秒，以提高访问效率。

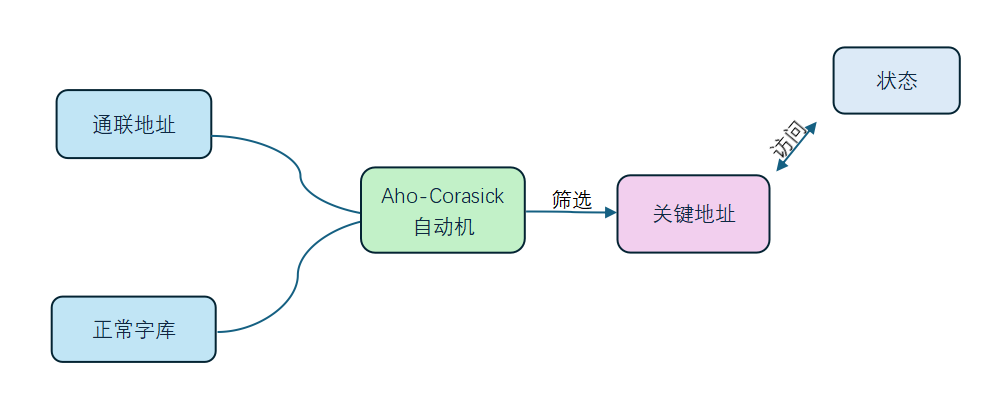


图4.4 通联地址分析

## 4.5基于集成学习的研判模型

#### 4.5.1数据处理

静态分析提取到的信息是多维度的，且是模型无法直接处理的数据。在数据处理的过程中，我们对多维度数据进行了补充、筛选和编码，得到模型所需要的输入向量。

我们补充了APK图片是否涉黄这个维度，这个维度有利于涉黄类别APK的鉴别。在静态分析时，从反编译文件中扫描提取的图片资源是庞杂的，并且大部分无意义，即图标类图片。我们从不同类别的APK中随机选取10个样本，观察其图片资源的分布特征，并按照图片文件大小进行排序，发现实际上有意义的图片集中在最大的几张或十几张图片中。

基于这个发现，我们设计了这样一个策略：挑选按照图片大小排序后的前7张图片，把这些图片通过用Tensorflow实现的雅虎Open NSFW模型处理，依次判断是否涉黄。为了兼顾模型的准确度，设置当涉黄图片数量大于等于2时，认为该APK的图片涉黄，否则不涉黄。

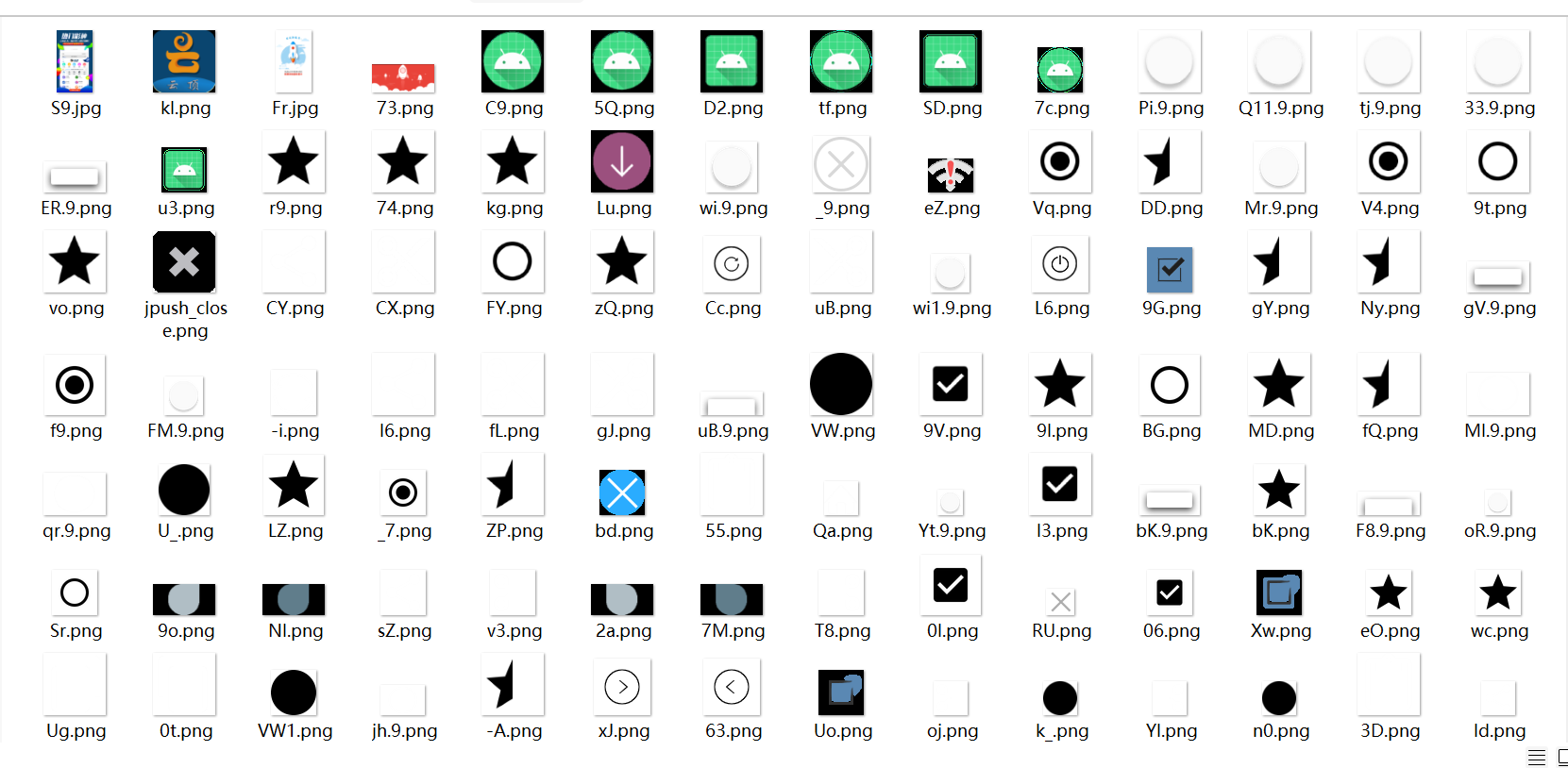


图4.5 排序后的图片

随后，对从大量数据集中提取到的多维度数据进行分类，按照每一个维度对比观察五类APK（涉赌、涉黄、涉诈、黑灰产、正常）的特征和区别，从中筛选有用的维度。这些维度及特征值整合为如下表格：

表4.1 特征维度

|  |  |
| --- | --- |
| **维度** | **特征值** |
| APK文件名长度 | 长度 |
| APP名称 | 有/无 |
| 版本名 | 无/低/高 |
| 版本号 | 无/正常区间值/非正常区间值 |
| 包名 | 有/无 |
| SDK版本 | 有/无 |
| 主活动 | 有/无 |
| 证书持有者和发布者 | 有/无 |
| 证书有效期 | 有/无 |
| 签名算法名称 | 有/无 |
| 签名主体公钥算法 | 无/1024位/2048位/其他位 |
| 签名方案 | 未签名/已签名 |
| APK大小 | 小于70MB/大于等于70MB |
| 组件总数 | 无组件/少于300/大于等于300 |
| 权限总数 | 无权限/少于45/大于等于45 |
| 图像是否涉黄 | 是/否 |
| 是否在白名单 | 是/否 |

#### 4.5.2特征编码

确定特征向量的维度后，我们对其进行编码。若维度的特征值为二元组，则编码0/1；若为三元组，则编码0/1/2。对于APK文件名长度，保留初始值，不对其进行更抽象的表达。

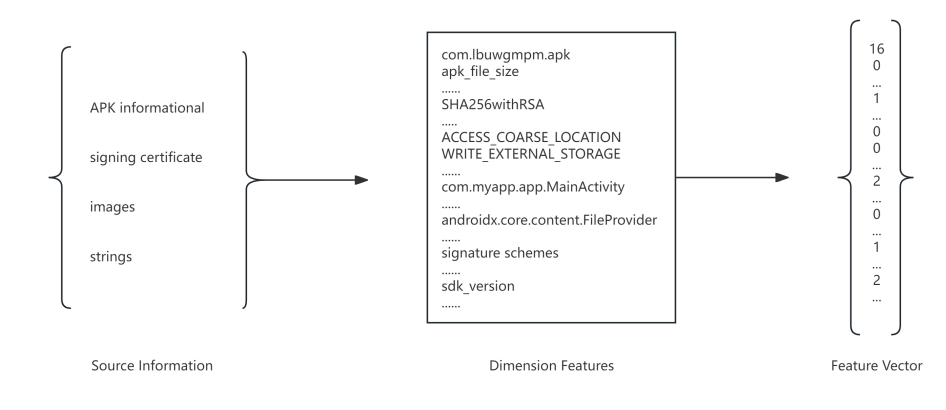


图4.6 特征编码过程

#### 4.5.3模型预测

构建好特征工程后，我们得到训练集和测试集。随后，将训练集分成train1、train2、train3、train4、train5。

接下来，选择GBDT、SVC和Random Forest这三种作为基模型。比如GBDT模型部分：依次用trian1、train2、train3、train4、train5作为验证集，其余4份作为训练集，进行5折交叉验证进行模型训练，再在测试集上进行预测。这样会得到在训练集上由GBDT模型训练出来的5份predictions，和在测试集上的1份预测值B1。将这五份纵向重叠合并起来得到A1，SVC和Random Forest模型部分同理。

三个基模型训练完毕后，将三个模型在训练集上的预测值分别作为3个“特征”A1、A2、A3，使用LR模型进行训练。

在对数据进行预测时，使用训练好的LR模型，在三个基模型之前在测试集上的预测值所构建的三个“特征”的值(B1,B2,B3)上，进行预测，得出最终的预测类别。

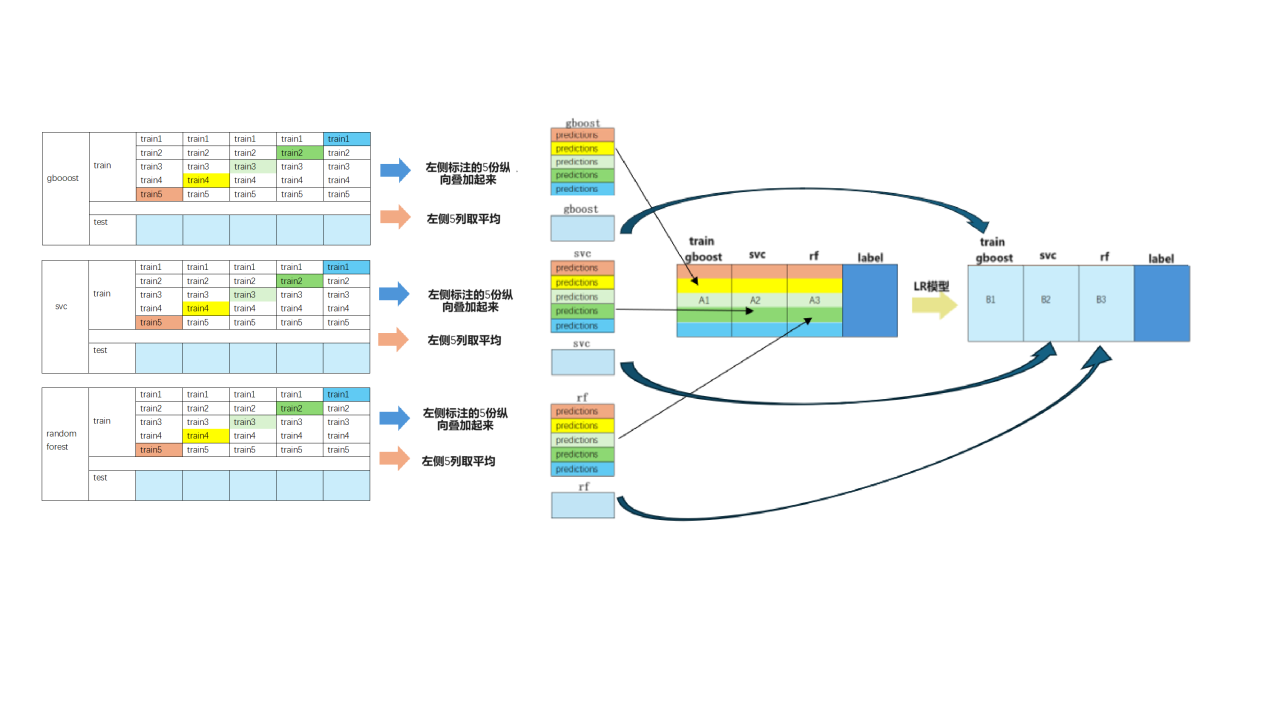


图4.7 模型预测流程

# 第5章 功能介绍

## 5.1 APK上传

APK上传页面提供了三个按键：本地文件上传、二维码下载、apk链接下载，为用户提供了三种上传APK的方式。



图5.1 三种上传方式

点击按键会出现相应弹窗，用户根据对应的方式上传内容后，服务器将开始采集APK文件。服务器采集APK成功或失败都将告知用户。

为了方便用户的使用，我们设计了许多便捷的操作方式：

1. 在本地文件上传窗口中，用户可以查看上传的进度；



图5.2 上传进度查看

（2）在二维码上传弹窗中，用户可放大或缩小图片，将二维码调整至正确位置。



图5.3 二维码缩放

## 5.2静态分析

APK采集成功后，服务器将进行静态分析。静态分析按照以下几个步骤进行：

（1）准备阶段：获取APK对象，如果获取失败，表明该文件不是APK格式，直接退出。否则，提取APK文件MD5值、APP名称和包名，根据这三个信息查询数据库，辨别是否黑白名单。

（2）解析APK基本信息：使用Androguard工具，调取对应方法获取版本名、版本号、SDK版本等信息。

（3）解析签名证书：使用keytool生成APK签名证书，提取其中的有效字段值

（4）解析签名方案：使用APKtool反编译APK，解析CERT.SF文件，得到APK签名方案。

（5）扫描APK文件：对反编译后的文件，扫描提取图片资源和地址信息。

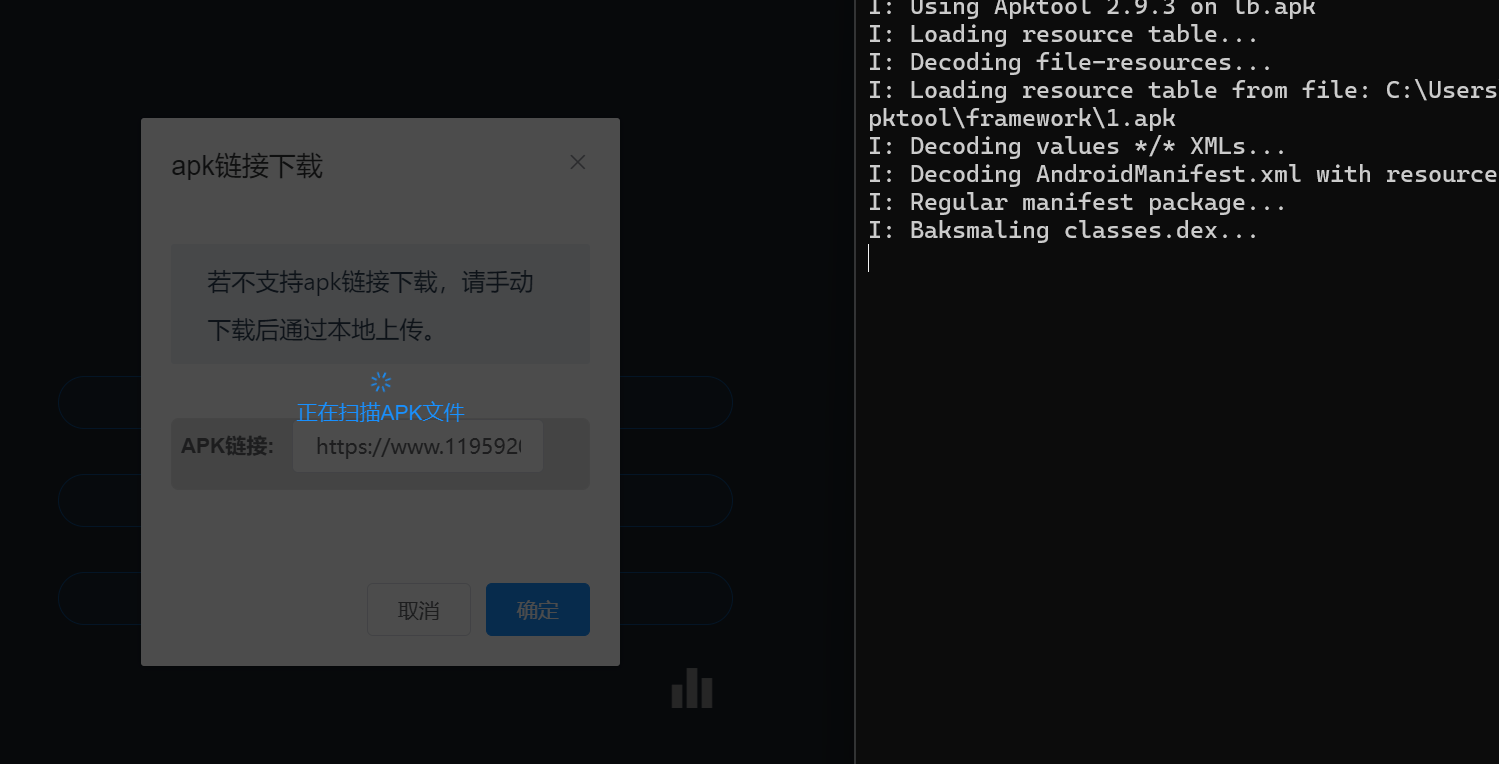


图5.4 正在进行静态分析

## 5.3动态分析

静态分析结束后，立即进行动态分析。动态分析按照以下几个步骤进行：

（1）准备阶段：检查APK包名及程序入口是否正常获取到。如果获取成功，则进行后续步骤；否则，系统将跳过动态分析。

（2）动态运行：使用adb和Monkey工具自动化安装、运行和卸载APK，记录运行日志，导出日志并提取通联地址。

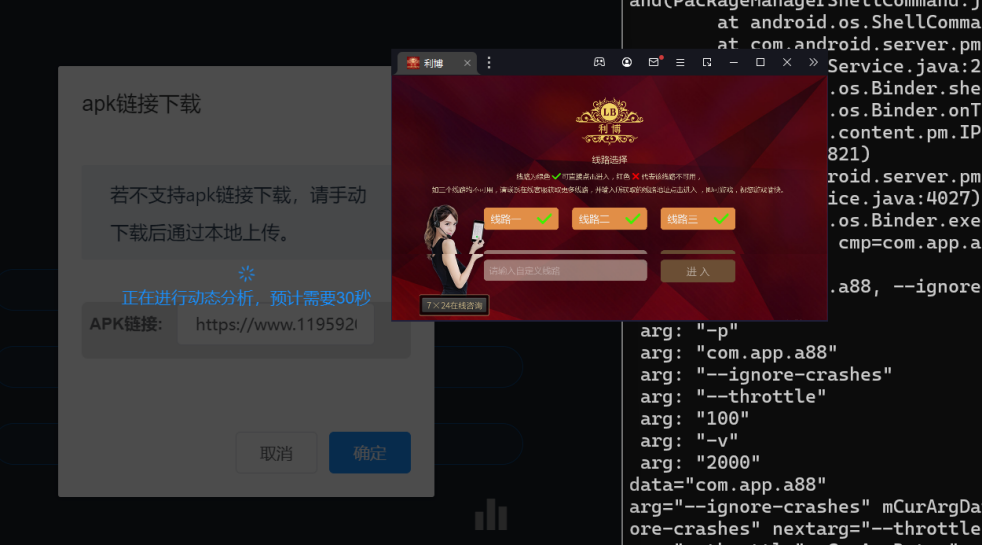


图5.5 正在进行动态分析

## 5.4通联地址分析

在静态分析和动态分析结束后，系统将进行通联地址分析，生成核心的关联网址及对应的访问状态。

首先，系统对静态分析和动态运行得到的地址进行排序，后者的优先级更高。随后，采用基于Aho-Corasick算法的过滤机制进行字符匹配，检查系统内置的正常字符串字库中，是否有某个字符模式存在于地址中，由此过滤掉正常的地址。

在过滤机制中，系统构建了一个 Aho-Corasick 自动机，自动机中存储了字库中的模式字符串。随后，使用 Aho-Corasick 自动机进行匹配，调用自动机的 iter() 方法来检查 site 字符串中是否包含字库中的任何模式字符串。最后，逐个访问筛选出来的通联地址，获取返回状态码或访问状态。

## 5.5模型研判

在分析过程结束后，系统将对APP进行类别研判。基于静态分析中获取的多维度数据，系统先按照一定的抽象和编码规则，生成初始向量。随后，系统调用Open NSFW模型，对AP图片资源进行涉黄检测，补充初始特征向量。最后，将特征向量输入到已集成到系统中的研判模型，输出APP研判结果。至此，整个APP的分析和研判过程结束。

## 5.6信息展示

当整个分析和研判过程结束后，用户可进入系统首页查看分析结果。在这里，系统对分析提取到的信息进行了可视化呈现，以图表的形式展现信息，可以让用户快速地了解APK文件中所包含和关联的内容。

信息展示页面分为六个部分对结果结果进行呈现：

（1）文件信息和APP信息

包含APP图标、APK文件名称、APK大小、APK文件MD5、APP名称、包名、SDK版本、版本名、版本号。

（2）签名证书

包含签名方案、所有者、发布者、序列号、有效期、证书指纹（MD5、SHA1、SHA256）、签名算法、主体公钥算法、版本。

（3）APP权限

以表格的形式展现APP申请的权限、权限的具体描述，以及权限的风险等级。权限分为正常、危险、未知三种风险等级，分别用不同的颜色显示在列表中，清晰明了。同时，支持以XLXS表格形式可导出。

（4）程序入口

（5）APP组件

按照组件的四大类型：Activity、Provider、Receiver、Service，分别以列表的形式展现。

（6）通联地址

以表格的形式展现APP的通联地址及其访问状态。其中URL链接可以点击跳转，访问状态用不同的颜色加以区分，可以让用户更直观地了解通联地址的信息。同时，支持以XLXS表格形式可导出。

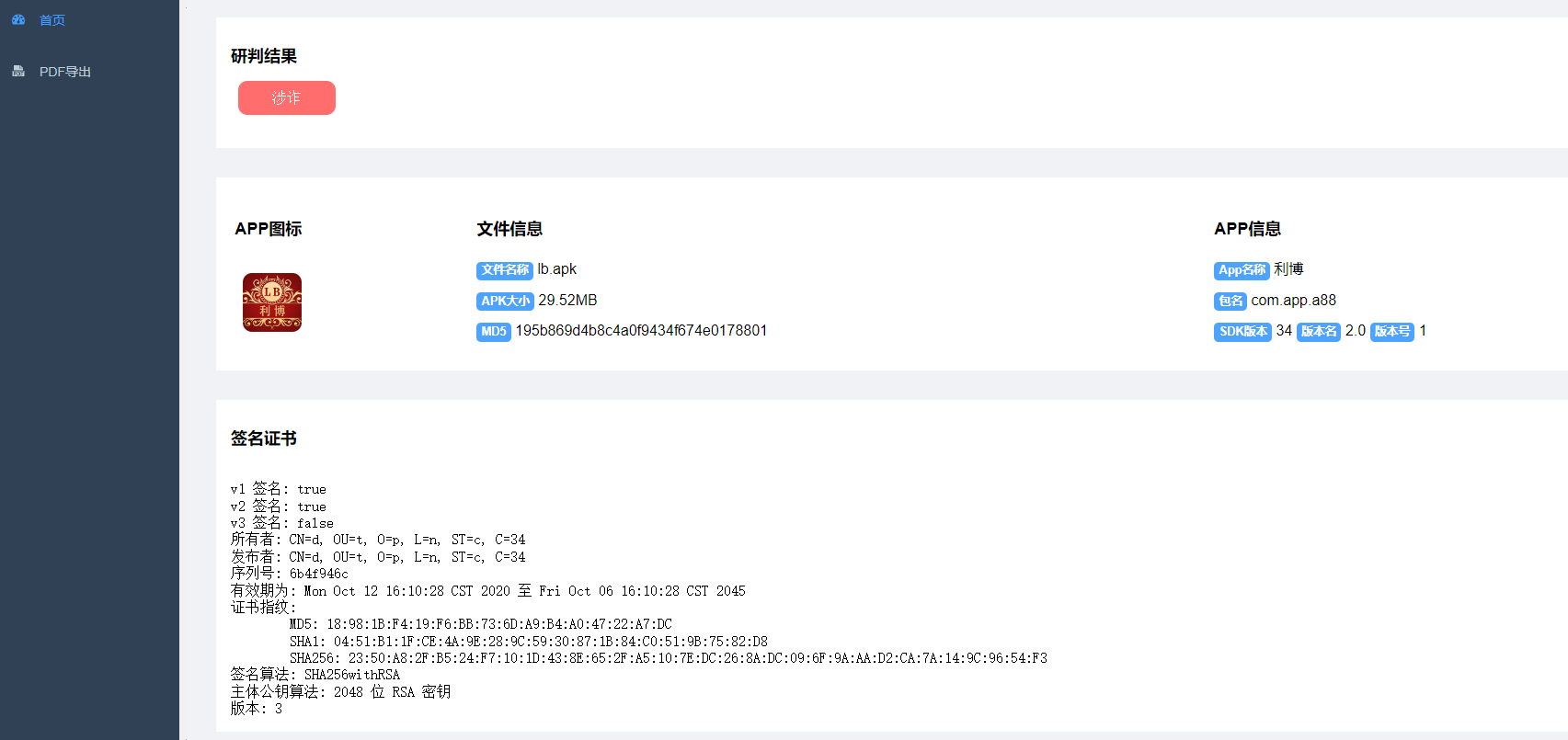


图5.6 信息展示（部分）

## 5.7报告导出

在信息展示界面的侧边栏，系统提供了PDF导出功能。用户只需点击对应的按键，系统将自动生成分析研判过程和结果的PDF报告。

在报告中，按照分析信息、特征抽象、特征编码、模型研判的顺序，给出了系统分析和研判的具体过程。用户可以了解具体的检测方法，判断结果的置信度，评估系统的性能。

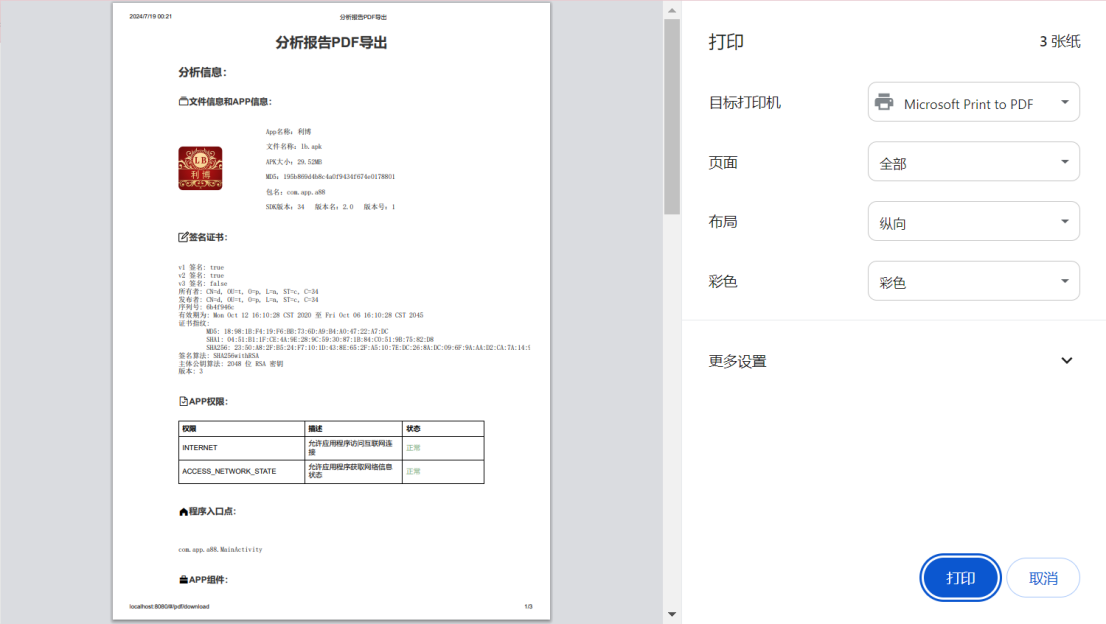


图5.7 PDF导出

## 5.8功能实现度汇总

表5.1 已实现功能汇总表



# 第6章 项目总结

## 6.1项目成果

本项目综合使用了多种静态和动态检测技术，对APK文件进行全面的信息分析和特征提取。并利用维度特征融合与集成学习构建了一个面向分类识别的APK研判模型，能够对APP在涉赌、涉黄、涉诈、黑灰产、正常五种类别中做出准确判决。

本项目弥补了当前技术背景下，APK文件分析检测和分类判决技术的不足，综合利用多种技术手段，实现了对APK的全自动分析，极大地减少了人力和物力消耗，并实现了APP类别的精准判别。为用户提供了一套便捷高效的Web系统，对待安装的APK文件进行快速分析和研判。

## 6.2亮点与创新

（1）全过程自动化的分析研判：通过集成多种分析工具，结合研判模型，实现对APK文件的自动化分析和类型判别。

（2）多维度的特征工程：基于静态和动态检测手段，提取包括图片、文本等多种信息，从中筛选特征明显的维度，并补充了图片是否涉黄、APK中文字信息的语义类别等维度，对APK类别进行准确有效的表征。

（3）机器学习库与计算机视觉库的应用：在APK采集方式中，我们使用了OpenCV对二维码进行光学识别，检测图片的有效性；在特征向量构建中，我们使用了Open NSFW模型对APK图片资源进行涉黄鉴别；在APP类型研判中，我们使用了集成学习模型，对APP类型进行准确判别。

（4）友好的操作界面：提供了包括APK文件上传进度查询、二维码的缩放、实时分析研判进度的查看、分析信息的分类展示、检测结果的PDF导出等在内的多项功能，使得用户的操作更简单、便捷、高效。